Tartalomjegyzék

1. fejezet

PHP-bevezető

Áttekintés

Üdvözlünk a PHP világában! Ha jelenleg valamely más szkript nyelven programozol, akkor itt az ideje, hogy élvezd a világ legjobb webes szkript nyelvének előnyeit. Még ha a szkript nyelvektől oly távol eső programnyelvet használsz is, mint az Assembler vagy a C, munkád úgy felgyorsul majd, mint Kirk kapitány, amikor bringájáról átszállt az Enterprise űrhajóra.

Jómagam azért kezdtem PHP-t használni, mert Periben írni néha rettentően unalmas volt, a Visual Basic pedig csak egy platformon elérhető. A PHP mindkét problémán segít. A 20. század végén a Visual Basic volt a világ legnépszerűbb programnyelve, akár a fogorvosok is írhattak vele fogászati alkalmazásokat. A Visual Basic a Basic nyelvhez kötődik, de a Basic család összes tagja különbözik valamiben, és még a Visual Basicnek is számos inkompatíbilis változata létezik.

A PHP-nek csak két változata terjedt el: a PHP3 és a PHP4. A PHP4 világszerte gyors ütemben váltja fel a PHP3-at.

A webszervereken kívül futó segédprogramok írására a Perl alkalmasabb, mint a PHP, jóllehet lehetőség van a PHP kibővítésére és ezen segédprogramok nagy részének helyettesítésére. Ha egy ideig használod a PHP-t, egyszerűbbnek fogod találni a PHP kiterjesztését, mint a Perire való visszatérést.

Korábban Assemblerben és Cobolban programoztam nagygépes szervereken. Az Assembler-hívők a gyors végrehajtás érdekében mindent Assemblerben akartak megírni, a Cobol-hívők pedig a gyors programozás miatt szerettek volna mindent Cobolban megcsinálni. Akkoriban a számítógépek annyival lassabbak és drágábbak voltak, hogy a műveletek végrehajtásának sebessége még kényes témának számított: egy jelentős processzorfejlesztés húsz programozó éves fizetésébe került. Most viszont a processzor sebességének nagymértékű növelése annyiba kerül, mintha egy hétre bérelnénk egy programozót. A gyors programvégrehajtás így nem téma többé.

A PHP3 elég lassú volt ahhoz, hogy mentségül szolgáljon a Perihez értő programozóknak, miért nem váltottak. A nagy teljesítmény eléréséhez azonban már nincs szükség a Perire.

A C az egyetlen a programozáshoz szükséges másik nyelv, és a C is csupán a PHP-hez való operációs rendszerek és fordítóprogramok megírásához kell. Ha értesz a PHP megírásához használt ANSI C-hez, akkor a PHP forráskódjával új változatokat hozhatsz létre, vagy fejlesztheted a jelenlegit.

A hozzám hasonló 99,75% , aki nem ismeri a C-t, sokrétűbbé teheti a PHP-t különféle kódok, függvények és objektumok összegyűjtésével. Számtalan honlapon előre megírt alkalmazások és komponensek gyűjteményeit találjuk, amelyek saját célra felhasználhatók. A Freshmeat **(freashmeat.net)** és a Sourceforge **(sourceforge.net)** jó kiindulási pont lehet mindehhez. Ha mondjuk PHP-alapú e-mail-klienst vagy -szolgáltatást szeretnél létrehozni, keress rá a Freshmeaten a „php email" kifejezésre. A könyv írása idején ez a keresés 56 találatot eredményezett, amelyek között szerepeltek alkalmazások, objektumok és fejlesztés alatt álló kódok béta-verziói is.

Programozási 1x1

Ha egyenesen a HTML-szerkesztésből csöppensz bele a PHP-programozás világába, hasznos lehet megismerned a szkript nyelveket és fejlődésük történetét. Ha esetleg programoztál már más nyelveken, akkor mind a GML-ről, mind a nyílt forráskódról szóló információ segíthet a PHP működésének megértésében.

Egy kis történelem

Charles Babbage 1833-ban réz- és ónkerekekből kezdett számítógépet építeni. Ez akkoriban, amikor Assemblerben kezdtem programozni, egy kicsit primitívnek tűnt számomra, mostanában viszont a modern programnyelvek és az Assembler közti különbség tűnik akkorának, mint a réz és a modern számítógépek szilikonchipje közti különbség.

Mintegy 30 programnyelvben dolgoztam, és írtam is egypárat, de jó néhányszor belefutottam az egyes cégek által kifejlesztett saját tulajdonú programnyelvek, illetve az egyetlen programozó által írt nyelvek okozta megszorításokba. A nyílt forráskódú szoftverfejlesztés lényegében megszüntette ezeket a korlátozó tényezőket.

Ha egyetlen ember fejleszt egy nyelvet, szorosan ellenőrzése alatt tarthatja azt, és képes a számítástechnikának új irányokat adni. Az IBM-nél 1969-ben dr. Charles Goldfarb vezette azt a kutatócsoportot, amely a Generalized Markup Language-t (GML), majd később a Standard Generalized Markup Language-t (SGML) kifejlesztette. Az SGML a Hypertext Markup Language (HTML) őse. Elmondhatjuk, hogy Dr. Goldfarb új irányba terelte a számítástechnikát.

A nagy vállalatoknál kifejlesztett saját tulajdonú programnyelvek ezzel ellentétes irányba haladnak. Az IBM két olyan termékével is dolgoztam, amelyek ugyanazt a piacot célozták meg, de mind a kettőt korlátozta, hogy nem voltak kompatíbilisek az IBM összes termékével. Az IBM szakemberei saját érdekeiknek megfelelően használták a fejlesztési projekteket, ami aztán mindkét termék végét jelentette. Nyílt forráskódú fejlesztés esetén nincsen .ehetőség az egyéni érdekek előtérbe helyezésére, hiszen ez a fejlesztési módszer lehetővé :sszi, hogy a programozók versenyeztessék saját ötleteiket, a felhasználók pedig eldöntik, nogy nekik melyik változat felel meg. Végeredményben a kevésbé hasznos termék beolvad más termékekbe, vagy háttérbe szorul, hiszen helyette a fejlesztők a népszerűbb termékek :ovábbfejlesztésével vagy teljesen új ötletek kidolgozásával foglalkoznak.

A nyílt forráskód megjelenése

1992-ben Linus Torvalds a Linux megírásával előtérbe helyezte a nyílt forráskódú szoftvereket. A Linux egy olcsó PC-n is futtatható ingyenes operációs rendszer. Emberek ezrei töltik szabadidejüket a Linux fejlesztésével, hogy emberek milliói ingyenesen használhassák.

1994-ben Rasmus Lerdorf kifejlesztette a PHP/FI-t, ami a nyílt forráskódú fejlesztéseknek köszönhetően PHP4-gyé fejlődött. A PHP4 egy olyan nyílt forráskódú nyelv az új webfejlesztők számára, amely folyamatosan váltja fel a Perihez hasonló régebbi nyelveket.

Fordítás és fejlesztés

A kezdeti programnyelveket papíron fordították le, és bináris formában gépelték be. Később megjelentek az automatikus fordítóprogramok, amelyek az ember által

értelmezhető nyelvet a számítógép számára megfelelő bináris nyelvre fordítják. A fordítás csökkentette a feldolgozási időt, de hosszadalmas késlekedést okozott a programozóknak.

A szkript nyelvek az egyszeri fordítást olyan értelmezési folyamattal helyettesítik, amely minden egyes alkalommal lezajlik, amikor a számítógép a szkriptet beolvassa. A programozás ezáltal felgyorsult, de a szkriptek lassabbak lettek.

Az 1980-as évek elején a szkript nyelveket fejlesztők fordítóprogramokkal kísérleteztek. Bizonyos szkriptértelmezők voltaképpen a memóriába fordították a szkriptet, így a szkript gyakran használt részeit csak egyszer kellett fordítani, és sokkal gyorsabban futottak. Néhány szkriptfejlesztő továbbment, a lefordított memóriatérképeket lemezre mentette, így a következő futtatások gyorsabbak lettek.

A 1990-es évek végén a webszerver nyelvek hasonló fejlődésen mentek keresztül. A Perit és az első PHP-t - mivel HTML-oldalak voltak - soronként értelmezték. A fejlesztők később olyan rendszereket választottak, amelyek a memóriába fordítottak. A PHP3 és a PHP4 közötti óriási teljesítménykülönbség az értelmezésről a fordításra való áttérésnek köszönhető.

Mind az Apache, mind a PHP számára léteznek már különböző cache-mechanizmusok, amelyek elmentik a lefordított memóriatérképeket a következő futtatások számára. Attól függően, hogy a szknpted hogyan működik, a webszervered jócskán felgyorsulhat a cache-szoftver használatától, de le is lassulhat, mert a pótlólagos szoftverráfordítás révén elillan a megtakarítás.

Költségek

A nagy hálózatokat kiszolgáló első szerverek nagygépes IBM-szerverek voltak, amelyek 10 000 000 dollárba vagy annál is többe kerültek. Amikor az első IBM nagygépes szerver klónját az IBM-mel versenyeztettem, volt szerencsém 60 százalékot lealkudni a szörnyeteg árából.

Ma viszont egy 400 dolláros, memóriával jól megpakolt, használt 486-os is működhet szerverként. Természetesen, ha nagyra törő terveid vannak, a legújabb PC-re van szükséged, amibe annyi memóriát préselsz, amennyi csak belefér. Ha a honlapod igazán beindul, akár 10 000 PC-re is szükséged lehet.

A nyílt forráskód használatáról

Ha nyílt forráskódú szoftvert használsz, mindig olvasd el a licencfeltételeket. Több száz PHP-alapú nyílt forráskódú alkalmazást tartalmazó oldal létezik, mindenféle licencekkel. A felhasználási feltételek a „Ezt a valamit a főiskolán csináltam. Tessék, használd ingyenesen"-től a „Szerzői jog fenntartva. Csak akkor használhatod, ha ..."-ig terjednek; utóbbi esetén lehet, hogy meg kell őrizned az eredeti kódot az eredeti csomagban, teljes dokumentációval és szerzői jogi figyelmeztetésekkel.

Vannak *memorialware-ek* (meg kell említened, hogy a kódot a fejlesztő halott barátjának vagy rokonának ajánlod), *postcardware-ek* (egy képeslapot kell a szerzőnek küldened), *charityware-ek* (meg kell becsülnöd a szoftver értékét és jótékony célra kell azt felajánlanod) és mégkikelltalálniware-ek, amikor is te írsz egy kódot, kiadod, és kitalálod a saját licencfeltételeidet. A megszokott normál licenc a GNU General Public License (GPL), amely a www.gnu.org/copyleft/gpl.html oldalon található.

Hibakeresés

?:rdítás során a fordítóprogram számtalan egyszerű hibát észrevesz. A lefordított progra-— ok általában tiszták, és könnyű bennük a hibákat megtalálni. Az értelmezett szkriptekre ;_ó átállás azt eredményezte, hogy bizonyos, a számítógép által észlelhető hibákat addig -em vesszük észre, amíg a szkript adott részét nem értelmezi a program, ami akár hóna-r-: *skA* a szkript forgalomba kerülése után is megtörténhet. Az értelmezett szkripteket ne-~éz tesztelni. Egész iparág telepedett a szoftverfejlesztők köré, akik szisztematikusan tesztelik az értelmezett nyelveket, ennek ellenére a szisztematikusan tesztelt szkriptek is meg-bukhatnak. A PHP3 is szenvedett ettől a problémától.

A „memóriába fordító" szkriptértelmezőkre való átállás egy lépés hátrafelé, az egész szöveg egyszerre történő szintaktikai ellenőrzése felé. A PHP4 használat előtt a teljes szkript szin-takukáját ellenőrzi, lefordítja az összes függvényt, és alaposan átnézi a szkript fő részeit. Ha a PHP4-ednek sikerül működő HTML-t produkálni, az azt jelenti, hogy a szkriptben nincsenek szintaktikai hibák és az összes függvény megfelelően került lefordításra, s ez ? okkal jobb ellenőrzést jelent, mint ami PHP3-ban lehetséges volt.

A PHP előnyei

Egyértelmű, hogy a PHP az *egyetlen igazi nyelv*, de vannak, akiket csak a tények győznek meg, a tények pedig azt mutatják, hogy napjaink weboldalaihoz a PHP a megfelelő szkript nyelv. Tapasztalataim alapján állítom, hogy a PHP-t könnyebb tanítani, mint az olyan keverék nyelveket, mint a Visual Basic vagy az US. Mivel több iskolában tanítanak Visual Basicet, mint PHP-t, rövid távon az informatikai menedzserek számára vonzóbb a Visual Basic.

Értelmezés kontra fordítás

Gondolj a számítógépes nyelvekre úgy, mint utasítások listájára. Aztán vegyél egy általad rendszeresen használt utasítást, legyen az ételrecept vagy feljegyzés arra vonatkozóan, hogy hogyan találod meg a barátod új házát. Képzeld el, hogy az utasításokat begépelő ember néhány, számára magától értetődő utasítást kihagy, vagy néhány szót hibásan gépel be. Az eredmény az lesz, hogy nem tudod az utasításokat követni.

Vonatkoztasd ezt a problémát a fordítóprogramokra. Ahogy a programozó befejezi a gépelést, a számítógép megnézi a helyesírást, és ellenőrzi, hogy az egyes utasítások ott kezdődnek-e, ahol az *előző* befejeződött. (Azonban a számítógép nem tudja ellenőrizni, hogy borsra vagy borsóra gondoltál, hiszen mindkettő szükséges hozzávaló.)

A fordított nyelveknek vannak hátrányai: az egész programot meg kell írnod fordítás előtt, hiszen minden hiányzó rész fordítási hibát okoz. Az értelmezett szkriptekkel lehet soronként haladni, mert az értelmező anélkül foglalkozik az adott sorral, hogy ellenőrizné a többi megvan-e.

Vonatkoztasd a számítógépes értelmezést a hibásan írt névre. Amikor a programozó befejezi az írást, a számítógép semmit nem tesz. A számítógép addig nem ellenőrzi az utasításokat, amíg el nem kezded használni azokat. A számítógép addig nem foglalkozik az adott

név helyesírásával, amíg a nevet tartalmazó sor utasításához nem ér. Egy félig kész leves receptjének a közepére érve a gép közli veled, hogy nincs „őrült bors". Az egész folyamatban lévő munkát kidobhatod, és másnap kezdheted elölről.

A PHP a fordítás és értelmezés majdnem tökéletes keverékét nyújtja. Annyit ellenőriz, amennyit egy jó fordítóprogram ellenőrizne, és emellett az értelmezés előnyeit is biztosítja.

Részkódok kontra programozás

Fájlokból, függvényekből, objektumokból és minden egyéb töredékkódokból is összerakhatsz alkalmazásokat, de ez azért nem olyan, mintha a saját kódodat írnád meg. Ha sosem írtál kódot PostgreSQL adatbázishoz való hozzáféréshez, nem fogsz tudni egy másvalaki által írt PostgreSQL alapú kódot ellenőrizni. Rengeteg ingyenes PHP-objektumot és -alkalmazást Ietölthetsz. Ezek azonban akár olyan veszedelmes kódolási csapdákat is rejthetnek, amelyek nehézkessé tehetik a működtetést, és programod üzleti célokra alkalmatlanná válhat.

Léteznek olyan kódok, amelyek egyáltalán nem használják a PHP4 fontos funkcióit, például a = = = operátort, más kódok nem ellenőrzik a bevitt adatok érvényességét, megint mások pedig anélkül használnak adatbázist, hogy az egyes mezőkhöz a megfelelő adattípusokat alkalmaznák. Például vegyük a MySQL-t: MySQL-ben sokan használják táblázatok létrehozására a phpMyAdmint és soha sem veszik észre, hogy a phpMyAdmin alapbeállításként NOT NULL értékre állítja a mezőket. Ezek az emberek közreadják kódjaikat a weben, amiket mások hagyományos SQL-táblázatok létrehozására használnak fel, ez MySQL-ben alapértelmezésben NULL értéket ad a mezőknek. Ha egy ilyen csapdába beleesel, az SQL output oldalain a nulla értékek különös eredményeket okoznak.

A saját kódok írásakor eleget tanulhatsz ahhoz, hogy teszteld, ellenőrizd és kijavítsd a letöltött kódokat. Ha továbbra is használod a letöltött kódot, tudasd változtatásaidat a szerzőkkel (vagy ajánld nekik ezt a könyvet), hogy kijavíthassák a kódjukat, és a következő 25 000 letöltő mind jobban járjon.

Kimenet-ellenőrzés

HTML-oldalakat kétféleképpen is programozhatsz PHP-vel. íme az egyik módja annak, hogy PHP-vel hogyan szúrhatod be a dátumot a szöveg egyik sorába:

```
<p>Today's date is <?php print(date("1 F j, Y")); ?>.</p>
```

Ez a módszer működik, és feltételezhetően gyorsan megvalósítható, hiszen csak egy kis PHP-t szúrsz egy létező HTML-oldalba. Az ilyen megközelítésben működő PHP alapú alkalmazások szerte az egész webről letölthetők. Mindazonáltal jellemzően sok apró változtatást igényelnek ahhoz, hogy honlapodhoz illeszkedjenek, és az adatbázisokhoz csak minimális hasznukat veheted.

A következő példa viszont - a PHP-kódba beszúrt kis szöveggel - lehetővé teszi, hogy mindent kézben tarts, és jobban használható a nagy, adatbázis alapú alkalmazásokra is:

```
<?php
print ("<p>Today's date is " . dateC'l F j, Y") . ".</p>");
```

Ahogy egyre kifinomultabb alkalmazásokat fejlesztesz, a kódok aránya növekedni fog, és a szknptekből a szövegek az adatbázisokba vándorolnak. A második típusú programozás az evolúció végpontja, így akár minden új honlapot ettől a végponttól kezdhetsz. Ebben a könyvben minden kód a második példa alapján működik, amellyel olyan hatékony alkalmazásokat fejleszthetsz ki, amelyek minden böngészővel működnek.

PHP4

A PHP4 divatba hozza a sebességet: a hosszú szkriptek végrehajtásának és a PHP4 függvények fejlesztésének sebességét. A nyílt forráskód egyik előnye éppen a hiányzó rész hozzáadásának lehetősége, és azon programozók, akik szoktak függvényeket hozzáadni, úgy tartják, hogy sokkal könnyebb bármit is hozzáadni a PHP4-hez, mint a PHP3-hoz. Itt vannak a PHP4 újdonságai, amelyek azonnali előnyökhöz juttatják kedves olvasóimat.

Új függvények

A PHP4 új függvények berobbanását váltotta ki. Sok közülük a PHP és a nagy alkalmazások kezelését kívánja megkönnyíteni. Mivel néhány internetszolgáltató korlátozza a php.ini konfigurációs állományhoz való hozzáférésedet, a PHP4 számos olyan függvényt tartalmaz, amellyel tesztelheted és beállíthatod a PHP-konfigurációs értékeket.

A PHP3 engedi a dinamikusan generált függvényekkel való kontárkodást, a PHP4 viszont a dinamikusan generált függvények kezeléséhez szükséges függvények széles választékát nyújtja. A PHP4-ben annyira egyszerű és népszerű a függvényfejlesztés, hogy a PHP-függ-vények egyetlen naprakész listája csak a **php.net** oldalon található meg — bár ez a dokumentáció is küzd azzal, hogy naprakész maradjon. Egyszerűen nincsen elég példa arra, hogy a valós életben mire használhatók a PHP-függvények. **PHPtect.com** (a "PHP and Web site architecture" rövidítése) címen gondozok a programozóknak egy kicsiny listát. Azt gondoltam, a könyv írásához használni fogom a példákat, de minden alkalommal, amikor PHP-t használok, új függvényekkel vagy már létező függvények új paramétereivel találkozom. Napról napra szembetalálom magam az adott függvények jobb felhasználásával és az érdekességek növekvő listájával, így a könyvben található kódokat naponta frissítem a gépemen. A PHP-vel lépést tartani olyan, mint a gyereket ruházni. Mire a gyerekszobából elsétálsz a fogasig és vissza, nőnek egy méretnyit.

Új név

Korábban az emberek a HTML-oldalaknak a .html, a PHP3-at tartalmazó oldalaknak pedig a .php3 kiterjesztést adták. A PHP4 fejlesztők úgy állították be a PHP4-et, hogy az a .php kiterjesztést használja. A két fájltípus, a PHP3 és PHP4 lehetővé teszi, hogy egyes oldalakat PHP3-mal, másikakat pedig PHP4-gyel dolgozz fel. Általában gyorsabbnak találom a kis website-okat egyenesen PHP3-ról PHP4-re konvertálni, mint a kettőt párhuzamosan használni, ezért én ritkán használom a kettős elnevezéses rendszert. A kettős elnevezéses rendszer inkább a külső szervezetek által írt összetett honlapokat és alkalmazásokat kiszolgáló szerverek számára hasznos.

Ahogy növekszik a honlapod, úgy fogsz egyre több HTML-oldalt PHP-re konvertálni. Miért ne kezdenéd tehát azzal, hogy minden oldaladnak .html kiterjesztést adsz és PHP-n keresztül futtatod őket? Kis többletráfordítással a webszervered beállítható úgy, hogy minden oldal PHP-n keresztül fusson, és ezt követően az összes fejrészt és navigációt elkezdheted PHP-kóddal egységesíteni.

Sebesség

A PHP4 sokkal gyorsabb, mint a PHP3, mert a kódot egyszer fordítja. A ciklusokban a második és az utána következő alkalmakkor a kódod időt takarít meg. Minél nagyobb a programod, a PHP3-hoz viszonyítva annál több időt takarítasz meg.

A PHP4 annyira gyors, hogy a honlapod gyorsaságát korlátozó tényező bármi lehet, csak a PHP nem. Több előnyöd származik az adatbázisod indexeléséből, mint a PHP-kód gyorsítási lehetőségeinek mélyreható elemzéséből.

Rengeteg olyan kódgyorsító ötlet, amely érvényes volt a PHP3-ra, nem érvényes a PHP4-re. Például van aki a sztringek körül a egyszeres idézőjelek használatát javasolja, feltéve, hogy nem kell a kétszeres idézőjelben megengedett helyettesítést elvégezned. Kipróbáltam PHP4-ben az egyszeres és kétszeres idézőjelek használatát, és azt találtam, hogy az egyszeres idézőjelek kismértékben lassabbak a kétszeresnél, azonban a különbség elhanyagolható. A következetesség érdekében PHP4-ben használj kétszeres idézőjelet!

Az egyetlen észrevehető lassulást csak a többszörösen egymásba ágyazott függvények nagy ciklusokkal egyszerre történő alkalmazásakor tapasztaltam. Ha olyan programod van, amely 100,000 elemű tömbön fut keresztül, és ciklusból hív meg egy függvényt, akkor némi gyorsulást érhetsz el, ha a függvényből közvetlenül a ciklusba helyezed át a feldolgozó kódot. Az időmegtakarítás még így is jelentéktelen ahhoz képest, amit azzal érhetsz el, ha egy 100,000 elemű tömböt használsz, és nem kell magad újra és újra átrágni különféle lemezen tárolt fájlokon vagy adatbázisokon.

PHP-hibakereső

A PHP-hibakereső terén izgalmas fejlesztések történnek, de az írás idején egyik jelentős alkalmazás sincsen fejlesztésre kész állapotban. Egy tényleg jó szintaktika ellenőrző szerkesztő és egy az alkalmazáshoz készített fejlesztő terv használatát javaslom.

Windows NT-n a Homesite-ot használom szerkesztésre. Számos hasonló termék létezik mind Windowsra, mind Unixra, a választás attól függ, hogy pontosan milyen nyelveket használsz. Némelyik szintaktikailag ellenőrzi a HTML-t vagy a PHP-t, de soha nem egyszerre.

Ha a szintaktikát ellenőrzöm, az Apache, a PHP és a MySQL lokális másolatait használom. PostgreSQL és más adatbázisok esetén a kódot először lokálisan próbálom ki, és csak a megfelelő adatbázist töltöm fel a szerverre. Ha egyszer a célszerveren fut, ott folytatom, mert az adatbázisokkal túl sok egyedi eset történik ahhoz, hogy oda-vissza ugrálgassak.

AWindowsos COM

A függvények egy nagyobb részének a használatát, így a PHP-szkriptek Microsoft Word-beli megnyitását is a COM (Component Object Model) teszi lehetővé. A COM-mal számtalan egyéb dolgot megtehetsz, de mivel a COM csak Windowsos környezetben használható, annak bármilyen használata azonnal leszűkíti az alkalmazásod használhatóságát.

A PHP COM-támogatás jó néhány függvényt biztosít, amelyekkel hozzáférhetsz a COM-objektumokhoz: **com_get(), com_invoke(), com_Ioad(), com_propget(), com_propput(), com_propset()** és **com_set().** A 7. fejezetben bemutatom a COM egy példáját. Egészen addig nem tudtam jó indokot arra, hogy miért használjak egy csak Windows alatt futó állományt, amíg eszembe nem jutott, hogy a COM-ot használhatod arra, hogy megnyiss egy Word-dokumentumot, majd HTML- vagy RTF-formátumban elmentve azt bárki, bármilyen operációs rendszerben elolvashatja.

Reguláris kifejezések függvényei

A PHP a reguláris kifejezések két fajtáját támogatja, a POSIX- és a Peri-kompatíbiliseket. A reguláris kifejezésekre jelen fejezet „Gyors Megoldások" részében a „Reguláris kifejezések" pontban találsz néhány példát. A **POSIX** reguláris kifejezésekről több információt és tájékoztatót kaphatsz, ha felkeresed a **www.alltheweb.com** oldalt, és a „POSIX 1003.2 Regular Expressions" kifejezésre pontosan rákeresel. A POSIX kibővített függvényei közé tartozik többek között a **ereg(), ereg_replace(), eregi(), eregi_replace(), split(),** és **spliti().**

A Periben programozók sokkal jobban járnak a Peri-kompatíbilis reguláris kifejezésekkel, amelyek közé a **preg_match{), preg_match_all(), preg_replace(), preg_replace_callback(), preg_split(), preg_quote()** és **preg_grep()** tartoznak.

PHP és Apache

Melyik webszervert szeressem? Az Apache-t vagy az Apache-t? Őszintén **be kell** vallanom az Apache iránt érzett vonzalmamat. Annyira könnyű telepíteni, annyira könnyű használni és annyira megbízható, hogy először az Apache-t ajánlom, és utána vitassuk meg a hozzávaló operációs rendszer kérdését.

Tény, az Apache 1 nem volt tökéletes - ezért fejlesztették ki az Apache 2-t -, de a PHP-hoz és az első honlapodhoz az Apache a megfelelő választás. Az Apache 2 architektúrája lehetővé teszi az Apache fejlesztőinek, hogy azon ritka esetekben, amikor az nem működik megfelelően, tökéletesítsék. Még ha egy olyan ritka operációs rendszert is használsz, mint az AIX, az Apache 2 akkor is könnyedén illeszthető.

A platformfüggetlen Apache

Néhány évvel ezelőtt a Sun Solarist ajánlották legtöbben gyors, megbízható webszervernek. A FreeBSD is népszerű lett, míg az árra érzékenyek a Debian Linuxot választották. Aztán jöttek a FrontPage-dzsel kísérletező százezrek, és sok internetszolgáltató a Windows NT-t, az egyetlen olyan egyszerű megoldást választotta, amelyik megbirkózik a FrontPage-dzsel. Végül a Red Hat Linux is elterjedt.

Az Apache gyakorlatilag minden szerveren fut. Ha nem tudod, melyik operációs rendszert fogod használni, akkor válaszd az Apache-t, és utána gond nélkül válthatsz az operációs rendszerek között. Ha a webszerveredet más állítja fel, ő is az Apache-t válassza (és legyél bizalmatlan mindenkivel, aki nem azt választaná). Ha új számítógépre vagy operációs rendszerre váltasz, csak a könyvtárak nevét kell a konfigurációs fájlban, a httpd.conf-ban megváltoztatnod.

Az Apache a Windows NT-re való telepítése gyors és megbízható: az apache.org-ról vagy egy helyi tükröldalról, vagy e könyv CD-ROM mellékletéről töltsd le az Apache Win32 bináris változatát, majd a telepítéshez kattints az apache_1.3.20-win32-no_src-r2.msi fájlra. (Az 1.3.20 szám változatonként eltérő lehet.) Ha nincs a gépeden telepítve az MSI, töltsd le az előbbi helyről a InstMsi.exe-t, telepítsd, és utána telepítheted az Apache-t. A Windows (a Windows 2000-től) tartalmazza az MSI-t, de az NT 4-ben, illetve a korábbi Win-dows-verziókban nincsen benne.

Ha a Linux Mandrake disztribúcióját telepíted, csak válaszd a webszerver menüt, és az Apache már működik is. A Linux minden disztribúciójának más menülistája van, de a legtöbb telepíteni fogja az Apache-ot, ha olyasvalamire kattintasz, ami tartalmazza a „Web server" kifejezést. Legegyszerűbb a dolog a Mandrake disztribúcióval **(www.linux-mandrake.corn).**

Ahova Apache telepíthető, oda PHP is. A PHP telepítése platformonként változik, de az Apache-ban a PHP-re vonatkozó részek ugyanazok. A legtöbb Linux-disztribúció alapértelmezettként olyan nyelvekkel telepíti az Apache-t, mint a Perl, és a PHP-t választható lehetőségként hagyja. Emiatt vagy mindent telepítened kell, vagy be kell ásnod magad a választható összetevők menüibe a PHP telepítéséhez.

Általános szabályként fogadd el, hogy ha frissítéshez letöltőd akár az Apache, akár a PHP újabb verzióját, akkor töltsd le a másik program legfrissebb verzióját is, frissítsd az Apache-ot és röviddel utána a PHP-t. A kereskedelmi oldalakra az újabb változatokat csak körülbelül 30 nappal a php.net-en való megjelenésük után teszem fel, így azon ritka esetekben, hogy ha hiba van is az új PHP-változatban, ennyi idő alatt megjelenik a kijavított változat is.

Modul vagy CGI

Az Apache a PHP-t Common Gateway Interface-ként (CGI) vagy modulként futtatja. A pontos részletekkel az Apache és a PHP mellé adott telepítési útmutató foglalkozik. Ráadásul mindkettő helyett használható az Internet Server Application Programming Interface-nek (ISAPI) nevezett valami is.

A CGI ideális a PHP és az Apache tesztelésére. Az Apache és a PHP nem tesz semmit, amíg le nem futtatod a szkriptet, utána pedig minden az ideiglenes tárterületen történik. CGI esetén a PHP nem olvassa be a php.ini fájlt, amíg az Apache át nem adja a szkriptedet a PHP-nek. A CGI egyetlen hátránya a sebesség, hiszen az Apache-nak és a PHP-nek minden egyes alkalommal meg kell ismételnie a beállítást, amikor szkriptet futtatsz.

Ha a PHP-t az Apache moduljaként futtatod, bizonyos feladatokat az Apache és a PHP egyszer végez el, amikor az Apache elindul, így minden szkript futtatása egy kicsit gyorsab-

:m történik. A hatékonyság érdekében *győződj* meg róla, hogy minden kereskedelmi " rbszervereden a PHP modulként fut.

Ha a PHP-t az Apache moduljaként futtatod, kapsz ugyan néhány újabb függvényt, de ezeket valószínűleg nem fogod az első szkriptjeidben használni. Mire gyakorlatot szerzel a PHP-vel, vagy befejezed ezt a könyvet, tökéletes php.ini beállításod lesz. Tovább már nem •;ell a php.ini változtatásával kísérletezgetned, és készen állsz arra, hogy a PHP-t CGI-ként _ttató teszt Apache-szervereden a PHP-t modulként futtasd.

A PHP Zenje

A PHP-ben bármilyen stílusban programozhatsz, de vannak dolgok, amelyek jobban működnek, mint mások - ebben a részben ezeket próbálom meg felfedni. A legtöbb prob-!éma onnan ered, hogy a PHP nem rendelkezik szigorú adattipizálással. A szigorú adattipi-:ilás hátráltatja a 100 soros szkripteket író kezdőket, de fontos biztonsági tényező az 1000 soros szkripteket író profik *számára. A* PHP4-ben megjelenő fontos változások közül jó néhány az adattípusok használatát és az típusellenőrzést érinti.

Szintaktika

Az itt látható kód első sorának begépelésével PHP-t szúrhatsz be a HTM L-f áj lókba. A második sor egy egysoros PHP-szkript, amely nyomtatási utasítást tartalmaz, a harmadiktól a hatodik sorig pedig egy többsoros PHP-szkriptet mutatunk:

```
<?php   ?>

<?php print("Hello"); ?>

<?php
Smessage = "Hello";
print($message);
```

A PHP-kód <?php-vel kezdődik. A HTML-oldal bármely részén elkezdheted, és a kód XML kompatíbilis, így mindig a <?php szintaktikai használd. Használhatsz <?-vel kezdődő dolgokat is, de XML-lel az nem fog működni. A PHP kód mindig ?>-vel végződik.

Ha PHP-fájlon belül PHP-fájlt ágyazol be, a PHP a beágyazott fájlt HTML-ként kezdi olvasni, ezért ne felejtsd el a beágyazott fájlt <?php-vel kezdeni, hogy egyenesen PHP-ba jusson.

Minden utasítás pontosvesszővel végződik,;, és az utasításblokkot kapcsos zárójelek foglalják keretbe { }, mint itt:

```
if("a"   ==  "a") {
    print("true");
```

Bizonyos esetekben kihagyhatod a pontosvesszőt és a kapcsos zárójelet, de ha később módosítod a kódot, ez megnehezítheti a kód megértését, és hibákhoz vezet.

A sor végére // jellel megjegyzést szúrhatsz be, mint itt:

```
print("true") ;    // This is a comment.
```

/* és */ közé megjegyzésblokkot tehetsz, mint itt:

```
/* This is a comment that goes all over
this line and another line and in fact uses up
a whole three lines.    */
```

Változókat úgy hozhatsz létre, hogy $ jelet teszel egy név elé. A változók nevében a kis- és nagybetűk különbséget jelentenek. Ha a következő két sort szúrod a kódodba, akkor a $a-t kinyomtatva a $a-hoz rendelt érték jelenik meg, nem pedig a $A értéke:

```
$a = "the contents of a variable";
$A = "contents of a different variable";
```

Numerikus változók esetén nem kell idézőjelet használni. Ha mégis használsz, a változó sztringként tárolódik, de ha számként van rá szükség, akkor számmá konvertálódik.

```
$a = 33;
```

Nevekkel logikai értékeket is definiálhatsz, mint a következő példában:

```
$a = null;
$b = true;
$c =
falsé;
```

A függvényeket a 10. fejezetben mutatjuk be. A függvények neveiben nem okoz különbséget a kis- vagy nagybetűs írásmód. Ha definiálod a hero()-t, és azután megpróbálod a Hero()-t is definiálni, szintaktikai hibát követsz el.

Ha, akkor, különben

Bizonyára tudod, hogy a bináris mezőknek csak két értékük lehet: 0 vagy 1, igaz vagy hamis, igen vagy nem. Régebben, az Assembler-programokban a bináris mezőket igaz vagy hamis értékként dolgoztam fel, hiszen a kód és az adat teljes mértékben ellenőrzésem alatt állt, változás esetén pedig a fordítóprogram figyelmeztetett. Amikor magasabb szintű nyelvekkel kezdtem foglalkozni, furcsa dolgokat tapasztaltam. Bináris mezőt használtam, és utána azt vettem észre, hogy több mint két értéke van. Egy mezőt binárisan definiáltam, majd a következő programozó megváltoztatta a mező típusát, a fordító vagy az értelmező pedig nem reklamált. Ekkor kezdtem mindenhez else-t adni.

PHP-ben kevesebb adattípus van, és a PHP automatikusan konvertál a típusok között, ezért az adattesztelés fontosabbá vált. Gondold végig nyelvtől függetlenül a következő kódrészletet:

```
if($a == 0)    {...}
if($a == 1)    {...}
```

Szigorú tipizálás és bináris mező mellett a kód tökéletesen működik, mert $a csak 0-t vagy 1-et tartalmazhat, és a kód mindkét esettel foglalkozik. A következő kód szintén működik egy ilyen nyelv esetén:

```
if{$a      0)}
==
else   {..
.
```

Mi történik, ha jön egy másik programozó, és egészként definiálja a változót? Az if **else-t** használó próba valószínűleg úgy működik, ahogyan tervezték, hiszen minden I-nél nagyobb érték esetén ugyanaz lesz az eredmény, mintha 1 lenne. A két if utasítást tartalmazó kód nem fog működni, hiszen minden I-nél nagyobb értéket figyelmen kívül hagy. Mi történik, ha a következő programozó negatív számot használ? A -1-et ugyanúgy értékelje, mint a + 1-et, vagy másként? Ha nincsen logikai **else,** használj figyelmeztető üzenetet, mint itt:

```
if($a   = = 0 )     {...}
elseif($a   = = 1 )     {...}
else

    print("Warning     message...");
```

A PHP liberálisabb az adattípusok terén, és a PHP4 bevezette a = = = összehasonlítást, amellyel típusonként ellenőrizheted az értékeket, de ezzel sem lehet az adattípusokat rögzíteni, ezért az adatok értelmezésénél nagyon figyelned kell. íme egy kísérlet egy egyszerű binárisnak feltételezett mező összes lehetséges kimenetelének definiálására:

```
if (!isset($a)     {...somé    sort    of    warning...}
elseif ($a    ===    falsé)     {}
elseif($a    ===    true)     {}
elseif($a    = = 0 )     {...}
elseif (Sa    <    0)     {...}
else   {...}
```

Először is ellenőrizned kell, hogy a mező létezik-e, mert a PHP a hiányzó mezőket ugyanolyan nevű változóval helyettesíti. Amit a „PHP konfiguráció" rész „Gyors megoldások" pontjában javaslok, az az, hogy minden figyelmeztetést állíts be, így minden olyan kísérletről értesülsz, amely nem létező mező beolvasására irányuló próbálkozás.

A következő annak ellenőrzése, hogy a mező igaz vagy hamis. A == jelet = = = jelre kell cserélned, mert a = = engedi a PHP-nek, hogy numerikus vagy sztring mezőket igazra vagy hamisra konvertáljon, viszont a = = = először ellenőrzi, hogy megfelelő típusú-e a mező.

A PHP a 0-t hamisnak, minden mást igaznak tekint, míg más nyelvek a negatív számokat hamisnak tekintik. Amikor más rendszerből vagy a PHP-tól különböző más programmal létrehozott fájlból kapsz adatot, ellenőrizned kell a negatív számokat, illetve azt, hogy az azokat létrehozó szoftver miként kezelte a negatív számokat.

A megfelelő adattípus

Amikor először kapsz adatot valamilyen forrásból, a szkriptedet olyan kóddal akarod kezdeni, amely ellenőrzi és megjeleníti az első rekord összes mezejének értékeit, és utána megjeleníti az első rekordtól eltérő rekordokat.

Az adatbázisok olyan alkalmazások, amelyek megbolygatják az adataidat, hiszen átkonvertálják őket belső tároló formátumukra, és az eltárolt formátumukat jelenítik meg. Adataidat olyan formátumúvá kell konvertálnod, ami az SQL-nek és az adatbázisnak megfelel, hogy azután az adatbázisszoftver az SQL-ábrázolásból a legkisebb helyet elfoglaló bináris formátumúra fordítsa azokat, majd a PHP-nek legjobban megfelelő formátumban juttassa őket vissza. Leggyakrabban az olyan értékekkel fordulnak elő hibák, amelyeket könnyen lehet más értékre konvertálni, mint az igaz vagy hamis.

Jelen rész írásakor a MySQL-adatbázis nem rendelkezett igazi bináris típusú mezővel (a közeljövőre ígértek egyet), így a bináris értékek tárolásához a MySQL TINYINT-jéhez hasonló valamit kell használnod. Az üres értéket tartalmazó TINYINT megzavarhatja az SQL- vagy PHP-kódodat, így NOT NULL-ra állítva tudod őket kikapcsolni. A mező negatív értéket is tartalmazhat. A negatív értékeket a PHP igaznak tekinti, bizonyos más nyelvek viszont hamisnak, ezért állítsd a mezőt UNSIGNED-ra (előjel nélkülire). Csak így lehetsz a nyelvekre való tekintet nélkül biztonságban.

Szűkítsd a bejáratot, tágítsd a kijáratot

Minden nyelvben a legbiztonságosabb programozási stratégia a hibás adatok bevitelének korlátozása, ha pedig ezzel megvagy, akkor úgy programozz, hogy minden átjusson. Ha egy látogatód kitölt egy űrlapot, az üresen hagyott mező azt jelenti, hogy „Nem tudom" vagy azt, hogy „Nem érdekel"? A választ semminek, hamisnak, 0-nak vagy ""-nek fordítsuk? Az ellenőrző rutinnak segíteni kell a látogatót, hogy az pontosan meghatározhassa, hogy mire gondolt. A dokumentációdnak pontosan definiálni kell, hogy milyen érték kerüljön tárolásra, az űrlap és az adatbázis közötti kódnak pedig biztosítania kell, hogy az adatbázis csak a dokumentált értéket kapja meg, függetlenül attól, hogy az adatbázismezőbe mi illeszthető be.

Ha az adat már az adatbázisban van, rninden lehetséges értékre definiálnod kell a feldolgozási folyamatot, még azokra is, amelyeket a beviteli rutin nem engedélyez. Ha a programodnak postai irányítószámokat kell megjelenítenie, az üres és hamis értékeket konvertáld zéró hosszúságú sztringekké. Ha az output adatok HTML-táblázatban jelennek majd meg, a zéró hosszúságú sztringeket konvertáld -vé, mert néhány böngésző törli a táblázatok üres celláit. Az ismeretlen értékeket olyan értékké konvertáld, amelyek nem akadályozzák az output oldal működését, és szúrj be figyelmeztetést vagy hibaüzenetet.

Ne felejtsd el, hogy mások is dolgozhatnak a kódoddal, és lehet, hogy ők nem olyan aprólékosak vagy precízek, mint te, így lehet, hogy kitörlik a NOT NULL-t a mezőkből. Sokan közülük gyakorlat és tapasztalat nélkül dolgoznak website-okkal. Törekedj arra, hogy a kódod túlélje a hibáikat!

Gyors megoldások

Apache-konfiguráció

Ha PHP-hez konfigurálsz Apache-t, a következő kódot szúrd be az Apache konfigurációs fájljába, a httpd.confba. Ahol a Directorylndex-et látod, szúrd be, hogy **index.php** vagy **index** és bármilyen kiterjesztés, amivel a PHP-oldalak működnek. Ugyanezt tedd meg az AddType-pal: minden olyan oldal kiterjesztéshez, amelyet PHP-vel akarsz feldolgozni, adj hozzá egyet. A **LoadModule** sor az Apache-ot a **PHP4 ISAPI** változatához irányítja, és a könyvtár nevét arra a könyvtárnévre kell változtatnod, ahova a PHP-t telepítetted. Az utolsó sorban # jel van, ami megjegyzést jelöl. Megjegyzéssé alakítottam azt a sort, ami modul helyett CGI-ként futtatná a PHP-t, és lecserélné a **LoadModule** sort:

```
Directorylndex index.html index.php
LoadModule php4_module "c:/Program files/php/sapi/php4apache.dll"
AddType application/x-httpd-php .html
AddType application/x-httpd-php .lib
AddType application/x-httpd-php .php
AddType application/x-httpd-php .php3
AddType application/x-httpd-php . php4
#Action application/x-httpd-php "/cgi-bin/php.exe"
```

Egyedi Apache-függvények

Az itt példaként bemutatott függvények információt adnak az Apache-ról, a HTTP-ről vagy az Apche-ot körülvevő környezetről. Az ezen függvények által adott elemek leírását egy az Apache-ról vagy a HTTP-ről szóló jó könyvben találod meg. Ajánlom a Greg Holden-Nick Wells: *Apache Server Commentary* (The Coriolis Group, Inc.) és Greg Holdén: *Apache Server for Windows Little Black Book* (The Coriolis Group, Inc.) című könyveket.

Tartsd az alábbi kódot mindig a kezed ügyében, hogy megjeleníthesd az e függvények által adott listákat:

```
function list_list($list) {
    Stext = ""; whilejlist($k, $v)
    = each($list))

    $text .= "<br>" . $k . ": " .
    $v;}return ($text);
```

apache_lookup_uri()

Az **apache_lookup_uri()** függvénnyel az Internet-erőforrásokról kaphatsz információt. A függvény csak akkor működik, ha a PHP modulként van telepítve, CGI-n keresztül futtatva viszont nem. Az **apache_lookup_uri()** eredményként egy objektumot ad, ezért a függvény kipróbálására használd ezt a kódot, és nézd meg az eredményt:

```
print(list_list(apache_lookup_uri($PHP_SELF)));
```

Ez az eredmény:

```
Status:    200
the_request: GET /phpblackbook/apache_lookup_uri.html HTTP/1.0
method: GET
content type: application/x-httpd-php
uri: /phpblackbook/apache_lookup_uri.html
filename: i:/petermoulding/web/root/phpblackbook/apache_lookup uri.html
path_info:
no_cache: 0
no_local_copy: 1
allowed: 0
sent_bodyct: 0
bytes sent: 0
byterange: 0
clength: 0
unparsed uri: /phpblackbook/apache lookup uri.html
request_time: 987732253
```

apache_note()

Az Apache megjegyzésrendszere lehetővé teszi, hogy megjegyzéseket továbbíts az Apache-modulok között. Ha a saját Apache-modulodat úgy írod meg, hogy az - mielőtt a PHP látná az oldalakat — dolgozza fel azokat, akkor egy adatokat tartalmazó megjegyzést állíthatsz be a PHP részére. A PHP-n belül üzeneteket tartalmazó megjegyzéseket állíthatsz be a moduloknak, amelyek a PHP után dolgozzák fel az oldalt.

Ha olyan preprocesszort írnék, mint például az Active Server Pages- (ASP-) fordító, akkor üzeneteket küldetnék a preprocesszorral a PHP-szknptnek, egyszerű PHP változó-hozzá-rendeléssel. Nem tudom, mi értelme lehet az oldalakat a PHP után feldolgozni, hiszen PHP-vel mindent elvégezhetsz, ami egy webszkripthez kell, így soha nem lesz szükséged arra, hogy a PHP-bói megjegyzéseket küldj a többi modulnak.

Egy Apache megjegyzést az alábbi kóddal állítasz be:

```
apache note("visitor", "Péter Moulding")
```

Apache megjegyzést a következővel kaphatsz:

```
print(apache_note("visitor"));
```

ascii2ebcdic() és ebcdic2ascii()

Ha történetesen OS390-t használó IBM vagy BS2000-t használó Fujitsu Siemens nagygépen futó Apache-ra írsz szkriptet, használhatod az **ascii2ebcdic()** vagy **ebcdic2ascii()** függvényeket. Ezek ASCII-ról EBCDIC-re és vissza fordítanak, de csak OS390-en és BS2000-en működnek. Ha egy fájlt EBCDIC-ről kell ASCII-re fordítanod, az EBCDIC gépen csináld, és utána küldd a fájl az ASCII gépre.

getallheaders()

A **getallheaders()** függvény az összes olyan HTTP-fejlécet megjeleníti, amelyek a szkripteddel előállított oldalak kérésére érkezett. A függvény akkor működik, amikor a PHP-t modulként telepítettük. A függvény eredményül egy objektumot ad, ezért a függvény kipróbálására használd ezt a kódot, és nézd meg az eredményt:

```
print(list_list(getallheaders())) ;
```

Az eredmény így néz ki:

```
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Charset: iso-8859-l,*,utf-8
Accept-Encoding: gzip
Accept-Language: en
Connection: Keep-Alive
Hőst: test.petermoulding.com
Pragma: no-cache
Referer: http://test.petermoulding.com/phpblackbook/
User-Agent: Mozilla/4.76 [en]  (WinNT; U)
```

PHP-konfiguráció

A PHP-konfigurációs kérdések eltűnőben vannak: az alapértelmezett telepítés jól használható, és a telepítési paraméterek a szkripten belüli paraméterekkel is beállíthatók. Ez a rész - ahol csak lehet - a futás közbeni konfigurációra koncentrál, mert azt kell használnod, ha a rendszergazda vagy az internetszolgáltató megakadályozza, hogy frissítsd a php.ini konfigurációs állományt.

Miután a php,ini-t használva konfiguráltad a PHP-t, észreveszed, hogy hiányzik egy PHP-kiterjesztés, amelyet a PHP-szkripteden keresztül akarsz betölteni. Ha olyan internetszolgáltatót veszel igénybe, amely lassan hajtja végre a változtatásokat, magadnak kell az új kiterjesztést feltöltened. Esetleg egy olyan monstre kiterjesztésre van szükséged, amely rengeteg memóriát használ, viszont csak egy szkript erejéig van rá szükséged. Ezeket a problémákat a dl()-függvénnyel mind orvosolhatod.

A dl() függvény fájlnevet fogad el, és a fájlnévhez pontos elérési információ szükséges. Alapbeállításként a dl() függvény a php.ini-ben meghatározott extension_dir-ből tölt be. Windowsban a dl()-t használva a gnu_gettext betöltéséhez, a következőket írd be:

```
dl("gnu^gettext.dll");
```

Unixban a dl()-t használva a gnu_gettext betöltéséhez, a következőket írd be:

```
dl("gnu_gettext.50");
```

extension_loaded()

Mielőtt a kiterjesztés betöltéséhez a dl()-t használnád, az extension_loaded() függvénnyel ellenőrizheted, hogy az adott kiterjesztés be van-e már töltve. Mint az itt is látszik, az extension_loaded() a PHP belső modulnevét használja, általában a .dll vagy .so kiterjesztés nélküli fájlnevet. Az extension_loaded() függvény eredményként igazat ad, ha a kiterjesztés be van töltve, és hamisat, ha még nincs:

```
extension loaded("gnu_gettext");
```

Hibakeresés

A szkripteken belüli hibakeresés legjobb módszere, ha a szkripteket tesztekkel és üzenetekkel mindjárt az elején elárasztod. Ha egy tesztet raksz be, építs be egy üzenetet is, amely megmutatja, hogy melyik adatot ellenőrizted és mi lett az eredmény.

Valójában a teszteket akkor kellene megírnod, mikor a szkript kódolása előtt dokumentálod az adatokat. A vevőazonosító mindig egész szám, vagy pedig kevert neveket és számokat is engedélyezel? Mit írsz be az érdeklődő potenciális vásárlókra?

Ha egyszer már túl vagy a tervezési fázison és igazán fogós problémákkal találod magad szembe, az alábbi hibakeresőket veheted igénybe.

assert()

Az assertQ egyenértékű az if(eval()) függvénnyel, ahol az egyetlen művelet a hamis ered mény esetén megjelenő üzenet. Az assert() bármilyen sztringet elfogad az értékeléshez, így használhatod hibakeresésre. Ne használd viszont felhasználói input érvényesítésére, mert kevés lehetőséged van az eredmény ellnőrzésére. A következő utasítás hamis, így rögtön az utasítás után az alábbi üzenet jelenik meg: /

```
assert(25 == 92);
Warning: Assertion failed in /petermoulding/assert.html on line
5 (Figyelem: Az állitás hamis a /petermoulding/assert.html 5.
sorában)
```

Az assert() szigorúan hibakeresési célokra szolgál, és 50 elkészített honlap után mellesleg megjegyezhetem, hogy soha, semmire nem használtam az assert()-et. Az assert()-tel kapcsolatban a következő három szabályt tartsd be:

1. Ne használd az assert()-et.

2. Ha mégis használod az assert()-et, olvasd el a dokumentációt a php.net oldalon.
3. Kövesd az 1. szabályt.

assert_options()

Az assert_options() függvény némi ellenőrzési lehetőséget ad az assert() használata után kapott eredmény felett. Az itt látható első kódsor egy az assert()-hez kapcsolódó opció aktuális értékét eredményezi, a második sor azt mutatja meg, hogyan kell az opciónak új értéket, jelen esetben 1-et adni:

```
$setting = assert_options(ASSERT_ACTIVE);
$setting = assert_options(ASSERT_ACTIVE,
1);
```

Az 1.1 táblázat a php.ini-ben és az assert_options()-on keresztül beállított assert()-opció-kat mutatja.

1.1 táblázat assert() opciók

Php.ini opciói	Alapértelmezett érték	assert_options()	Használat
assert_active	1	ASSERT_ _ACTIVE	Bekapcsolja az assert()-et
assert_bail	0	ASSERT_ _BAIl_	Befejezés, ha a kiértékelés nem sikerül
assert_callback	üres	ASSERT_ _CALLBACK	Függvényt hív be, ha a kiértékelés nem sikerül
assert_quiet_eval	0	ASSERT_ _QUIET_EVAL	Kikapcsolja az error_reporting-ot kiértékelés alatt
assert_warning	1	ASSERT_ ,WARNING	Figyelmeztet a sikertelen kiértékelésről

die()

A die() a szkripted utolsó leheleteként egy üzenetet ad ki. Sokan szeretnek olyan szkriptet írni, amelyik rejtélyes üzenetet küld és megáll. Sose tedd ezt egy ügyféllel. Kereskedelmi honlapokon mindent gondosan próbálj ki, naplózd a problémákat, és hasznos üzeneteket küldj.

íme egy gyilkos kód, amely megöli a szkripted, és az üzenet, amit produkál:

```
die("This script is suffering acute bad
programming") ; This script is suffering acute bad
programming (Ez a szkrip rossz programozástól
szenved)
```

A fájlok hibamentesítésére a die() függvényt is tartalmazhat. Ha van egy clean_up_files() nevű hibamentesítő függvényed, akkor a következő kód lefuttatja, majd megöli a függvényt:

```
clie(clean up
```

19

Hibák

A php.ini-ben alapértelmezett beállítás:

```
error_reporting = E_ALL &
~E_NOTICE
```

Ezt azonnal állítsd át erre:

```
error__reporting = E_ALL
```

A két beállítás közötti legfőbb különbség az, hogy az alapértelmezett beállítás megengedi, hogy minden figyelmeztetés nélkül hiányzó változókat használj, míg a második ilyen esetben figyelmeztet. így elkerülheted a hiányzó változó — egyszerű gépelési hibák miatt történő - használatát. (A php.ini a beállítást leíró megjegyzéseket tartalmazza.)

Az **error_reporting()** használatával a php.ini-ben hatályon kívül helyezheted a hibajelentő beállításokat, de erre soha nem lesz szükséged. Ha az internetszolgáltatód nem engedi, hogy megváltoztasd a php.ini-t, kérd meg, hogy állítson be .htaccess fájlokat. Ha ezt sem teszi meg, keress másik szolgáltatót.

Ha meg akarod szüntetni a hibákat, a függvénynevek elé tegyél @ jelet, amellyel leállíthatod a hibát előállító függvényeket, majd kiiktathatod a hibákat. Ha az SQL-ben hiba van, a MySQL-függvények hibaüzenetet eredményeznek! Te viszont nem akarod, hogy a felhasználók lássák ezeket. Ezért @-val szüntesd meg a hibákat, és utána a MySQL által előállított hibamezó'kön végezz hibaellenőrzést. Az egyéb függvényekből származó hibákat a globális PHP-hibamezőben, a $php_errormsg-ben találod.

Naplózás

A PHP hibanaplót vezet, és a naplót a PHP saját naplójába, a rendszernaplóba vagy a php.ini **error_log** konfigurációs sor használatával egy fájlba irányíthatod. A következő utasítással saját üzenetedet is hozzáadhatod a hibanaplóhoz:

```
error logC'this is a message", 0);
```

A 0 azt jelenti, hogy az üzenet oda kerüljön, ahova az **error_log** beállítás mutat. Az üzenetet e-mailen keresztül is elküldheted, ha a 0-t l-re változtatod és a **mail()** függvényben használt fejlécekhez további fejléceket adsz hozzá. A **mail()** függvénnyel a 15. fejezetben foglalkozunk, így feltétlenül nézd át azt a fejezetet, mielőtt a hibaüzeneteket e-mailben próbálnád továbbítani.

Kimenet-ellenőrzés

Előfordulhat, hogy a HTML kimeneti adatokat anélkül akarod megírni, hogy a kimenetet elküldenéd a böngészőnek, majd úgy döntesz, hogy törlőd a kimenetet. A PHP kimeneti pufferrel kontrollálhatod, hogy a kimenet ténylegesen mikor kerüljön a böngészőbe. Ha a kimenetet még nem akarod a böngészőbe küldeni, írd ezt:

```
ob start ();
```

Ha már készen áll a kimenet arra, hogy a pufferbe küldd, írd ezt:

```
ob_end_flush{);
```

Ha úgy döntesz, hogy nem küldöd a kimenetet a pufferba, írd ezt:

```
ob_end_clean();
```

Ha el akarod olvasni vagy lemezre akarod menteni a puffereit kimenetet, írd ezt:

```
$x    =    ob_get_contents();
```

Néhány webszerver a PHP-ból puffereli a kimenetet, így azt PHP-ból nem tudod kontrollálni. Néhány böngésző puffereli a bemeneti adatokat, és azokat sem tudod PHP-ból kontrollálni. A Netscape legtöbb változata addig nem jeleníti meg a táblázatokat, amíg nem kapja meg a **</table>** végtag-et, így akármit is csinálsz, a hosszú táblázatok megjelenítése örökkévalóságnak tűnhet.

Biztonság

A PHP legfőbb biztonsági problémája akkor jelentkezik, ha egy látogató által bevitt adatot az **evalQ** függvénnyel dolgozunk fel. Felhasználói adattal soha ne futtasd az **eval()-t,** mert a felhasználó bármilyen neki tetsző PHP-kódot alkalmazhat, és ezzel tönkreteheti a rendsze red. Ha azt akarod, hogy a felhasználók pofás formátumellenőrzőt használva vihessenek be adatokat, állítsd a bemenetet XML-folyamra, és csak azokat a tag-eket dolgozd fel, amelye ket elfogadsz.

Reguláris kifejezések

A reguláris kifejezésekkel sztringen belül kereshetsz sztnngekre, sztringekkel helyettesítheted őket. A PHP számos függvénye sokféleképpen teszi lehetővé a reguláris kifejezések használatát.

ereg()

A következő kód azt írja ki, hogy *igaz*, mert az ereg() megtalálja *xftear-sztr'mget* a $x-en belül.

```
$x = "apples and pears are fruit"; if
(ereg("pear", $x)
    {
    print ("true");            ;
    }
```

Felhívom a figyelmed, hogy az **ereg()** a *Pear-t* nem találná a $x-ben, hiszen az **ereg()** megkülönbözteti a kis- és nagybetűket.

Az 1.2 táblázat olyan speciális karaktereket tartalmaz, amelyekkel az eregQ-keresés kiegészíthető.

1.2 táblázat eregQ **speciális karakterek**

		Szintaktika	Eredmény
	Jelzi, hogy a keresési sztringnek a keresett sztring elején kell lennie	^pear	pear
$	Jelzi, hogy a keresési sztringnek a keresett sztring végén kell lennie	pear$	pear
*	Olyan sztringet keres, melyben a keresési sztring után bármilyen karakter lehet	an*	ann vagy annn vagy annnnnnnnn
	Asztringben a kapcsos zárójelben meghatározott számú karaktert keres	an{3}	annn
	A sztringben a kapcsos zárójelben meghatározott számú karaktereket	an{3,4}	annn vagy annnn

Sok más szimbólumot is használhatsz a reguláris kifejezések eredményeinek ellenőrzéséhez (a **www.phpbuilder.com/columns/dariol9990616.php3** nagyon jó cikket tartalmaz a reguláris kifejezésekről), én most mégis továbblépek más PHP reguláris kifejezésekre.

ereg_replace()

Az **ereg_repkce()** függvénnyel az ereg()-et helyettesítheted. A következő kód a *pear-t orange-dzszí* helyettesíti:

```
$x = "apples and pears are
fruit"; ereg_replace ("pear", "orange",
$x);
```

eregi()

Az **eregí()** függvény az **ereg()** olyan változata, amely nem különbözteti meg a kis- és nagybetűket. Ha az **eregí()** keresi a *Pear-t* a $x-sztringben, akkor *a pear-t* találja meg, hi szen az **eregi()** figyelmen kívül hagyja a kis- és nagybetűk közötti különbséget.
Az eregi{)-t a következőképpen tartsd fejben: \

```
ereg (strtolower("pear"), strtolower($x));
```

eregi_replace()

Az **eregi_replace()** függvény az **ereg_replace()** olyan változata, amely nem különbözteti meg a kis- és nagybetűket.

split()

A split()-függvény egy sztringtömböt eredményez, amely a sztring határolójel mentén felosztott részeit tartalmazza. A következő kód a $x-et a $a tömbbe töri le, majd kinyomtatja a tömböt. Az eredmény *pears,apples,oranges:*

```
$x = "pears and apples and
oranges"; $a = split(" and ", $x);
while (üst ($k, $v) = each($a))
    {
    print($v . ",");
```

spliti()

A spliti()-függvény a **split()** olyan változata, amely nem különbözteti meg a kis- és nagybetűket, és akkor is működne, ha az előző példában az egyik *and* helyett *And-et* írtunk volna.

2. fejezet

Adatok

Áttekintés

Az adat a program alapvető alkotóeleme. Ha az adatokat hibásan adod meg, a program ugyan működni fog, de hibás eredményeket kapsz, ami rendkívül zavaró lehet. A PHP automatikus típuskonverzióval rendelkezik, ami fokozza a hibakeresés nehézségeit. Szerencsére a PHP4-ben számos további eszköz áll rendelkezésre az adatok egyszerű ellenőrzéséhez.

Ez a fejezet az adatdefiniálással, -létrehozással, -teszteléssel és -konvertálással foglalkozik. A fejezetben egyedi adatforrásokra és rendeltetési helyekre, például MySQL-adatbázisra hivatkozunk. A példák segítenek abban, hogy adatokat olvass be saját forrásokból, és adatokat írj saját adatbázisba. Az egyes adatbázisokkal részletesen az 5. és 6. fejezetben foglalkozunk.

Adattípusok

PHP-ben az adatokat definiálhatjuk sztnngként, egészként, lebegőpontos számként, logikai értékként és a típusok összetett keverékeként. Ezenkívül az összetett matematikai műveletekhez a PHP speciális matematikai függvények két halmazát biztosítja. íme egy egyszerű sztring:

```
$a = "any alphanumeric characters";
```

A „Változók" című részben meglátod, hogyan tudod az adattípusokat kontrollálni, amit a PHP automatikusan végez. A „Műveletek az adatokkal" című részben bemutatom, hogyan tudod az adatokat összehasonlítani és megváltoztatni. A tömböket részletesen a 3. fejezetben tárgyalom, de néhány egyszerű tömböt ebbe a fejezetbe is becsempésztem, hiszen tömbökkel könnyebb az adatgyűjtés. Valójában az előző kódban példáként bemutatott $a-sztring is tömb. A $a ötödik eleméhez a következő utasítással jutsz:

```
$fifth = $a[4];
```

Mint látni fogod, a PHP a sztringekben, a tömbökben és a függvényekben nullától kezd számolni, így a $a ötödik elemére [4]-ként hivatkozunk. Ha a PHP-adatokkal szokatlan eredményre jutsz, ellenőrizz minden indexet és hivatkozást, így például a kezdő karakterszámot a substrQ-függvényben, arra az esetre, ha 0 helyett 1-gyel kezdted a számolást.

Változók

A $a egy változó, hiszen értékét bármikor megváltoztathatod. A konstansok értékét és a változók értelmezési tartományát nem lehet megváltoztatni. Mindezekről a fejezet későbbi részében beszélek.

A változók meghatározott, például egész- vagy sztringtípusú változók lehetnek. Azonban a PHP szükség esetén automatikusan megváltoztatja a változók típusát, így ha feltételezéssel élsz egy változó típusát illetően, meglepő eredményeket kaphatsz. Egy zéró karaktert tartalmazó sztring (mint a következő példa első sora) sztring ugyan, de üressé, hamissá vagy nullává konvertálható.

A sztring, amely O-át tartalmaz (a péda második sorában) automatikusan átkonvertálható üressé vagy hamissá:

```
$s   =   --;
$t   =   "0";
```

Ha az adattípus nagy jelentőséggel bír, a == műveleti jelet a ===== jellel helyettesítsd, vagy használj olyan típustesztelő függvényt, mint az is_integer{), amelyet a következő részekben mutatok be.

A változók neve különbséget tesz a kis- és nagybetűk között, azaz a Speter nem azonos a $Peter-rel, és mindkettő különbözik a $PeTeR-től. A $Peter_The_Great változó nem azonos a $PeterTheGreat-tel, ráadásul az előbbit sokkal könnyebb elolvasni. Takarékoskodj az alulvonás (_) karakter használatával - bár hosszú változóneveket kreálhatsz, de belefáradsz a gépelésbe. Az alulvonás karakter működik MySQL-ben, de nem az összes adatbázisban. Éppen úgy, ahogy bizonyos operációs rendszerekben a fájlnevek tartalmazhatnak alulvo-nást, bizonyos operációs rendszerekben nem. Mivel nehéz megjegyezni, hogy hol használhatom és hol nem, külső nevek esetén inkább nem használom az alulvonást.

A változók hosszúsága maximált, de a sztringek majdnem korlátlanul hosszúak lehetnek (csak a memória korlátoz), és mivel bármi sztnnggé konvertálható, a sztring megjelenítés hosszúságával gazdálkodhatsz. Ha számokkal dolgozol és az adatbázisban tárolod őket, észre fogod venni, hogy a PHP-függvény által jelzett hosszúság nem segít annak kiszámítá-sában, hogy a szám a lemezen mennyi területet foglal majd el az adatbázisban.

Automatikus létrehozás

A hiányzó változókat a PHP automatikusan létrehozza. A következő példában a definíciós sorban $metre-ként begépelt változóra a print()-utasításban $meter-ként hivatkozom. A PHP alapértelmezetten létrehoz a print-utasításban egy üres változót $meter névvel. Ha a figyelmeztető üzeneteket bekapcsoltad, akkor a PHP figyelmeztet, amikor létrehozza a változót. Jómagam minden ellenőrzést a figyelmeztetéseket bekapcsolva végzek el, így ész-lelhetem és kijavíthatom az ilyen jellegű hibát:

```
$metre = 3 5;
print{$meter) ;
```

Az isset()-függvény ellenőrzi, hogy a változó létezik-e, az unset()-függvény pedig teljesen eltávolítja a változót. Az unset() olyan esetekben használható, amikor a kód az Ísset()-et használja annak ellenőrzésére, hogy a változó létezik-e. Körültekintően használd az unset()-függvényt a különleges értékek jelölésére, mert lehet, hogy más nem annyira kö-vetkezetesen használja az isset()-et, mint te. Ha a php.ini értékei nem megfelelően állítottad be, előfordulhat, hogy a hiányzó változók miatti figyelmeztető üzenetek elmaradnak, amelyek máskülönben figyelmeztetnék a programozókat az isset() használatára. A szükséges php.ini beállításokat az 1. fejezet tárgyalja.

Konstansok

A *konstanst* begépeléskor kell meghatározni: .'■■»-.

def ine ("a", "any alphanumeric charapfceí^s") ;

A define parancs az első paraméterként megadott névvel létrehozza a konstanst, és a második paraméterként adott értéket rendeli hozzá. A fenti példában az a konstansnak az *any alphanumeric characters* az értéke.

A define parancs bárhol alkalmazható, és ahogy ezt a következő részben bemutatom, a definiált konstansoknak globális a hatókörük, azaz mindenhol, így függvények belsejében is használhatóak, ahova a változók hatóköre egyébként nem ér el. A define parancs szimpla adatdefiníciókat fogad el, így olyan összetett adatstruktúrákra, mint az objektumok, nem használható. A define-t használhatod név tárolására, a nevet alkalmazhatod egy objektum létrehozásakor, majd pedig a definiált névvel közvetetten utalhatsz az objektumra. Egy így létrehozott alkalmazástól azonban biztosan felforr a következő programozó agyvize.

Hatókör

A *hatókör* határozza meg, hogy az adat hol használható. A függvényeken kívül definiált változók a függvényen belül nem láthatóak, a függvényeken belül definiáltak pedig a függvényeken kívül nem láthatóak. így a függvényt majdnem teljesen az azt körülvevő kódtól függetlenül írhatod meg. Az egyes adatok hatóköre eltérő és változtatható: a definiált konstansok átnyúlnak a függvények határain, és a hagyományos változók is benyúlhatnak egy függvénybe, ha a függvényben a változót globálisként definiáltuk. A hatókörről és a függvényekről további részleteket a 10. fejezetben olvashatsz, míg a hatókörök objektumokkal kapcsolatos részleteit a 17. fejezetben találod.

Mivel a PHP automatikusan létrehozza a hiányzó változókat, könnyen megzavarodhatsz az adatot tartalmazónak vélt, de valójában üres változók miatt. Ha a kódodban $whiskey néven hozol létre változót, majd egy függvényen belül $whisky-ként hivatkozol rá, a $whisky üres lesz. Ettől összezavarodhatsz, és miközben megpróbálod kijavítani alkalmazásodat, pánikba esel, így akár szükséged is lehet erre az italra.

Először is nyugtasd meg magad egy csésze forró kamillateával, majd az alábbiról

```
; error_reporting = E_ALL & ~E_N0TICE
```

változtasd a php.ini-t a következőre:

```
error_reporting = E_ALL
```

Az -~E_NOTICE eltávolításával a PHP figyelmeztet a hiányzó változódefiniálásra és más ördögi csapdákra.

Műveletek az adatokkal

A PHP számtalan kifejezést, operátort, ellenőrzést és szerkezetet, függvényt, osztályt és objektumot kínál, továbbá sokféle lehetőséget nyújt az adatok kezelésére mind az adatbázisok, mind pedig a HTML felé.

Kifejezések

A PHP-ban minden kifejezés. Gépeld be a koromat, a 29-et egy önmagában álló sorba, és egy 29 értékű kifejezést kapsz. Bővítsd ki a sort "$age = 29;"-re. Most már három 29 értékű kifejezésed van: az eredeti 29, a $age, amit 29-re állítottál be, és maga az egész utasítás. A következő sor a 29-et fogja kinyomtatni, hiszen a printQ-függvény a kifejezés értékét jeleníti meg, és a kifejezés értéke nem más, mint a kifejezés bal oldalához hozzárendelt érték:

```
print($age = 29);
```

PHP-ban lehetséges több egyenlőségjel használata is, melyeket az jobbról balra értékel. A következő példában a $age 29, így utána a $number is 29, majd a print()-utasítás eredményeként megjelenik a 29:

```
print($number = $age = 29);
```

A kifejezések értékét olyan ellenőrző függvényekben is használhatod, mint a következő példában bemutatott if(). Ez a példa triviális ugyan, ám ez a tulajdonság jól használható ellenőrzéskor. Az if() és while() ellenőrző függvényekkel a fájl- és adatbázisfüggvényeket, a grafikus függvényeket, illetve majdnem bármilyen függvényt körbecsomagolva a kapott adatok alapján könnyen ellenőrizhető az adatfeldolgozás folyamata:

```
if($age = 29) {
    print("Age is
    true");
```

A következő példa a mysql_fetch_row()-függvénnyel olvas be egy MySQL-adatbázisból SQL-lekérdezéssel generált adatsort. (Az adatbázisfüggvények részletes magyarázatát az 5. fejezetben találod.) A mysql_fetch_row() addig ad adatot, amíg a sorok el nem fogynak, majd a válasz hamis lesz. Amikor a $row hamissá válik, az egész kifejezés ($row = mysql_fetch_row($query_result)) értéke hamis lesz, így a while()-ba is hamis kerül. A while()-ciklus addig lép vissza a kifejezésre, amíg az hamissá nem válik, ha ez bekövetkezik, átugorja a program további részeit. A programodban nem csak egy, például a példában szereplő print-utasítás lenne a ciklusban, hanem a $row feldolgozására szolgáló kód is:

```
while($row = mysql_fetch_row($query_result))
    {
    print("Another row from the SQL query");
```

A PHP mindent megtesz annak érdekében, hogy a kifejezéseket használható módon értékelje. A 3. fejezetben látni fogod, hogy a következő while()-szerkezet hogyan lépked a tömbökön keresztül. Az each() a tömb egy elemét adja vissza, rálép a következő elemre, és hamis eredményt ad, ha elén a tömb végét. A Hst() az elemet kulcsra és értékre bontja, és igaz választ eredményez. Amíg az eachQ ellátja adattal a list()-et, a while() igaznak értékeli a kifejezést, és folytatja a ciklust. Amikor az each() hamis értéket továbbít a list()-nek, a listQ hamis értéket továbbít a while()-nak, és a while() leállítja a ciklust:

```
while (Üst ($key, $value) = each { $array) )
```

A PHP korai változatának függvényei érdekes dolgokat tettek, ha egy folyamat véget ért, de a PHP fügvények most már csaknem teljesen egységesen hamis eredményt adnak, ha az adat végéhez érnek, vagy ha hibát találnak. így a kifejezések különbséget tesznek a között, hogy a függvény működik és nem eredményez adatot, vagy hogy a függvény nem működik. A új PHP4 = = = operátor, melyet a következő részben mutatok be, döntő fontosságú a kifejezések és függvények kezelésében.

A kifejezés változók, értékek és operátorok keveréke, így az operátorok ismerete nélkül nem mehetünk tovább. A következő, operátorokról szóló részben mindenről szó esik, amit az ellenőrző utasítások használata nélkül meg lehet csinálni. Utána továbbmegyünk a „Ellenőrzés és szerkezet" résszel, amelyben mindent megtudsz a feldolgozási folyamat ellenőrzéséről. Ha ismered a Perit vagy a C-t, akkor eleget tudsz ahhoz, hogy az esetek többségében jól tippelj, a fennmaradó kisebb rész esetén pedig dühöngjél, hiszen a PHP a két nyelv legjobb részeit ötvözi az elavult trükkök és csapdák mellőzésével. Lehet, hogy a kedvenc trükkjeid nem működnek majd, de többségük - elsősorban azok, amelyeket az egyszerű halandók is megértenek - továbbra is használhatók.

Operátorok

Kezdj el ismerkedni a =, ==, = = = , ! = , ! = =, + = ,. = , +,-, *,/,$, & jelekkel, illetve a billentyűzet minden szokatlan karakterével. Ezek a PHP-operátorok. Ebben a részben olyan sorrendben mutatom be ezeket, ahogyan egymásra épülnek. Ahelyett, hogy az egyes utasítások eredményét külön sorban mutatnám meg, megjegyzésként a sor végére szúrom őket.

Hozzárendelési operátor: =

Kezdésként rendeljük az életkoromat egy változóhoz a hozzárendelési operátor, az egyenlőségjel = segítségével. Ez a baloldali változóhoz a jobb oldal értékét rendeli. A példánkban a $age változót 25-re állítjuk. Az eredmény, 25, a // jel után látható (a // jellel egy egysoros megjegyzés kezdődik, így a kód működni fog a szkriptedben):

```
$age = 25;          // 25
```

Ahogy a következő szövegben is látható, egy utasításban több hozzárendelési operátort is használhatsz. Ebben a példában minden születésnapomba az az érték kerül, amelyet az elmúlt kb. 10 évben használtam:

```
$age_2004 = <$age_2003 = <$age_2002 = ($age = 25)));      // 25
```

A következő sor az egyenlőségjel egy gyakori csapdáját mutatja, amikor ezt a jelet a két érték egyenlőségének vizsgálatára használják. Ebbe a csapdába leggyakrabban azok esnek bele, akik más nyelveken nőttek fel. A következő kód eredménye mindig igaz, hiszen az egyenlőségjel a **$width** értékét a **$height** értékével teszi egyenlővé. A **Sheight** általában pozitív szám, és az if() a pozitív számokat igaznak értelmezi. Ennek a problémának az egyik tünete a **$width** szokatlan értéke:

```
if ($width = $height);      // true
```

Összehasonlítási operátorok: == === /= /==

A $height és a Swidth változókat helyesen így kell összehasonlítani. Az én esetemben a $width nem egyenlő' a $height-tel:

```
if(Swidth —— $height);        //    falsé
```

Az összehasonlítási operátorok == két értéket hasonlítanak össze, és a PHP automatikus adattípus-konvertálása miatt hibához vezethetnek. Ha egy függvény eredménye rekordszám, és a hiba jelzésekor hamis eredményt ad, a = = kód használatával a zéró rekordszám hamissá konvertálódik, és a függvény hibát jelez, amikor nincs is hiba. A leggyakoribb példák erre az adatbázisfüggvények, ahol az SQL select utasítás működik és zéró rekordot választ ki, de szerencsétlen programozó napokig próbálja megfejteni, miért van hiba az SQL-utasításban, pedig csupán az if()-utasítás értelmezi a zérót hamisnak. Helyettesítsd a = = jelet = = = jellel, amely összehasonlítja az értéket, és megkülönbözteti a hamist a zérótól. A mysql_num_rows()-függvény a sorokat számolja meg, de hamis eredményt ad, ha a $result változónak rossz az értéke vagy rossz változó:

```
Jrows = mysql_num_rows($result) ;
if($rows --- falsé)  // true if MySQL error, falsé if row count
zero
```

A = = = fordítottja a ! = = , a == fordítottja pedig a ! = . A 2.1 táblázat az összehasonlító operátorok rövid összefoglalását tartalmazza.

2.1 táblázat Összehasonlító operátorok

Operátor Jelen			Hamis
□Egyenlő	Egyenlő ertek	8 ==8	8 ==4
!=	Nem egyenlő érték	8 !=4	8 !=8
===	Egyenlő érték és egyező típus	0===0	0 === falsé
!==	Nem egyenlő érték vagy nem egyező típus	0 !== falsé	0!==0
<	Kisebb	4<8	8<4
>	Nagyobb	7 >6	7>8
<=	Kisebb vagy egyenlő	4 <=4	8<=4
>=	Nagyobb vagy egyenlő	7>=6	7>=8

Néhány nyelvben a => ugyanúgy működik, mint a > = , de a PHP-ban nem. Nincs sem > = =, sem < = = operátor, de a !-t máshol is használhatod, nem csak a fenti példában. A ! használatáról a következő részben további példákat találsz.

Aritmetikai operátoroké - / %*

Tudnak a PHP-operátorok összeadni? Hogyne tudnának. A + összeadáshoz, a - kivonáshoz, a * szorzáshoz, a / pedig osztáshoz használható, a % pedig az osztás maradékát adja. Ezek az operátorok mind lebegőpontos számot adnak eredményül, ha az input lebegőpontos, és lebegőpontossá konvertálják az egészeket, ha az eredmény nem egész.

A matematikai műveleteket az operációs rendszer végzi, amely ehhez a számítógéped hardver részét használja, így ezen műveletek pontossága a hardvertől és attól függ, hogy az operációs rendszered hogyan kezeli a hardver eredményeit. A következő részekben leírt GMP (egy nyílt forráskódú matematikai csomag) matematikai függvényekkel korlátlan pontosságú egész-matematikai műveletek végezhetőek, ha a GMP telepítve van. Ha egy lebegőpontos számból egész számot szeretnél, a floorQ a lebegőpontos szám egész részét adja, a ceil() pedig felfelé kerekítve ad egészt eredményül.

Sztringoperátorok:. .=

Két sztringet a pont (.) használatával fűzhetsz össze, a pont és az egyenlőségjel kombinálásával pedig egy sztring végéhez csatolhatsz egy másikat: . = . A következő kétsoros kód első sora a $x változónak az "open source" értéket adja az "open" sztring, egy szóköz és egy "source" sztring összefűzésével. A második sor azután a "software"-t ennek a végére teszi, így a $x végső értéke az "open source software":

```
$x = "open" . " " . "source";
$x . = " software";
```

Dokumentumbejegyzés: <«

Nem ismerem a < < < hivatalos PHP-beli nevét, pedig magát az operátort állandóan használom. Ha nagyobb szöveget, például e-mail szövegét akarod sortörésekkel együtt beszúrni, a < < < operátort kell a következő példa szerint használnod. A példában a $x változó az első és utolsó sor közötti szöveggel töltődik fel. A beolvasandó szöveg határolására a < < <anytest és az anytext szolgál. Az anytext;-nek új sorban kell kezdődnie, a szöveg pedig a $ karakteren kívül bármilyen karaktert tartalmazhat (a $ jelet itt, éppúgy, mint a kétszeres idézőjelben levő sztnngben, egy \ jellel kell kiemelni):

```
$x = <<<anytext
Dear Coriolis Editors,
The PHP Black Book is proving truly magnificent and worth
every cent.
It lives permanently next to my copy of Coriolis's Linux Core
Kernel
Commentary. B.
Gates III
anytext;
```

Az operátort akkor is használhatod, ha egy kódot úgy akarsz ellenőrizni, hogy egy oldalon megjeleníted a kódot, és le is futtatod, hogy lásd az eredményt. Töltsd be a kódot a $x-be pontosan úgy, ahogy a példában az e-mailt töltöttük, a print()-tel jelenítsd meg a $x-et egy oldalon, hajtasd végre a kódot az eval()-lel, majd jelenítsd meg a kód által beállított változókat. A következő egyszerű példában $c-nek értéket adunk, megjelenítjük, majd értékeljük a kódot, végül megjelenítjük a $c-t. Akik a kód eredményét igazolni akarják, a $c-re a végén 4-et kell, hogy kapjanak.

Mivel a kód S karaktereket tartalmaz, az összes $-t cseréld Vra (a $x-en belül ezek $-ként lesznek eltárolva). Annak érdekében, hogy a $x biztosan megjelenjen egy HTML-oldalon, a print()-en belül tedd a $x-et htmlspecialchars()-be. A kód minden sora egy láthatatlan új-sorkarakterreí végződik, \n, amelyet a HTML figyelmen kívül hagy, így a \n
\n-re cserélésére használd a str_replace()-t, hiszen a <br> sortörést hajt végre a HTML-oldalon:

```
$x = <<<codedelimiter
\$a = 2;
\$b = 2;
\$c = \$a + \$b;
codedelimiter;
eval($x);
$x = str_replace("\n", "\n<br>", $x)■
print(htmlspecialchars($x));
print{"<br>c: " . $c);
```

Növelő/csökkentő operátorok: ++ - -

A + + és -- operátorok változó előtti vagy utáni használatával növelni vagy csökkenteni lehet azt, hasonlóan a C nyelv egyenlőségjeléhez. Ha a $b-t 7-re állítod, és azt írod be, hogy "print(+ +$b);", a $b 7-ről 8-ra nő, és utána jelenik meg. A "print($b+ +);" fordított sorrendben működik, a $b 7-ként megjelenik, utána majd 8-ra nő. Ha a $b-t egy számlázó programban az aktuális számla sorszámaként használod, a **++$b** használatával kapod a következő sorszámot.

A -- operátor csökkenti a változót, így a —$b csökkenti a $b-t, és utána a csökkentett értéket adja vissza, míg a $b~ az aktuális értéket adja vissza, majd csökkenti a változó értékét. Ha a $b a nyomtatóban lévő papírok számát tartalmazza, akkor minden oldal nyomtatása után az — $b használatával csökkentheted, majd megjelenítheted a még bent lévő papírok számát.

Hibaellenőrző: @

A (§. operátor elrejti a függvények és különböző kifejezések hibáit, amely kereskedelmi weboldalakon hasznos lehet, hiszen az alapértelmezett hibaüzenetek a felhasználók számára rejtélyesek és általában haszontalanok. Ezeket az üzeneteket jobb, ha elrejted, és ha a hibákat te magad veszed észre, hasznos, tartalmas hibaüzeneteket hozhatsz létre helyettük. Ha MySQL-függvényeket használsz, tegyél a függvény neve elé @ jelet, ellenőrizd az eredményt, majd a MySQL-hibamezők használatával fejtsd meg a hibákat. (A MySQL-lel az 5. fejezetben foglalkozom.) Ha egy függvénynek nincsen olyan egyedi hibajelentési mechanizmusa, mint a MySQL-nek, a PHP **$php_errormsg-ben** keresheted meg a hiba PHP-s változatát.

A következő példát egyenesen egy MySQL-alapú oldalról vettem. A **$sql-t** a **mysql_query()** MySQL-függvénybe helyezve egy lekérdezés beszúrásának az eredményét ellenőrzi, és üzenettel jelzi, ha hamis az eredmény. Néhány függvény bonyolultabb ellenőrzést igényel, mert azok a hamist és a nullát egyaránt jelentő zérót adhatják eredményül, amelyet a PHP hamisnak is értelmezhet, hacsak nem a mágikus = = = összehasonlító operátort használod az ellenőrző kódban. Ellenőrizd az egyes függvények leírásában, hogy miket adhatnak eredményül:

```
if(J$query_resulc =
    @mysql_query($sql)) {
    print("Error in query:" . $sql . "<br>Error number: ".
          mysql_errno{) ", error text: " . mysql_error() ) ;
```

Bináris operátorok: & | ^ ~ « »

Feltételezem, azzal tisztában vagy, hogy a karakterek bináris sztringként vannak tárolva, az egészek bináris sztringként vannak tárolva, és egyáltalán, a számítógépedben minden különböző hosszúságú bináris sztringként van tárolva. Hogyan alakíthatod a PHP-ban a biteket? Persze, hogy a bináris operátorokkal!

Az első példa az 1 és 2 számokat használja, egyrészt egész számként másrészt egyszerű tárolt karakterekként. Az egész alak és a karakteralak bináris megjelenítése is a bináris 01-gyel végződik az 1 esetében és bináris 10-zel a 2 esetében. Napjaink 32 bites PC-in az 1 nem más, mint 31 db 0 után egy l-es, az *1* karakter pedig 00110001.

Ha a biteket a logikai ÉS (AND) művelettel vonod össze, tehát $b = 1 & 2;, a $b-ben a bitek ott 1 értékűek, ahol a megfelelő bitek az l-esben és a 2-esben is 1 értékűek, a többi esetben 0 értékűek. A 01 (azaz az 1) és a 10 (azaz a 2) AND művelettel a 00 eredményt adják, így 1&2 eredménye 0 és "1"&"2" eredménye 0.

A logikai VAGY (OR) művelet, példánkban a $b = 1 | 2; eredménye ott 1, ahol az 1 vagy a 2 megfelelő bitjei közül legalább az egyik l-es, mindenhol máshol 0. A 01 (azaz az 1) és a 10 (azaz a 2) OR művelettel a 11 eredményt adják, azaz 1 | 2 eredménye 3 és "1" | "2" eredménye *3*.

A logikai XOR (kizáró) vagy $b = 1 "" 2;, eredménye a $b-ben ott 1, ahol az 1 vagy 2 megfelelői bitjei közül az egyik és csakis az egyik 1, egyébként 0. A 01 (azaz az 1) és a 10 (azaz a 2) XOR-művelettel a 11 eredményt adják, azaz 1 ~ 2 eredménye 3 és "1" ~ "2" eredménye "D' (a meg nem jeleníthető karaktert jelző karakter). Hoppá, mi történt a "1" ~ "2"-vel? Az *1* karakter alakja 00110001, a 2-é pedig 00110010. így a $b tartalma 00000011, ami nem más, mint egy meg nem jeleníthető karakter, amely a böngészőben bármi lehet, akár szóköz, akár egy kis doboz, ami a megjelenítési hibát jelzi.

A logikai NO a $b = ~ 2;, az összes bitet invertálja, azaz l-re állítja a $b-ben, ha a 2-ben a megfelelő bitek 0-k és fordítva. A 10 (azaz a 2) invertált alakja 01, és a 10 előtti sok 0 mind l-es lesz, így a -—2-ből -3 lesz. A ~"2" eredménye 11001101, ami az ékezetes nagy i, az /.

Izgalmas idáig? Már csak két bináris operátor van hátra, a jobbra tolás >>, és a balra tolás < <. Mindkettő egy és az aktuális mezőhossz között az általad meghatározott helyi értékkel teszi odébb a biteket. Mindkettő eltávolítja a végéről a biteket, és nullát tesz a másik végére. Néhányan azt mondják, hogy a levágott bitek a bináris mező másik végére kerülnek, de azokon a rendszereken, amit én használok, ez nem igaz. Ha a $b-t 4-re állítom, $b = 4;, és utána a biteket egy helyi értékkel jobbra tolom, $b >>= 1;, akkor a $b 2 lesz. A következő jobbra tolás eredménye 1, a következőé 0, és utána már végig nulla. Ismét a $b = 4;-val kezdve, egy helyi értékű balra tolás ($b <<= 1;) eredménye 8, két helyi értékű tolás ($b <<= 2;) eredménye pedig 16. Ha elég sokáig balra tolod a biteket, végül negatív számot kapsz, mert az 1 bitet az előjeljelző pozícióba juttattad. Ezután nullát kapsz, mert az utolsó l-es bit is kikerült a mezőből. A bináris operátorok is használhatók a $a = $b > > 1; formában és számokkal is, mint a $a = 126 > > 1;.

Végrehajtás-operátor:

A billentyűzet bal felső sarkában magányosan álldogál a ritkán használt visszaposztróf, a \ A visszaposztróf fontos akar lenni, de a PHP-t leszámítva mindenki összekeveri szegényt az idézőjellel. A PHP viszont úgy kezeli, mint a system()-függvényt, és mindent, ami visszaposztrófok között van, az operációs rendszernek küld végrehajtásra. Bármit is küld vissza az operációs rendszer, az egy sztringbe kerül, amit feldolgozhatsz vagy megjelenít-hetsz. Jóllehet a visszaposztróf hasonlatos a system()- és az exec()-függvényekhez, a visszaposztróf az egyetlen, ami az outputot számodra teljesen megfelelően adja vissza.

A példa a Unix/Linusos Is parancsot használja a fájlnevek és helyük későbbi megjelenítés vagy esetleg elemzés céljából történő $list-be való listázására. Jobban járnál, ha a példában bemutatott egyszerű fájlnévlistát az egyik PHP-fájlfüggvény használatával kapnád meg, de előfordulhat, hogy olyan paranccsal találkozol, amelynek nincsen a PHP-ban pontos vagy egyszerűen használható megfelelője:

```
$1i S t     =      ' 1S ' ;
```

Hivatkozások: &=&

A hivatkozások a változók neveinek aliasai. Ha a $x-et 5-re állítod, a $x egy olyan név, amely az 5, egészmezőben tárolt értékre mutat, és a PHP lehetővé teszi, hogy más nevekkel is ugyanerre az egészmezőre mutassunk. A $f =& $x;-vel beállíthatod, hogy a $f ugyanerre az egészmezőre mutasson. Az 5-öt tartalmazó egészmező változatlan marad, és a $x ugyanúgy erre az egészmezőre mutat. Csak annyi történt, hogy a PHP szimbólumtáblázatában egy újabb név mutat ugyanarra az egészmezőre. Ha a $f-hez 3-at hozzáadsz, az egészmező 8-ra változik, és a $x használatakor a $x is 8-at ad eredményül.

Nem használhatsz hivatkozást egy másik hivatkozás megváltoztatására, így az eddigi pél-dánknál maradva, a $f-et új mezőre irányítva a $x nem fog arra az új mezőre mutatni. Ha van egy Sb-nek nevezett mező, és a $f =& $b;, használatával $f ugyanarra a mezőre mutat, mint a Sb, csak a Sf-et változtatod. A $x továbbra is ugyanarra a mezőre mutat. Ugyanez vonatkozik a hivatkozások feloldására is: ha feloldod $f-et, **unset($f),** eltávolítod $f-et, de a $x-et és az 5-öt tartalmazó mezőt érintetlenül hagyod.

A következő példa első sora nem működik. A második sor nem más, mint a hibaüzenet ele-je, ha mégis kipróbálod a kódot. A kód megpróbál hivatkozást adni egy értéknek, de a PHP nem tudja, hogyan tegye ezt. A PHP csak olyan mezőkhöz tud hivatkozást adni, amelyeket rendes hozzárendelő paranccsal hoztunk létre:

```
$a   =&   5;
Parse   error:   parse   error,   expecting   'T_NEW'   or   'T_STRING'
```

Vannak nyelvek, amelyekben a mezők részeihez, illetve együtteséhez is rendelhetsz hivat-kozást, de a PHP-ban nem. Mikor kipróbáltam a **$b** =& **$a[15];,** a PHP szintaktikai hibát jelzett, és amikor a $a =& $b . $c;-vel kipróbáltam az együttes hivatkozást, azt a PHP csak az első, a $b mezőhöz hozta létre.

A függvényekben a paraméternevekhez az egyszerű **and** karaktert, az &-et adva a paramé-terrel hivatkozhatsz egy, a függvényen kívüli mezőre. Ez sokkal jobb, mint ha a külső

mezőt bemásolnád. A következő példában a **plus_one()** függvény fogad egy változót, hozzáad egyet, majd megjeleníti a változót. Mivel a példa elején a $x 11-re van beállítva, a függvény 12-t eredményez, és az első **print** utasítás a 12-t jeleníti meg. Azonban a második **print** utasítás a 11-et, az eredeti, változatlan $x-et jeleníti meg.

```
$x = 11
function plus_one($value)

    return(++$value);

print(plus_one($x));
print($x);
```

Most szúrj csak egyetlen karaktert a példába, az St-et a **$value** elé. így már a második print-utasítás is a 12-t jeleníti meg a 11 helyett, hiszen a $x által mutatott mező másolata helyett az **&$value** most már ugyanarra a mezőre való hivatkozás, mint amire a $x mutat:

```
$x = 11
function plus_one(&$value)

    return(+ + $value) ;

print(plus_one($x));
print($x);
```

A —*&C* használatával a függvények eredményeként kapott hivatkozásokra is létrehozhatsz aliasokat. Ha az *előző* kód $a = **&plus_one($x);** lenne, a $a egy a függvény eredményeként kapott értékre vonatkozó alias, ezáltal a $x változóra vonatkozó hivatkozás lenne. A hivatkozások nem csak változókra, hanem objektumokra is vonatkozhatnak.

Egyéb hivatkozások a függvényekben a **global** utasítással, az objektummódban pedig **$this**-szel hozhatók létre. A függvényeket a 10., az objektumokat a 17. fejezetben tárgyalom. A következő példa a **find_variable()**-függvényt mutatja be, amely érték helyett hivatkozást ad eredményül, hiszen a függvény definiálásakor & karakter van a függvény neve előtt. Ha a find_variable() egy belső mezőre vonatkozó hivatkozást eredményezne, a mező eltűnne, amikor a függvény véget ér, így a **find_variable()** külső mezőket igényel. Ezeket a külső mezőket vagy a paraméternevek elé írt &-el vagy a **global** utasítással biztosíthatod. Ha már egyszer vannak bemenő hivatkozásaid, és a függvényt úgy definiáltad, hogy eredményül hivatkozásokat adjon, akkor tetszőleges feldolgozással választhatod ki az eredményül adódó hivatkozást. Ebben a rövid példában a **find_variable()** egyszerűen az inputot adja vissza. Hogy tesztmezőt hozzunk létre, a $b-t 5-re állítjuk, majd beadjuk a **find_variable()** függvénybe, amely eredményül egy a $b-re vonatkozó hivatkozást ad. A =&-el a $a-t a **find_variable()** eredményére hivatkoztatjuk, és mivel az eredmény a $b-re vonatkozó hivatkozás, a $a most ugyanarra a mezőre hivatkozik, mint a $b. Ha ezután $a-t megváltoztatjuk (a példában eggyel növeljük), a $b ugyanarra a megnövelt mezőre mutat és az utolsó sor a "a: 6, b:6"-ot jeleníti meg:

```
function    &find_variable(&$parameter)

    return($parameter);
```

```
$b = 5;
$a -Se f ind_variable ( $b) ;
$a + +;
print("a: " . $a . ", b: "
. $b) ;
```

bcilins

Műveleti sorrend

A következő lista az operátorokat a műveletek prioritása alapján állítja sorba, a legmagasabb prioritásúval kezdve. Az ugyanolyan prioritású operátorok jobbról balra haladva hajtódnak végre. Zárójelekkel (), megváltoztathatod a műveletek végrehajtási sorrendjét:

new

! —h -\— **(int) (double) (string) (array) (object)**

/ /o

- **< <= > > =**
- **== i= = = = ! = =**
- **&**

I

- **) :**
- **= + = .= *= /= .= %= & = i = - = ~= <<= >> =**
- **print**
- **AND**
- **XOR**
- **OR**

Ellenőrzés és szerkezet

Ebben a részben azokról az utasításokról lesz szó, amelyekkel ellenőrizheted a szkripted végrehajtódását, így megmutatom az if(), **else** és **elseif()** döntési utasításokat, majd egy alternatív szintaktika is szóba kerül. A switch()-utasítással, amely hasonló a más nyelvek select utasításához, nagyszerűen ki lehet váltani az if()-utasítások hosszú sorát. A **whileQ**-és for()-utasításokkal adott feltételek mellett működő ciklusokat építhetsz a kódba, hasonlóan a más nyelvek **while()-** és **for()**-parancsaihoz.

Az include()- és requireQ-utasításokkal más fájlokból származó kódokat ágyazhatsz a szkriptedbe, és mind a kettő ugyanúgy működik, mint a C-beli megfelelője. Van pár csapda,

amire figyelned kell, ha PHP-ban használod ezeket, ezért az új require_once() és include_once() utasítások segítenek azon a gyakori problémán, ha egy fájl - akaratod ellenére - többször ágyazódik be.

if()

Az if() PHP-beli használata hasonló annak más nyelvekbeni használatához. Ez a rész be-mutatja a if() használatának trükkjeit, és ismertetek néhány elkerülendő csapdát is. A következő rövidke példa a $a értékét teszteli, és ha az nagyobb, mint 0, akkor kiírja, hogy "ok":

```
if($a >
    0){
    print("ok"); }
```

A PHP a 0-t hamisnak értékeli, minden 0-nál nagyobbat vagy 0-nál kisebbet pedig igaznak. Vannak nyelvek, amely a negatív számokat hamisnak értékelik, így könnyen csapdába kerülsz, ha más rendszerekből származó adatokat dolgozol fel. Mivel a PHP a karakteradatatokat automatikusan egészre konvertálja, a 0-t is hamisnak tekinti, hiszen az a 0 hosszúságú sztring. Ha azt írod, hogy if($a == falsé), mind a 0, mind a "" hamisat adna eredményül, így az if($a = = = false)-t kell használnod, miként azt az „Operátorok" részben megbeszéltük.

A PHP a mezőket egyenesen logikai értékekké konvertálja, így az if($a != 0) helyett elegendő az if($a), és a $a-t a PHP fogja az if() számára igazra vagy hamisra konvertálni. Az if utasításokat az if(I$a) formában is megadhatod, amely akkor ad eredményt, ha a $a hamis. Mivel ezek a változatok mind nem szándékolt eredményhez vezethetnek, javaslom az értékek és típusok összehasonlításának formális használatát, azaz az if($a = = = true) és if($a = = = falsé) formákat.

Az if több feltételt is vizsgálhat, ha a feltételeket az and-del kapcsolod össze. Mivel a feltételek balról jobbra kerülnek kiértékelésre, mielőtt az értéket vizsgálod, ellenőrizheted, hogy az adott változó létezik-e. A következő példa az isset()-nél megáll, ha a $a nincs meghatározva, így nem fog a hiányzó $a-ra vonatkozó hibaüzenet megjelenni. Ezzel a rövid kóddal opcionális változók meglétét ellenőrizheted (például egy űrlapmezőt):

```
if(isset($a) and $a >
    0) { print ("ok") ,-
```

else

Az else-utasítást nagyszerűen lehet bármilyen érték esetén használni, és pontosan úgy működik, mint bármely másik nyelvben. Az if () után használható, ahogy a következő példa is mutatja:

```
if($a == $b)(
    print("equal");
```

```
else

    print("not    equal");
```

elseifQ

Az elseif()-utasítással az if()-utasítás különböző feltételeire lehet ugrani, és hasonló a más nyelvekben használt elseif()- és else if()-parancsokhoz:

```
if($a    --    "hot")

    print("Turn on air-conditioner");

elseif($a = = "warm")

    print("Enjoy the weather");

else

    print("Turn on heater");
```

switch ()

Az elseif()-utasítás példájában a több választási lehetőséget a switch()-utasítással is meg lehet jeleníteni. A switch() használatával könnyebb a több lehetséges érték és művelet kezelése, íme az előző példa switch()-utasítással megvalósítva:

```
switch($a)

    case "hot":
        print("Turn on air-conditioner");
        break;
    case "warm":
        print{"Enjoy the weather");
        break;
    default:
        print("Turn on heater");
```

A switch()-utasítással változót vagy kifejezést hasonlíthatsz össze, jelen esetben a $a-t, először az első case utasítás után álló kifejezéssel, "hot", és egyezőség esetén a case utasítás utáni kód végrehajtása történik meg. Ha nincs egyezőség, a switch() a következő case-re ugrik és ismét összehasonlít.

A default:-utasítás egyenértékű az if()-fel használt else-utasítással - ha az összes case-uta-sítás sikertelen, akkor ennek végrehajtása valósul meg. A default:-utasítás nem kötelező, miként az else sem kötelező az if () után.

Miután a switch() befejezi a case-utasításban talált kódot, folytatja a teljes rész lefuttatását, hacsak nincsen break-utasítás, amelynek hatására kiugrik a switchQ futtatásából. (Bizonyos

nyelvek a break megfelelője nélkül is kiugranak a switch()-ből, így a következő példa egyenértékű megfelelője nem mindenhol használható.) Az előző példánál maradva előfordulhat, hogy azt akarod, hogy "hot" esetén a légkondicionáló is bekapcsoljon, és az időjárásnak örvendő üzenet is megjelenjen. Csak annyi változtatást kell tenned, hogy a "hot" case után kihagyod a break-et:

```
switch($a) {case
    "hot":
        print{"Turn    on    air-conditioner");
    case   "warm":
        print{"Enjoy    the    weather") ;
        break;
    de fault:
        print("Turn    on   heater");
```

Előfordul, hogy amikor azt szeretnéd elérni, hogy különböző értékek esetén is ugyanaz a művelet történjék meg, és sok nyelvben elegendő az értékek egyetlen case utasításban való megjelenítése. A PHP-ban minden egyes értékhez külön case-utasítás szükséges, amellyel jobban nyomon követhető az egyes értékek dokumentálása (akár minden sorba megjegyzés beszúrásával). A következő példában a programozó kedveli a hőséget, így kikapcsolva hagyja a légkondicionálót:

```
switch{$a) {
    case    "hot":
    case   "warm":
        print("Enjoy    the    weather");
        break;
    default:
        print("Turn    on    heater");
```

whileQ

Ha egy tömböt vagy egy fájlt, vagy egy adatbázis egy sorát olvasod be, jó hasznát veszed a while()-nak a program futásának szabályozására. A while() egyetlen PHP-beli vetélytársa a for(), de jómagam inkább a while()-t használom, mert az a legtöbb függvény és szerkezet esetén jobban megfelel, mint az if(),

A nyelvek a while() különböző változataival lehetővé teszik a feltételek ciklus előtti vagy utáni tesztelését, így a ciklussal legalább egyszer lefuttathatod a kódot. A PHP-ban ez a do while()-utasítással érhető el. A következő példa egy általános while()-ciklust mutat be, amely egyszer sem fog lefutni, hiszen a $a már egyenlő a $b-vel. A példa második része a do whíle()-változatot mutatja, amely legalább egyszer lefut a $a==$b egyenlőség tesztelése előtt. Az első rész egyetlen sort sem nyomtat, míg a második pontosan egyet:

```
$a   =   $b   =   5;
while($a    !=
$b)

    print("This will never print");
```

```
do {
    print("This will print at least once"); }
while($a ! = $b);
```

A while()-ciklus, ahogy azt a következő példában láthatjuk, a **Iist()-** és each()-utasításokkal kombinálva kiválóan használható a tömbökön belüli lépkedéshez. A példa létrehoz egy rövid tömböt, a tömbmutatót a tömb elejére állítja, majd a whileQ használatával keresztüllépked a tömbön. Az each()-utasítás egyszerre egy tömbelemet, a tömb végén pedig hamis értéket eredményez. A list()-utasítás átveszi az each()-tól az adatokat, a tömbelem számát az első' mezőbe, a $k-ba, a tömbelem értékét pedig a második mezőbe, a $v-be helyezi:

```
$white_crystalline_substance[] = "C12H22O11";
$white„crystalline_substance[] = "C8H10N4O2.H20";
$white_crystalline_substance[] = "NaCl";
reset($white_crystalline_substance);
print("Warning, the most abused and addictive substances are:");
while(list($k, $v) = each($white_crystalline_substance))
    {
    print           $v);
    }
```

Megjegyzendő, hogy nem szükséges a tömbelemek létrehozásakor indexértéket megadni, a PHP az üres szögletes zárójelet látva automatikusan a következő üres indexszámot használja fel. A tömbök további részleteit a 3. fejezetben találod.

Az adott példában a **while()** háromszor fut át a kódon, minden egyes elemnél egyszer, és amikor az each() hamis eredményt ad, akkor áll meg:

```
Warning, the most abused and addictive substances are:
C12H22O11
C8H10N4O2.H20
NaCl
```

for()

Bizonyos alkalmakkor, például amikor a lista elemeit meg kell számolni vagy az elemek sorszámát kell megjeleníteni, akkor a **for()** jobb, mint a **while().** A **for()**-utasítás, akárcsak C-ben, három, pontosvesszővel elválasztott kifejezést tartalmaz, és pontosan úgy értelmezi a kifejezéseket, mint a C. íme egy gyors példa, amely egytől háromig jeleníti meg a számokat:

```
for($i = 1; $i <= 3;
    $i++) {
    print($i);
    }
```

Az első kifejezés a **for()**-ciklus kezdetekor fut le, és egy megfelelő kezdőértékű indexet hoz létre. Ha a megfelelő értéket tartalmazó indexmező már létezik, akkor az első kifejezés elhagyható, de ne felejtsd el a pontos vesszőt kitenni, mert a **for()** csak így fogja tudni, hogy a következő kifejezés mire vonatkozik.

A második kifejezés ciklusonként egyszer értékelődik ki a ciklusok elején. Ha az indexmező már elérte a második kifejezés által meghatározott értéket, akkor a ciklus végrehajtása elmarad. A második kifejezés kihagyásával a ciklus végtelenné válik, amiből egy break-uta-sítás használatával törhetsz ki. Ez azonban veszélyes, hiszen a kód későbbi megváltoztatása miatt lehet, hogy a break-utasításhoz soha nem jut el a ciklus, és a végrehajtása elmarad. Ha tudod, hogy pontosan hányszor fut le a ciklus, akkor írd be ezt az értéket a for()-utasí-tásba, és tartsd készenlétben a break-utasításokat a ciklusból való korábbi kilépéshez.

A harmadik kifejezés ciklusonként egyszer, a ciklus végén értékelődik ki. Ezt a kifejezést is ki lehet hagyni, és lehet helyette mondjuk egy print-utasítás. Általában ezt a harmadik ki-fejezést az indexmező eggyel való növeléséhez használják, hogy az indexmező végül elérje a második kifejezésben beállított határértéket, de ez nem szükséges, ha a ciklusban a kód egy másik része ugyanezt teszi az indexszel. Példa lehet erre az each(), a next() vagy a mysql_fetch_row() használata, amelyek mind a következő sorra vagy elemre ugranak.

A következő példa az előző whíleQ-példát használja, és a tömböt az elemek számával egytől növekvő sorrendben jeleníti meg, úgy, ahogy egy TOP 10-es listát számoznál:

```
print("The most addictive substances are (in order of
addictiveness) :") ; for($i = 0; $i <
count($white_crystalline_substance); $i++)
    {
    $n = $i + 1;
    print("<br>" . $n . " " .
    $white_crystalline_substance[$i]);
    }
```

Lehet, hogy ez nem a legelegánsabb megoldás, viszont biztos, hogy a legpraktikusabb, ezért elmondom az előnyeit és a lehetséges alternatíváit is. A tömbök nullától számolnak, így a for()-utasítás az index nullára állításával ($i = 0) kezdődik, és az elemek számánál eggyel kisebb értékig fut, eggyel növekedve, amit a $i++ parancs visz véghez. A kód nem használ tömbmutatót, így a kód elején azt nem kell újraállítanod. A kód nem használja a tömbelemek kulcsát sem, így akkor is működik, ha a tömbből kitöröltek elemeket, vagy ha a tömbelemeket az elemek kulcsától eltérően rendezték sorba. Tulajdonképpen a kulcs egy mennyiség is lehet, például évenként felhasznált kilogramm.

A tömb nullától van számozva, de mivel a listát egytől kezdve kell sorszámozni, a kód $i + 1 diszkrét mezőként létrehozza a $n-t. Ez azt jelenti, hogy a $n számtalan módon használ-ható a print-utasításban, jóllehet a példában csak a sor elejére van helyezve. Ezt a kódot könnyebb megváltoztatni, ha a megjelenítés formátumán változtatnod kell.

requireQ, require_once(), includeQ és include_once()

Ha akkora szkriptet írsz, mint ez a fejezet, akkor legfőbb ideje elővenned a láncfűrészed, és felaprítanod a szkriptet. Ebben a részben segítségképpen bemutatom a szóba jöhető mód-szereket, illetve azt, hogy ehhez melyik PHP include()- vagy require{)-utasítás a megfelelő.

A requireQ- és include()-utasításokról szóló részt kivághatnám a dokumentum többi részéből és külön szerkeszthetném, hiszen e között és a többi rész között nincsenek ke-reszthivatkozások. Amennyiben egy rész nem tartalmaz általam függő hivatkozásnak neve-zett kapcsolattípusokat, egyszerűen és megbízhatóan kivághatod a PHP-kódból. Mi az a

függő hivatkozás, és hogyan tesztelheted a függőségek meglétét? Válaszd ki a kód bármely részét, vágd ki jelenlegi helyéről, és illeszd be bárhová máshová. Vajon a szkripted még mindig működik?

Amennyiben igen, akkor független kódrészt találtál, amely alkalmas arra, hogy különálló, később beágyazandó állományba tedd. A függvények és objektumok kiválóan alkalmasak arra, hogy ilyen állományba tegyük Őket, hiszen a szknptbe bárhová beágyazhatóak (PHP4-ben, PHP3-ban ez még nem működött). Kezdhetsz vagdosni.

Ha a honlapodon mezőgazdasági gépeket értékesítesz, és a szkripted szerszámgépeket jelenít meg, elgondolkodhatsz azon, vajon hol használhatnád fel újra a függvényeket. Tegyük fel, hogy a szerszámgépes oldaladat powertoolpage.html-nek nevezed, és van öt olyan függvényed, amelyeket csak a powertoolpage.html-ben használsz. Ezt az öt függvényt a powertoolpagefunctions.html-be teheted. A más oldalakon is használt formázó függvények mehetnek a formattingfunctions.html-be, az adatbázis-hozzáférési függvények pedig, amelyeket csak olyan oldalakon használsz, ahol terméklista jelenik meg, menjenek a productdbfunctions.html-be. Mivel bármely függvényt bárhova beágyazhatsz, és az a leghasznosabb, ha rögtön az elején beágyazod őket, az oldalad kezdődjön így:

```
require("powertoolpagefunctions.html");
require("formattingfunctions.html");
require("productdbfunctions.html");
```

A require()-utasítás veszi a fájl nevét, és rögtön a PHP-feldolgozás kezdetén, a szintaktikai ellenőrzés alatt, de még a szkript végrehajtása előtt beágyazza a fájlt. így a require()-t nem lehet az if () vagy a hozzá hasonló logikai ellenőrző utasításokkal kontrollálni, emiatt a require() gyorsabb, de kevésbé rugalmas. Az állandóan beágyazott fájlokhoz ezért a require() jobban használható, mint a csupán alkalmanként használt fájlokhoz.

Az include()-utasítás veszi ugyanazt a fájlnevet, de addig nem ágyazza be a fájlt, amíg a program végrehajtása el nem éri az include()-utasítást. Ha egy adott kódszakaszt csak néhány oldalon használsz, tedd ezt egy különálló beágyazott fájlba. Utána használj egy include()-utasítást a fő dokumentumban, és tedd az include()-utasítást egy if()-utasításba, amely eldönti, hogy a külön kódrészre szükség van-e. Tegyük fel, hogy a szerszámgépes oldalaid egy részének van egy visszacsatoló oldala, amely a vásárlókat a szerszámgépek szabadidős használatáról kérdezi. Állítsd a Sask-ot igazra, ha szükség van a kérdőívre, a kérdőívet mentsd el a questionnaire.html-be, majd használd a következő if()- és include()-utasításokat a kérdőív beágyazására:

```
if(5ask) {
    include("questionnaire.html");
```

A beágyazott fájlok tartalmazhatnak include()- és require()-utasításokat, így lehetséges, hogy az a.html oldal tartalmazza a b.html és c. html oldalakat, és a b.html tartalmazza a c.html-t. Ha ez megtörténik, a c.html kétszeresen lesz beágyazva, és a szkripted a kód-duplikáció miatt nem működik. Az include_once()- és require_once()-utasítások segítenek ezen a problémán, hiszen a fájl beágyazása előtt ellenőrzik, hogy az adott fájl nincs-e

már beágyazva. Példánkban a b.html-be nem fog a c.html beágyazódni, ha az a.htlm már tartalmazza a c.html-t. Ahhoz, hogy ez így működjön, az összes c.html-re vonatkozó include()-utasítást include_once{)-ra kell cserélni, mert egy kósza include("c.htmr) még beágyazhatja a c.html-t.

Függvények

A függvényeket teljes részletességgel a 10. fejezetben magyarázom el, hiszen az adatokhoz nincsen különösebben szükség a függvényekre. Ezen fejezet „Hivatkozások: & =&" című részében szó esett a függvényeknek a hivatkozásokhoz való használatáról, a ,,require(), require_once(), includeQ és include_once()" részben pedig a gyakran használt függvények különálló fájlba ágyazásának egyszerű módjairól volt szó.

Kategóriák és objektumok

Az objektumokat részletesen a 17. fejezetben magyarázom, hiszen az adatokhoz nincsen különösebben szükség a objektumokra. Ugyanúgy kezelhetőek, ahogy az elő'ző részben leírt függvények. Az objektum az adatok egy olyan típusa, amely gyakran használható ott, ahol sztringeket és számokat használsz (például egy tömb elemeiben). Azt, hogy egy adatelem objektum-e, az is_object()-függvénnyel ellenőrizheted.

Adatok az adatbázisoknak

A legtöbb relációs adatbázisban az SQL használatával viszel adatokat az adatbázisba, és az SQL idézőjelekkel határolja a sztringeket. így ha az SQL-lel sztringet akarsz elfogadtatni, a sztringen belüli idézőjelet valami mással kell helyettesítened. Az addslashesQ-utasítás pontosan ezt teszi. Az addcslashesQ-utasítás további ellenőrzési lehetőséget biztosít, így egy karakterlistával, amely meghatározza, hogy melyik karakterek változnak, bármely karakter lecserélhető. Az addcslashesQ-utasítás C-szerű konverziót végez a nem megjeleníthető karakterek nyolcas alapú megjelenítésére konvertálásával. A MySQL-hez addslashes()-re van szükség. Az addcslashes()~t még soha semmire nem kellett használnom.

A stripslashes()- és stripcslashes()-utasítások ennek pontosan az ellenkezőjét csinálják, és előfordulhat, hogy szükséged lesz rájuk. Ha a MySQL-nek címzett adatokhoz per jelet teszel, a MySQL visszakonvertálja a helyes formátumba az adatokat, mielőtt az adatbázisban elmentené. Ezután, ha az adatot visszakeresed, a MySQL a megfelelő formátumban jeleníti meg azt, így nincsen a stripslashes()-re szükséged. A stripslashes() használata előtt ellenőrizd az adatbázisodat.

Adatok a HTML-nek

Képzeld el, hogy a HTML használatáról írsz segédletet, és azt akarod, hogy a <br> karaktérként jelenjen meg, nem pedig sortörésként. Csak tedd a htmlspecialchars()- vagy htmlentities()-függvényeket a
-t tartalmazó sztring köré, amely a <-t <-ra és a >-t >-ra konvertálja. A htmlspecialchars()-utasítás a <, >, ", "", és & karaktereket változtatja meg, míg a htmlentities() minden olyan karaktert, amelynek van HTML-beli

megfelelője. Mindkét függvényben lehetőség van egy második paraméter megadására, amellyel meghatározhatjuk miként kezeljék az idézőjeleket. Az alapbeállítás azonban megfelelően működött minden alkalommal, amikor ezeket a függvényeket használtam.

Dátum és idő

A PHP több dátum- és időfüggvényt tartalmaz, mint a legújabb csúcstechnológiájú karóra. Tekintsd ezt egy gyors tájékoztatónak arra, hogy mikor használd a függvényeket. A részletes példákat a „Gyors megoldások" részben találod.

Juíianus-dátum

A *Julianus-áitum* különbözik a következő részben bemutatott Julianus-naptártól. A Julia-nus-dátum olyan formátum, melynek egy különlegessége van: nincsen benne hónap. Két olyan Julianus-dátum van, amit valaha is használni fogsz.

A/«/zím«s-napszámlálót, amely Kr.e. 4000-rel kezdődik, PHP-ben a dátumok kiszámolására használod. Amikor a PHP egy dátumot az egyik naptárból a másikba vált át, az eredeti naptárból a Julianus-napszámlálóba váltja át, és csak ezután a kívánt naptárformátumba. A unixtojd()-függvény a Unix-időt Julianus-napszámlálóba váltja át, a Julianus-napszámlálót pedig a jdtounixQ váltja Unix-időre.

Az IBM *Julianus-dátum* az év és az év napja (három számjegyben) formátumban jeleníti meg a dátumot, így a Juhanus-nap a dátumformátumtól függetlenül változatlan marad. A 2001. május 2-a Julianus-dátumban 2001122. Az évet és az év napját a date()-, getdate()- és localtimeQ-függvények használatával kapod vissza, és velük hozhatod létre a Julianus-dátu-mot.

Idő

Ismered a reggel 8.00 és az este 7.00 órát, tudod azt is, hogy az este 7.00 24 órás üzemmódban 19.00. Azt viszont talán nem tudod, hogy a webszerver miért jelezhet mókás időt, illetve hogyan állíthatod be a pontos időt. íme egy gyors magyarázat.

Greenwichi idő (GMT)

Amikor a brit haditengerészet uralta a világot, a greenwichi idő (GMT) volt a nemzetközi időszámítás alapja. (Greenwich, a Londonhoz közeli városka ideális tavaszi kirándulóhely, az idő megjelenítését és történelmét bemutató pazar kiállítással. Ha az időről vagy a londoni és greenwichi városnézésről szeretnél hasznos linkeket, hagyj üzenetet a honlapomon petermoulding.com!)

A Windows alapú számítógépeken megjelenő idő a GMT-hez viszonyított helyi idő. Ausztrália az egész világ előtt jár, így 10 órával a GMT előtt van, míg New York 5 órával utána.

UTC

Természetesen az amerikaiak nem tudták elviselni, hogy a brit haditengerészet uralja a világot és az időt, így az Egyesült Allamok kitalálta a Coordinated Universal Time-t (UTC). A Coordinated Universal Time lehető'vé teszi, hogy a számítógépedet az USA haditengerészetének Washington D.C-ben található obszervatóriumának atomórájához igazítsd a Network Time Protocol (NTP) segítségével, amely a User Datagram Protocollal kommunikál (UDP, olyan valami, amit akkor kell beállítanod, ha a számítógéped proxy szerver mögött van). A UTC a Csendes-óceánon található nemzetközi naptárvonaltól számol, és pontosságát az egész világ több atomórájának idejét összehasonlítva ellenőrzik.

Az általad használt operációs rendszertől és dátum/idő beállítástól függően a fájlokban használt és megjelenített dátum és idő lehet UTC vagy helyi idő. A munkaállomásokon általában a helyi időt állítják be a nyári és téli időszámítás figyelembevételével, persze előfordulhat ettől eltérő eset, amely váratlan dátumot és időt eredményezhet. A legtöbb szervert a UTC-hez igazítják, és olyan függvényeket használnak, amelyek a helyi időt adják eredményül, azonban vannak rossz szerverek, illetve olyanok, amelyek rossz időzónára vannak állítva. Dolgoztam már szerverek olyan hálózatával, ahol az összes szerver egy időre volt állítva annak ellenére, hogy különböző időzónákban helyezkedtek el, de legalábbis különböző időzónában élő vásárlókat szolgáltak ki. A PHP lehetővé teszi mind a helyi, mind a UTC dátum és idő használatát.

Ha az adatbázisok között összehangolod a tranzakciókat, és a másodperc töredékével egyenlő eltérés van, az a hálózati terjedési késleltetés. Ha az eltérés néhány percnyi, akkor az időt valakinek a karórájáról állították be, így kérd meg a hálózatot fenntartó embereket, hogy rendesen hangolják össze a szervereket. Az egy órás eltérés oka az, hogy az egyik gépen van nyári időszámítás, a másikon pedig nincs.

Az szerverek össze nem hangolt UTC-dátumait és az időt a PHP time()-függvénnyel hasonlíthatod össze. Ahhoz, hogy a helyi időt megkapd, használd a date()- és getdate()-függ-vényeket, amelyek az időzónákra és a nyári időszámítás használatára vonatkozó beállításokat is tartalmazzák.

Naptár

A Római Birodalomban a Hold járását alapul vevő naptárt használtak. Július Caesar ezt teljesen megváltoztatta. Gergely pápa egy Július Caesarnál is pontosabb naptárral rukkolt ki. A PHP nem csak a római naptár használatát teszi lehetővé. Ha egy történelemmel foglalkozó honlapot kell készítened, a PHP a számodra tökéletes szkript nyelv.

Zsidó időszámítás

A zsidó naptár a legrégibb a PHP által kezelt naptárak között. A naptár Kr.e. 3761-től kezdődik, eredetileg a Holdhoz igazodott, és számtalan merev szabállyal tartja az évet a Nappal szinkronban. Mivel a PHP dátumkonvertáló rendszere Kr.e. 4000-ig működik, a PHP képes a zsidó naptár kezelésére. A jdtojewish()-függvény a Julianus-napszámláiót a zsidó naptárra konvertálja, és a jewishtojdQ konvertálja vissza a dátumot a Juliánusz-napszámlálóra.

Julianus-naptár

Július Caesar Kr.e. 46-ban hozta létre a Julianus-naptárt 12 hónappal, évente 365,25 nappal, és négyévente szökőévvel, amellyel a 0,25 napot korrigálja. A júliust az általa legjobban imádott emberről, önmagáról nevezte el. A jdtojulian()-függvény a Julianus-napszámlálót a Julianus-naptárra konvertálja, és a juliantojd() konvertálja vissza a Julianus-naptár dátumát a Julianus-napszámlálóra.

Gergely-naptár

A Julianus-naptár a 16. századra 11 nap késében volt. XIII. Gergely pápa a szökőévekre vonatkozó szabályt úgy változtatta meg, hogy a századfordulós évek csak akkor szökőévek, ha 400-zal is oszthatóak. Ennek a változásnak köszönhetően egy év átlagosan 365,2425 napra csökkent, így már csak 0,000411 nap a különbség. A jdtogregorian()-függvény a Julianus-napszámlálót a Gergely-naptárra konvertálja, és a gregoriantojd() konvertálja vissza a Gergely-naptár dátumát a Juhánusz-napszámlálóra.

A Francia Köztársaság naptára

A Francia Köztársaság naptárát 1793. október 24-én, a köztársaság megalakításának egyéves évfrodulóján kezdték alkalmazni. Ez a naptár 12 harminc napos hónapból állt, és számtalan reformkísérlet után 1806-ban törölték el. A jdtofrench()-függvény a Julianus-napszámlálót a köztársasági naptárra konvertálja, és a frenchtojd() konvertálja vissza a Julianus-napszámlálóra.

Húsvét

Ha tudni szeretnéd, hogy 1970 és 2037 között milyen napra esik a húsvét, írd be az easter_date()-függvénybe az évet úgy, ahogy az a Gyors megoldások „Húsvét" részében látható. Az ettől eltérő évekre az easter_days()-függvény használatos, ahogy a Gyors megoldásokban látható. Az easter_date() a Unix időjelzést használja így a Unix időjelzés által lefedett szűkebb időintervallumra használható, míg az easter_days() a Julianus-napszámlá-lóval sokkal nagyobb időintervallumot fed le.

ICAP

Az Internet Calendar Access Protocollal és a PHP ICAP-függvényekkel az Internet alapú naptárakhoz férhetsz hozzá és oszthatod meg őket. A www.ietf.org/, www.imc.org/ és www.w3.org/ helyeken sokkal többet megtudhatsz az Internet-protokollokról és a naptárakról. Az ICAP-pal csatlakozhatsz az MCAL-hoz, amelyet most mutatok be.

MCAL

Az MCAL, azaz a Modular Calendar Access Library egy olyan rendszer, amellyel modulokhoz csatlakozva, különböző naptárakhoz férhetünk hozzá. A MCAL kód, dokumentáció és modulok a http://mcal.chek.com/ címen érhetők el.

A naptárak lehetnek a gépeden, a LAN-en, ICAP-szervereken az Interneten, vagy másmilyen típusú szervereken, amelyeken van mcal plug-in modul. A naptárakat lekérdezheted, eseményeket kereshetsz, emlékeztetőt (ezeket triggernek nevezik) és ismétlődő eseményeket állíthatsz be. Az MCAL-függvények PHP-okumentációja a www.php.net/manual/en/ref.mcal.php címen többek között az mcal_opem()-függvényt

is tartalmazza, mellyel kapcsolatot hozhatsz létre az MCAL-szolgáltatáshoz. Az MCAL külsőre és működésre is nagyban hasonlít az IMAP-hoz, amellyel a 15. fejezetben foglalkozom, így ha van tapasztalatod az IMAP-pal, könnyű lesz az MCAL-t megértened.

Matematika

A számítógéped sztenderd matematikai függvényeinek korlátjáig könnyen használhatod a PHP egész, és lebegőpontos matematikai függvényeit. Ha nagyobb számokkal akarsz dolgozni, a PHP-vel hozzáférsz a kibővített matematikai szoftverhez. A GMP-vel a nagyobb egészekkel számolhatsz, a BCMath-tal pedig mindennel, amit a GMP nem tartalmaz.

Egész matematika

™

A PC-n a sztenderd egészek 32 bites számokra vannak korlátozva, és kizárólag előjeles egészként kezeli őket a gép, ezért a mínusz 2 milliárdtól plusz 2 milliárdig terjedő intervallumra számíthatsz. Előjel nélküli 32 bites számot (például "int unsigne"-ként definiált CRC32 számokat vagy a MySQL kulcsmezőket) nem tárolhatsz egészként, erre a célra sztringek vagy lebegőpontos számot kell használnod. A GMP olyan alternatíva, amely korlátlan hosszúságú egészeket biztosít. (A CRC-t - a Cyclical Redundency Checks - a fájlok és hálózati átvitel igazolására használják. A CRC32-t a 8. fejezetben mutatom meg.)

A GMP, vagy GNU MP egy nyílt forráskódú matematikai csomag, amely a www.swox.com/gmp/ címről tölthető le, és beillesztésével korlátlan hosszúságú egészekkel 1
lehet dolgozni a PHP-ban. A GMP odalon erről az áll, hogy „a GMP tetszőleges pontossá-

■
gú aritmetikai programeljárások ingyenes gyűjteménye, amellyel előjeles egész, racionális és lebegőpontos számokon lehet műveleteket végrehajtani. Csak a GMP-t futtató gép memó- I
nája szab határt a műveletek pontosságának".

A GMP használatához le kell töltened és fordítanod a kódot, és a megfelelő opciót kell a PHP-ban kiválasztanod. Mikor legutóbb megnéztem, a GMP Windowson még nem volt használható, így lehet, hogy további információk után kell nézned, és lehet, hogy mire ezt olvasod, a PHP alapesetként tartalmazza a GMP-t.

Hogyan használd a GMP-t? A számokat és a számokat tartalmazó sztringeket GMP-szá-mokká konvertálhatod a **gmp_init()** alábbi használatával:

```
$gmp_number = gmp_init(1234567890987654321 );
```

A legtöbb esetben a PHP a PHP-egészeket és -számokat tartalmazó sztringeket automatikusan GMP-számokra konvertálja, így a **gmp_init()** csak vészhelyzet esetére szolgál. Ugyanerre az eredményre juthatsz, ha a következő értéket használog a GMP-függvényekben:

```
$gmp_number = "1234567890987654321";
```

A következő példa két nagyon nagy szám összeadását mutatja. Próbálkozz a GMP-vel és /
olvasd a leírását a www.swox.com/gmp/ címen:

```
$gmp_number = "1234567890987654321";
print (gmp_add($gmp_number, "9999999999999999999") ) ,-
```

Lebegőpontos matematika

A legtöbb számítógép az IEEE 64 bites lebegőpontos sztenderdet használja, ami 14 számjegyre épül. Az ennél nagyobb számokhoz használd a Tetszőleges pontosságú matematikai függvényeket vagy a GMP-t.

A lebegőpontos számok nem minden decimális számot képesek pontosan megjeleníteni (8 helyett esetleg 7,9999999-et kapsz eredményül), így célszerűbb az egészeket egészként megtartani, és a lebegőpontos számokat csak akkor használni, amikor nincsen szükséged pontos decimális értékekre. Ha eredményként olyan számot kapsz, amely nem pont az, amire szükséged van, használd a **floor()** és **ceilQ** PHP-függvényeket. A **floor()** a szám egész részét adja, a **ceil()** pedig felkerekíti a számot - jóllehet egyik sem segít azon a problémán, hogy vajon hány tizedesjegytől tekinthető egy szám pontosnak, és hánytól nem. Ha százalékokkal dolgozol és egy tizedesjegyig vagy kíváncsi a válaszra, a kiinduló adataid pedig 5 tizedjegy pontosságúak, akkor az eredmény elég pontos lesz, hiszen például a 7,9999999-et a **round()**-függvény 8.0-ra kerekíti. A problémák inkább akkor jelentkeznek, amikor konkrét összehasonlításokat végzel. Például a 7,9999999 soha nem lesz pontosan egyenlő 8-cal. Ha lebegőpontos számokat hasonlítasz össze, valami hasonlót használj, mint a következő példa, ami az összehasonlítás mindkét oldalán először négy tizedesjegyre kerekíti a számokat:

```
if(round($a, 4) == round($b, 4));
```

Tetszőleges pontosságú matematika

A tetszőleges pontosságú matematika olyan matematika, ahol te választod ki a számodra szükséges pontosságot. A szoftver, jelen esetben a BCMath, valamilyen módon mindenképpen megbirkózik a nagy számokkal, legfeljebb bizonyos műveletek tovább tartanak, mint a normál matematikai függvények, hiszen a nagy számokat kisebb bitekre szedi szét, részenként számol, majd összegez. Bizonyos függvények hosszú ideig dolgoznak majd, hiszen a matematikai szoftver próbálja az óriási műveletet milliónyi kis összeadás és kivonás elvégzésére bontani.

A PHP a Phil Nelson-féle BCMath gyűjteményt használja, amelyet nemrégiben adtak ki LGPL-licenc alatt, amely lehetővé teszi, hogy a PHP normál disztribúciója tartalmazza a BCMath-ot. Azonban nem mindig volt ez így, ezért a régebbi PHP-k esetleg frissítésre szorulnak. A következő példa a BCMath **bcadd()**-függvényét mutatja be. A $a-ba és $b-ben két hosszú számot definiáltam, és a bcadd()-ban a választható pontossági paraméter nélkül adódnak össze. Az eredmény az utolsó sorban jelenik meg:

```
$a = "256.58589575987595875004982409248209 44";
$b = "111.00000000000000000000000000000011";
$c = beadd($a, $b); print($c);
```

A BCMath alapértelmezett beállítása a 0 tizedesjegy, ezért a következő példában ugyanezt a két számot a bcadd{)-függvénybe a 34 tizedes]egyű pontossági paraméterrel együtt adjuk be. Az eredmény most 34 tizedesjegy pontossággal:

```
$c = bcadd($a, $b,
34}; print( $ c ) ;
```

```
3 67.5858957 59 87 59 587 50049 82 409 2 48209 55
```

Gyors megoldások

Adatok létrehozása

A következő példákban megmutatom, hogyan lehet a különböző típusú adatokat létrehozni.

Sztring-adatok

Sztring típusú adatot könnyű létrehozni, és a sztringek különböző módon való összefű-zésével hasznos adatokat kreálhatsz. A következő kódban a $a egyszeres idézőjelet tartalmazó, de kétszeres idézőjelben lévő sztrmget fogad be, ami a szöveginputokkal gyakran előfordul. A Sb egyszeres idézőjelek között kétszeres idézőjelet tartalmazó sztringet fogad be, amit akkor használsz, ha a szöveged idézetet tartalmaz. A $c a Holt költők társaságában híressé vált háromsoros verset tartalmazza. A \n karakter sortörést szúr be: ezt használhatod a kétszeres idézőjelben levő sztringeknél, de az egyszeres idézőjel esetén nem:

```
$a = "The cat's dinner smells fishy.";
$b = 'The cat said "meow".';
$c - "The cat\nsat on\nthe mat.";
```

Egyszeres idézőjeles sztringekben egyszeres idézőjel csakis akkor lehet, ha visszaperjel előzi meg, mint a $d-ben. Mivel a visszaper különleges karakter, használatához is visszaperjel kell, ahogy a $e mutatja:

```
$d = 'Single quote \' in a single quoted
string.'; $e = 'Backslash \\ in a single
quoted string.';
```

Kétszeres idézőjeles sztringekben kétszeres idézőjel csakis akkor lehet, ha visszaperjel előzi meg, mint a $f-ben. Mivel a visszaper különleges karakter, használatához is visszaper jel kell, ahogy a $g mutatja. A kétszeres idézőjeles sztringekben lehetőség van a sztringek változókkal való helyettesítésére, így $i eredményül a "The cat likes tuna for dinner." Szöveget tartalmazza, hiszen a $h tartalma a "tuna". A kétszeres idéző'jeles sztringekben a \n-nel sortörést lehet generálni (az összes operációs rendszerben), a \r-rel pedig visszatérési karaktert (amely különböző operációs rendszerekben különböző módon használható):

```
$f = "Double quote \" in a double quoted string.";
$g = "Backslash \\ in a double quoted string.";
$h = "tuna";
$i = "The cat likes $h for dinner.";
```

Két sztringet (példánkban $j és $k) a következő módszerek bármelyikével összefűzhetsz. A Sl-ben kétszeres idézőjelen belüli helyettesítéssel kapcsolódnak össze. A $m-ben pont használatával fűztem őket össze, hiszen a pont a PHP összefűzési operátora. Az eredmény "catfish", egy hal, amelyet szeretnek a macskák:

```
$j ="cat";
$k ="fish";
$1 ="$j$k";
$m = $j . $k;
```

Az összes egy tömbben lévő sztringet összefűzheted az **implode()** használatával, amelyet az 5. fejezet adatbázisos példáinál SQL-sztringek építésére alkalmazok. A következő példában a Sn tömb tartalmazza a mondat szavait, és a $o-ban állnak össze a szavak mondattá az implode() és egy pont használatával, amely összefűzéssel adódik a szavakhoz. Az implode()-függvény szóközt szúr be az egyes szavak közé, így az eredmény a "cats drop fur everywhere." mondat:

```
$n = Array("cats", "drop", "fur",
"everywhere"); $o = implodeC ", $n) .
```

A serialize()-függvény adatok sokaságát fordítja egy olyan sztringbe, amely az adatbázisban vagy egy session-mezőben tárolható, miként azt a 16. fejezetben bemutatom. Az **unserialize()**-függvény a sztringet egyedi elemekre bontja vissza. A következő példa a $n-tömböt sztringgé konvertálja, majd a sztringet a $m-tömbbe konvertálja vissza:

```
$string = serialize($n);
$m =
unserialíze($string);
```

Egész és lebegőpontos adatok

Az egész és lebegőpontos számok automatikusan jönnek létre. A következő példába a $p egész lesz, míg a $q lebegőpontos. Ha azt akarod, hogy egy egész mindenképpen lebegőpontos legyen, megteheted, ahogy azt a $r-ben látod, ahol a **(double)** meghatározás felülírja a PHP alapértelmezett típust. A gettype()-pal az összes típust megkapod, ha print-utasí-tással megjeleníted. Az utolsó sor tartalmazza az eredményt:

```
$P = 111; $q =
11.1; $r = (double)
111;
print("Types: p: " . gettype($p) . ", q: " .
    gettype($q) . ", r: " . gettype($r) ) ;

Types: p: integer, q: double, r: double
```

A **settype()** használatával is megváltoztathatod a változók típusát. A settype() hamisat ad eredményül, ha a konverzió nem lehetséges. A következő példa azt mutatja, hogyan lehet a $p-t lebegőpontosra változtatni:

```
settype($p, "double");
```

Meg kell mondanom, hogy nem bízom a típusok közvetlen megváltozatásában, így azt javaslom, ne használd a settype()-ot. Ha meg akarod változtatni a típust, hozz létre egy olyan típusú új változót. A tesztelés során összehasonlítást tehetsz előtte és utána is.

Adatok ellenőrzése

Az adatokat összehasonlító operátorokkal és függvényekkel ellenőrizheted. Az alábbi lista azokat a tesztváltozókat tartalmazza, amelyeket az ellenőrzés során használni fogunk:

```
$a = "";
```

```
$c = 0;
$d = 22.55;
$e = falsé;
$f = null;
$g = "no new taxes";
$h = "30 May, 2002";
```

A hagyományos összehasonlító operátor, a == azt ellenőrzi, hogy a két adat ugyanolyan értékű-e, és nem működik, ha a PHP az értékeket automatikusan egyik típusról a másikra váltja (mint a példa első sorában, ahol a PHP a "0" sztringet az összehasonlítás előtt nullára konvertálja). A második sorban az új PHP4 összehasonlító operátor az összehasonlítás előtt ellenőrzi, hogy a változók ugyanolyan típusúak-e, és ha nem, hamisat ad eredményül:

```
if($b == $c)      //      true
if($b === $c)     //      falsé
```

A következő példa függvényei változó attribútumokat jelenítenek meg. Az egyes függvények eredményei // jellel kezdve megjegyzésként jelennek meg a függvény után (az 1 igazat jelöl, míg a üres hamisat):

```
print(empty($a));       II   1
print(empty($b));       II   1
print(empty($c));       II   1
print(empty($d));       II
print(empty($e));       II   1
print(empty($f));       II   1
print(empty($z));       II   1
```

and Warning: Undefined variable: z

Az empty()-függvénnyel gyorsan ellenőrizhető, hogy egy változó üres-e, de problémát jelent, hogy a 0-t tartalmazó sztringet is üresnek tekinti. Ha egy 0-t tartalmazó változó nem tekinthető üresnek, a következő függvényt használd az üresség ellenőrzésére. A függvény eredményei a függvény után jelennek meg:

```
function is_empty($field)
    {
    if(!isset($field) or !
        strlen($field)) {
        return(true);
        }
    else {
    return(falsé); } }
print(is_empty($a))
print(is_empty{$b))
print(is_empty($c))
print(is_empty($d))  print(is_empty($e))
print(is_empty($f))                      1
print(is_empty($z))
                                         1
```

A következőkben hasznosabb függvények vannak felsorolva, az eredményeiket pedig megjegyzésként láthatod. Ha egy elemet tévesen használsz, a PHP gyakran beszúrt meg-

jegyzéssel segít kijavítanod a hibát. Ha egy sztring helyett tömböt raksz a print-utasításba, a PHP a tömböt az array szóval helyettesíti:

```
prin (gettype($a));     / /     string
prin (is_array($a));    / /
prin (is_bool($e));     / /    1
prin (is_double($d));   / /    1
prin (is_long($c));     / /    1
prin (is_null($f));     / /    1
prin (is_numeric($h));  / /
prin (is_numeric($b));  / /    1
prin (is_object($d));   / /
prin (is_string($g));   / /    1
prin (isset($b));       / /    1
prin (isset($z));       / /              there is no $z
prin (strlen($g));      / /    12        defined)
```

Az isintQ- és is_integer()-függvények az is_long()-függvény aliasai, az is_float() és az is_real() pedig az is_double() aliasai. Az is_scalar()-függvény azt teszteli, hogy a változó egész, lebegőpontos, sztring vagy Boolean-változó-e:

```
print(is_scalar($a)
print(is_scalar($b)
print(is_scalar($c)
print(is_scalar($d)
print(is_scalar($e) $i
=       array("Coke", Pepsi", "Jolt[1]
print(is_scalar($i)
print(is_scalar($z)
                                    Warning: Undefined variable: z
```

Az is_numeric()-függvény a $h-ban levő 30-at nem veszi figyelembe, de a következő szakaszban bemutatott intval() egészként értelmezi a 30-at. A print_r()- és var_dump()-függvények információt jelenítenek meg a változókról, és hibakeresésre használhatók. A következő példa a $i-tömbről jelenít meg információt:

```
print_r
Array (  [0] => Coke [1] => Pepsi [2] => Jolt )
```

Figyelem: Ha a print_r() olyan tömböt jelenít meg, amely ugyanarra a tömbre vonatkozó hivatkozást tartalmaz, a print_r() print-ciklusba kerül. Ha megpróbálod a $GLOBAL-töm-böt megjeleníteni, ciklusba kerülsz, hiszen a $GLOBAL egyik eleme a $GLOBAL.

A var_dump() több információt nyújt. A következő példa ugyanazt a tömböt jeleníti meg, mint amit a print__r()-nél használtunk. Eredményként a tömb elemeinek a számát és az egyes sztringek hosszúságát adja (amivel segít a sztring végén véletlenül ott felejtett szóközök által okozott bonyodalmak elkerülésében). A var_dump() ugyanúgy ciklusba kerülhet, mint a print_r(), ha olyan tömböt dolgoz fel, amelyben ugyanarra való hivatkozás található:

```
var_dump($i);
array(3)    {    [ 0 ] = >          "Coke"    [ 1 ] = >    string(5)    "Pepsi"
string(4! strmg ( 4 )          [ 2 ] = >    _
"Jolt"     }
```

A var_dump() eredménye elég zavaros is lehet, ha a tömbön belül tömbök is vannak. A var_dump() newline-karaktert tartalmaz az eredményben, és a következő példa HTML-

tag-eket használ arra, hogy a böngésző a newline-karaktereket sortörésként értékelje. Az eredmény még összetett adatok estén is egy könnyen olvasható lista:

```
print("<pre>");
var_dump(\$ i) ;
print("</pre>");

array ( 3) { [0]=>
   string(4) "Coke"

   string(5) "Pepsi"
   [2]=>
   string(4) "Jolt"
```

Adatkonvertálás

Az intval()-függvény egészértékeket keres, és levágja a tizedespont utáni értékeket. Sztringekből is képes egészet létrehozni, ha az egész a sztring elején van. A következő példában láthatjuk, hogy az **intval()** hogyan emel ki értékeket, és mi az eredménye. Figyeld meg, hogy az **intval()** a "30 May"-ből kiemeli a 30-at, de a "May 30"-ból nem:

```
intval(22.45)                2
intval("30 May, 2002")       30
intval("May 30, 2002")       0
```

A base_convert()-függvény a számokat az egyik számrendszerből egy másik számrendszerbe váltja a 2-estől a 32-es számrendszerig eső tartományban. A 2-es alapú számrendszer a közkedvelt bináris, a 8-as az alkalmanként használt oktális, a 10-es a sztenderd decimális, a 16-os pedig a hexadecimális számrendszer. A következő példák azt mutatják meg, hogy a **base_convert()** hogyan váltja a decimális 30-at bináris számmá, a decimális 30-at hexadecimálissá, a hexadecimális 0-t binárissá és a hexadecimális 30-at binárissá. Az eredmények a jobb oszlopban láthatók. Vedd észre, hogy a hexadecimális 0 bináris eredménye egyszerű 0, de a 0 a hexadecimális 30-ban 0000-t eredményez, hiszen a base_convert{) a kezdő nullákat elhagyja. Bizonyos esetekben van értelmük a csonkolt számoknak, de problémát jelenthetnek, ha az eredményt egy nagyobb számmal fűzöd össze:

```
print("<b >" . base_convert "30" , 10, 2)) 11110 le 11 0
print("<b >" . base_convert "30" , 10, 16) 110000
print("<b >" . base_convert "30", 16, 2))
print("<b >" . base_convert "0", 16, 2));
print("<b >" . base__convert "30" , 16, 2))
```

A decbin()-függvénnyel egy 2 milliárdnál nem nagyobb decimális számot konvertálhatsz egyszerűen binárissá. A bíndec()-függvénnyel egy 31 bitesnél nem nagyobb bináris számot konvertálhatsz egyszerűen decimálissá. A dechex()-függvénnyel 2 milliárdnál kisebb decimális számot konvertálhatsz hexadecimálissá. A **hexdec()**-függvény 7FFFFFFF-nél kisebb hexadecimális számot konvertál decimálissá. A **decoct()-** és **octdec()**-függvényekkel ugyanilyen típusú konvertálást végezhetsz oktális és decimális számok között.

A **deg2rad()** függvény fokról radiánra váltja az adatokat, a **rad2deg()** pedig radiánról fokokra. Az igazat megvallva, a PHP számtalan adatkonvertáló függvénnyel rendelkezik. A 2.2 táblázat a matematikai átváltások gyors áttekintését tartalmazza.

2.2 táblázat A PHP matematikai átváltásai

$integer_or_f!oat =	abs($int_or_float)	Abszolút értékét adja
Sfioat =	acos($another_float)	A szög arkusz coszinuszát adja radiánban
Sfioat =	asin($another_float)	A szög arkusz szinuszát adja radiánban
Sfioat =	atan(Sanotherfloat)	A szög arkusz tangensét adja radiánban
Sfioat =	atan2($float_1, $float_2)	A két változó arkusz tangensét adja
Sfioat =	cos($another_float)	A radiánban meghatározott szög koszinuszát adja
Sfioat =	exp(Safloat)	Az e-t a Safloat hatványára emeli
Sfioat =	lcg_value()	Kombinált lineáris kongruencia generátor
Sfioat =	log(Safloat)	A Safloat természetes logaritmusát adja
Sfioat =	log10($afloat)	A Safloat 10-es alapú logaritmusát adja
Svalue =	max(Sarray)	A tömben levő maximális értéket adja
Svalue =	max{$va!ue1, $value2 ... Svalue n)	A paraméterlistában levő maximámális értéket adja
Svalue =	min(Sarray)	A tömben levő minimális értéket adja
Svalue =	min($value1, $value2 ... Svalue n)	A paraméterlistában levő minimális értéket adja
Sstring =	number_format($float, $decimal_places, Sdecimal_point, Sdecimalseparator)	A tizedesjel és az ezreseket elválasztó karakter használatával megformáz egy számot
Sfioat =	Pi()	A pi értékét adja 13 tizedes pontosságban
Sfloata =	pow{$floatx, Sfloaty)	A $floatx-ot a $floaty-dik hatványra emeli
Sfioat =	sin($another_float)	A radiánban meghatározott szög szinuszát adja
Sfioat =	sqrt(Safloat)	A Safloat négyzetgyökét adja
Sfioat =	tan(Sanotherfloat)	A radiánban meghatározott szög tangensét adja

Megjegyzés: Ha részletesen érdekelnek a lineáris kongruencia ***generátorok, látogass*** *el a* ***http://crypto.mat.sbg.ac.at/results/karl/server/node3.html oldalra.***

A max()- és min()-függvények tömböt is és két vagy több (korlátlan számú) paraméterbó'l álló listát is fogadhatnak. A number_format()-függvény egy, kettő vagy négy paramétert fogad el: a formázni kívánt számot, a tizedesjegyek számát, a decimális hely jelölésére használandó karaktert (amennyiben nem tizedespontot akarsz használni), és az ezres csoportosítást elválasztó karaktert (amennyiben nem vesszőt akarsz használni). Azokban az európai országokban, ahol tizedespont helyett tizedesvesszó't használnak, ezt írnák:

```
Sstring = number_formát($float, 2, ",", ".");
```

Sztringfüggvények

A sztringfüggvényekkel dolgozni a PHP-ban szórakoztató dolog, és az adatkezelésnek egy nagyon hasznos módja. A legtöbb sztringfüggvény bináris adatokat kezel, és a PHP a számokat azonnal sztringgé konvertálja, hogy függvényeknek inputot biztosítson. A 2.3 táblázat a sztringfüggvények összefoglalását tartalmazza.

2.3 táblázat PHP-sztringfüggvények

Eredmény	Függvény	
Sstring =	addcslashes{$string, $characters)	C-stílusú visszaper jeleket ad a Scharacters-ben felsorolt karakterekhez.
$string =	$addslashes($string)	Visszaper jeleket ad az idézőjelekhez, kétszeres idézőjelekhez, visszaper jelekhez és az üres karaktrekhez, így a sztringet SQL-ekkel be lehet szúrni adatbázisba.
$string =	bin2hex($string)	Az input sztring bináris adatának hexadecimális megjelenítését adja vissza.
$string =	chop($string)	Eltávolítja a szóvégén álló szóközöket, így a szóközöket, \r, \n, \t és chr(13) karaktereket.
Sstring =	chr($integer)	A szám ASCII karakterkészletben megfelelőjét adja (az ord() ellentétje).
Sstring =	chunk_split($string, Slength, $end)	Minden Slength (alapértelmezésbe 72) karakterenként beszúrja a $end karaktert (alapértelmezésben \r\n) a sztringbe. E-mailekben csatolt állományokhoz használják base64 kódolás mellett.
Sstring =	convert_cyr_string (Sstring, Sfrom, $to)	Az egyik cirill karakterkészletről a másikra konvertálja a karaktert.
Sarray =	count_chars($string, $mode)	Egy tömböt ad vissza, mely a sztring egyes karaktereinek gyakoriságát tartalmazza. ASmode opcionális és ellenőrzi, hogy mi van számolva
$int =	crc32($string)	A sztring CRC32 értékét számolja ki. (8. fejezet)

2.3 táblázat PHP-sztringfüggvények (folytatás)

$string =	crypt($string, $key)	Egy opcionális kulcs használatával DES eljárással rejtjelez egy sztringet.
	echo($string)	Egy vagy több sztringet írat ki, hasonló a print()-hez.
$array =	explode($separator, Sstring)	Tömböt hoz létre a sztringböl a $separator egyes előfordulásainál szétvágva azt.
Sstring =	get_html_translation_-table($table, $quote)	A htmlspecialchars() vagy a htmlentities() által használt fordítótáblát adja vissza. Az opcionális $quote-tal azt lehet meghatározni, hogyan kezelje az idézőjeleket.
$array =	get_meta_tags ($file_name)	A megnevezett fájl méta tag tartalmát adja tömbben vissza.
$string =	hebrev($hebrew_text, $chars)	A bementi sztringben levő héber szöveget soronként megjeleníthető szövegre konvertálja a $chars-szal.
$string =	hebrevc($hebrew_text, Schars)	A hebrev{} egy változata, amely a \n-t \n-re változtatja a HTML részére.
$string =	htmlentities($string)	A <, >, ', ", & és a nehezen megjeleníthető karaktereket HTML-elemekre konvertálja.
$string =	htmlspecialchars($string)	A<, >, ', ", és & karaktereket HTML-elemekre konvertálja.
$string =	implode($separator, Sarray)	A tömb elemeket egyesítve sztringet hoz létre. Az explode() ellentétje (lásd 3. fejezet).
$string =	join($separator, $array)	Az implodeO aliasa. Használd az implode()-ot.
$int =	levenshtein($string, Sstring)	A két sztring közötti Levenshtein-távolságot számítja ki, ami a egyezőség mértéke. Lásd még metaphone, similar_text() és soundex().
Sstring =	localeconv()	Egy tárhely setlocale()-!al beállított formázási információját adja vissza (lásd 12. fejezet).
Sstring =	ltrim{$string)	Levágja a szóközöket, \n, \r, \t, \v és \0 karaktereket a sztring elejéről.
$string =	md5()	A sztring MD5-típusú digitális kivonatát adja vissza. (Az MD5-ről a www.faqs.org/rfcs/rfc1321 .html oldalakon találsz részleteket.)
$string =	metaphone($string)	Egy sztring metaphone-értéket számolja ki, ami egy jelzés arra, hogyan hangzik a szó. Az „úgy hangzik, mint" típusú kereséseknél használható. Hasonló a soundex()-hez.

2.3 táblázat PHP-sztringfüggvények (folytatás)

Eredmény	Függvény	Cél
$string =	nl2br($string)	A sztringben az összes newline (\n) elé HTML-sortörést ( ) szúr be.
$integer =	ord(Sstring)	A karakter ASCII értékét adja vissza (a karakter karakterkészleten belüli ordinális számát adja vissza).
	parse_str($string, $array)	Változóba helyezi a sztringet, mintha az URL lenne. Ha meg van adva a második, opcionális paraméter, akkor változók helyett tömbként tárolja el az értékeket.
$boolean =	print($string)	Egy sztringet írat ki, siker esetén igazat, hiba esetén hamisat ad vissza.
$boolean =	printf($format, Svariables)	A Sformat sztringgel formázott változókat írja ki.
$string =	quoted_printable_-decode(Sstring)	Egy idézőjeles megjeleníthető sztringet 8 bites sztringre konvertálva ad vissza.
Sstring =	quotemeta($string)	A sztringben a méta karakterek \\ + * ? [^] ($ és) elé visszaper jelet rak. A htmlspecialcharacters() jobb, ha a sztring SQL-be kerül, a htmlentities() pedig akkor, ha HTML-lel használva.
$string =	rtrim($string)	Eltávolítja a szóvégén álló szóközöket, így a szóközöket, \r, \n, \t, \v, és \0 karaktereket. A chopQ aliasa.
Sarray =	sscanf($string, Sformat)	Egy tömböt ad vissza, melynek az értékei a Sformat használatával kerültek be a sztringből.
Sstring =	setlocale(Scategory, Sstring)	Beállítja a Sstring értékére a kategória által meghatározott lokális információt.
$int =	similar_text($string, Sstring)	A két sztring közötti hasonlóságot számolja ki. Hasonló a levenshtein()-hez, de több feldolgozási időt igényel.
Sstring =	soundex(Sstring)	Egy sztring soundex értékét számolja ki, ami egy jelzés arra, hogyan hangzik a szó. Az „úgy hangzik, mint" típusú kereséseknél használható. Hasonló a metaphone()-hoz.
Sstring =	sprintf($format, Svariables)	A print() egy változat, mely ahelyett, hogy közvetlenül az oldalon megjelenítené a sztringet, visszaadja azt.

2.3 táblázat PHP-sztringfüggvények (folytatás)

Eredmény	Függvény	
$int =	strcasecmp($string1, $string2)	Bináris-biztos, a kis- és nagybetűket nem megkülönböztető sztring összehasonlítást végez el. 0-t ad vissza, ha $string1 == $string2, -1-et, ha $string1 < $string2, vagy 1-et ha $string1 > $string2.
$int =	strncasecmp($string1, $string2, Slength)	A strcasecmpO változata, mely a sztringeket a $length által megadott hosszúságig hasonlítja össze.
$int =	strchr($search, $string)	Az strstr() aliasa.
$int =	strcmp($string1, $string2)	A strcasecmpO kis és nagybetűket megkülönböztető változata.
$int =	strcoll($string1, $string2)	Az strcmpO egy változata, mely a helyi beállításokkal módosított változatok összehasonlítását végzi.
$int =	strcspn($string, $chars)	A $string kezdő szakaszának a hosszát adja vissza, amelyben nincsenek a $chars-szal egyező karakterek.
$string =	strip_tags($string, $tags)	Eltávolítja a sztringből a HTML tag-eket a $tags-ben felsoroltak kivételével.
$string =	stripcslashes()	Az addcslashes()-sel hozzáadott visszaper jeleket távolítja el.
$string = stripslashesQ		Az addslashes()-sel hozzáadott visszaper jeleket távolítja el. Ha addslashes()-t használsz, majd ezt követően MySQL-ben tárolod az adatokat, a MySQL eltávolítja a visszaper jeleket és használatra kész adatokat ad vissza.
$int =	stristr($search, $string)	A strstr() kis- és nagybetűket megkülönböztető változata.
Sint =	strlen($string)	A sztring hosszát adja vissza.
$int =	strnatcasecmp($string1, $ string2)	Természetes rendezést (lásd 3. fejezet) használó, a kis- és nagybetűket nem meg-különböztető sztring-összehasonlítás, tömbök rendezésénél
$int =	strnatcmp($string1, $ string2)	A strnatcasecmpO kis- és nagybetűket megkülönböztető változata.
$string =	strncmp($string1, $string2, $length)	Astrncasecmp()kis- és nagybetűket megkülönböztető változata.

2.3 táblázat PHP-sztringfüggvények (folytatás)

Eredmény	Függvény	
$string =	str_pad($string, Slength, $pad, Smode)	A szóközökkel a Slength mértékben jobbra igazított sztringet adja vissza. Ha az opcionális $pad sztring meg van adva, akkor azt használja a szóközök helyett. Az opcionális $mode-dal választhatsz a balra vagy középre igazítás között.
$int =	strpos($string1, $string2)	A$string1-el adja vissza a $string2 pozícióját.
$string =	strrchr($string, $char)	A $string-ből az utolsó $char utáni sztringet adja vissza
$stríng =	str_repeat($string, Sinteger)	Olyan sztringet ad vissza, amely a bemeneti sztringet Sinteger alkalommal tartalmazza.
$string =	strrev(Sstring)	A sztringet fordított karakter sorban adja vissza.
$int =	strrpos{$string, $char)	A $string-ben levő utolsó $char pozícióját adja vissza.
$string =	strspn{$string, $char)	A Sstring első részét adja vissza, amely csak a $char-ban levő karaktereket tartalmazza
$int =	strstr($search, Sstring)	A $search-nek a $string-en belüli első előfordulásának pozícióját adja vissza.
Sstring =	strtok($string, $separator)	A $separator-nál elválasztja a sztringet és egyszerre egy szakaszt ad vissza. Az explodeQ használatával jobban jársz, utána ciklussal végigfuthatsz az explode{) által adott tömbön.
Sstring = strtolower(Sstring)		Kisbetűsre konvertálja a sztringet.
Sstring = strtoupper(Sstring)		Nagybetűsre konvertálja a sztringet.
Sstring =	str_replace($find, Sreplace, Sinput)	A sztringben a Sfind összes előfordulását $replace-re cseréli. A PHP 4.0.5 utáni verzióiban az input és output lehet tömb is.
Sstring =	strtr($string, $from, $to)	A sztringben a karaktereket $from-ról $to-ra fordítja.
Sstring =	substr($string, $start, Slength)	A sztring $start-tól kezdődő és a Slength hosszúságú részét adja vissza. Ha a Slength nincs meghatározva, akkor a sztring végig adja vissza azt.
$int =	substr_count($string, Schar)	A $string-ben levő Schar előfordulásának számát adja vissza.

2.3 táblázat PHP-sztringfüggvények (folytatás)

Eredmény	Függvény
$string =	substr_replace($string, Egy sztringet ad vissza, amelyben a $start-tól kezdődő szöveg a $length hosszúságban a $replace-re van cserélve. Ha a $length nincs megadva, a sztring végéig cseréli.
Sstring =	trim($string) Levágja a szóközöket, \n, \r, \t, \v és \0 karaktereket a sztring elejéről és végéről. Megegyezik a !trim(rtrim())-mel.
$string =	ucfirst(Sstring) A sztring első karakterét nagybetűssé változtatja. Különösen mondatok formázásánál hasznos.
$string =	ucwords($string) Minden szó első karakterét nagybetűssé változtatja. Fejezetcímek formázásánál hasznos (bár magyar nyelvben ez nem szokás).
Sstring =	wordwrap($string, Slength, $char) newline-okat szúr be a sztringbe úgy, hogy a szöveget a $length-nek megfelelő hosszúságú *sorokra* töri. Ha a Slength nincs megadva, akkor alapbeállításban soronként 72 karakter. Ha a Schar meg van határozva, akkor az van a newline helyett.

randQ, srandQ és microtime()

A rand()-függvény véletlen számot állít elő. Az mt_rand()-függvény tökéletesebb véletlen számot biztosít, ahogy ez a **www.math.keio.ac.jp/~matumoto/emt.html** oldalon le van írva. Az **mt_rand()** gyorsabb is, bár a sebesség ritkán szempont, hacsak nem oldalanként többször használod a **rand()**-függvény.

Mind a rand()-hoz, mint az **mt_rand()-hoz** szükség van a véletlen szám generáció kezdő értékére. A **rand()** használatánál a szkriptedben első **rand()** előtt egyszer le kell futtatnod a srand()-ot, de ugyanazon szkripten belül ezt csak egyszer kell megtenned. Az **mt_rand()** használatakor az **mt_srand()-ot** ugyanazzal az inputtal egyszer kell lefuttatni:

```
Ssrandl(double)    microtime{)    *    1000000);
```

A **rand()** és az **mt_rand()** is elfogad intervallum-paraméter. Ha a használható intervallum maximumát akarod megtudni, getrandmax()-szal vagy **mt_getrandmax()-szal** teheted meg. Egy NT-munkaállomáson PHP 4.0.5-öt használva a getrandmax()-ra 32767-et kaptam eredményül.

A következő kódot használd, ha 1 és 99 között akarsz egy véletlen számot kapni. Ha üresen hagyod a második paramétert, a maximum értéket, a **rand()** a **getrandmaxQ** által adott maximumot használja. Ha az első paramétert hagyod üresen, a **rand()** a nullát használja. A legtöbb esetben adott intervallumot fogsz használni, így mindkét paramétert meghatározod:

```
$integer_number    =    rand(1,    99);
```

Ha 1,00 és 2,00 közötti számot akarsz, a 100 és 200 közti intervallumot válaszd, és az eredményt oszd el 100-zal.

Nézzük meg a microtime()-ot, az ideális szkript időzítőfüggvényt. A microtime() két, szóközzel elválasztott számot ad eredményül, például ezt: 0.62186100 988888118. A második szám Unix-idő másodpercekben, az első pedig a másodperc törtrésze. A következő kóddal tudod a microtime() eredményét használható dátum- és időformátumra konvertálni. Az explodeQ-függvény a szóköz alapján elválasztja a két mezőt, a **date()** a Unix-időt dátum és idő formátumra alakítja, a **sustr()** eltávolítja a kezdő nullát az idő mikrorészéről, a második dateQ pedig beállítja a napszakot (am vagy pm):

```
$mstart = explode{" ", $m);
print("The date and exact time is:<br>"
    . dateC'l F j, Y h:i:s", $mstart[l])
    . substr(Smstart[0], 1)
    . date(" a", $mstart[1]));

The date and exact time is:
Thursday May 3, 2001 09:08:38.62186100 pm
```

A microtime()-ot egy művelet elején és végén lefuttatva lemérheted a művelet idejét, de ne felejts el két dolgot. Az idő megjelenítése időt vesz igénybe, ezért a művelet kezdetén mentsd a **microtime()** értékét egy változóba. Csak azután jelenítsd meg ezt, vagy csak azután végezz vele műveleteket, hogy a művelet végén is elmentetted a **microtimeQ** értékét. Vedd azt is figyelembe, hogy a **microtimeQ** eredményeképpen kapott számok meghaladják a PHP-ban alapértelmezett egészintervallumot, így a számoláshoz vagy a BCMath vagy a GMP függvényeit használd.

A következő kód a **$a microtime()** értékének $b microtimeQ értékéből BCMath-tal elvégzett kivonását mutatja meg. Az explodeQ- és substrQ függvények ugyanazok, mint az előbb. A tömb részeit aztán összefűzzük, hogy a **bcsubQ** részére sztring típusú inputot biztosítsunk, és a végén megjelenik az eredmény:

```
$a_array = explode(" ", $a); $b_array
= explode(" ", $b); $a_array[0] =
substr($a_array[0], 1); $b_array[0] =
substr($b_array[0], 1); $a„string =
$a_array[l] . $a_array[0]; $b_string =
$b_array[l] . $b_array[0];
print(bcsub($b_string, $a_string));

0.00325400
```

Dátum és idő kiszámítása

A PHP-ban a dátum- és időfüggvények széles választéka érhető el, amiket a 2.4 táblázatban foglaltam össze. A dátummal és idővel kapcsolatos számításokat az ezekből a függvényekből kapott eredményekkel, illetve ezen eredmények alakításával lehet elvégezni.

2.4 táblázat A PHP dátum- és idő-függvényei

Függvény	
checkdate()	Ellenőrzi, hogy a Gergely-naptár szerinti dátum helyes-e
date()	A helyi dátumot és időt formázva adja vissza
getdate()	A helyi dátumot és időt tömbként adja vissza
gettimeofday()	Az aktuális időt tömbként adja vissza
gmclateO	Ugyanaz, mint a date(), de a helyi idő helyett a GMT-t használja
gmmktime()	A dátumot Unix-időjelzésre konvertálja
gmstrftime()	A GMT dátumot és időt formázza
localtime()	A helyi időt tömb elemeiként adja
microtime()	A Unix-időjelzést a másodperc törtrészével adja vissza
mktime()	A Unix-időjelzést adja vissza egy dátumból
strftime()	Ugyanaz, mint a gmstrftime(), csak helyi időt használ
time()	Unix-időjeSzést ad vissza
strtotime()	A szöveges dátumod/időt Unix-időjelzésre konvertálja

A Unix időjelzés nem más, mint az 1970. január l~je óta eltelt másodpercek száma, ez 2037-ig fog futni, így nem nehéz megjósolni a Unix Y2.037K problémát. (Ha elfejtetted volna, bizonyos számítógépeknek olyan dátumrendszerük volt, amely nem tudta a 2000-es évet kezelni, és ezt a problémát hívták a Y2K problémának. A Unixnál 2037 előtt hasonló változtatásokat kell majd végrehajtani.) A Unix időjelzéssel a másodpercek összeadásával és kivonásával végezhetsz számításokat.

Tegyük fel, hogy 20 órát akarsz a jelenlegi időhöz hozzáadni. A 20 óra 1200 perc vagy 72 000 másodperc, így a következő egyszerű kóddal add ezt a számot a **time()**-függvényhez. Az eredményt, mint minden dátumot és időt, a **date()**-függvénnyel alakíthatod:

```
$new_time = time() + 72000;
```

Mi a teendő, ha 3 napot, 4 órát és 22 percet akarsz hozzáadni a jelenlegi időhöz? A **strtotimeQ**-függvény a szöveges dátum- és időformátumokat Unix időjelzésre konvertálja, amelyet a time()-hoz hozzáadhatsz vagy kivonhatsz. Ezt a függvényt bármilyen, a Unix időjelzést használó dátum/idő kalkulációhoz használhatod. A kód következő sora a **$second** változónak olyan értéket állít be, amelyet hozzá lehet a **time()**-függvényből vagy egy adatbázisból nyert értékhez adni. A **strtotime()**-függvénnyel is lehet számításokat végezni. A kód második sorában a **strtotime()** 3 napot, 4 órát és 22 percet ad a "now"-hoz (a mostani időhöz), ami a **strtotime()-ban** a **time()** megfelelője:

```
$seconds = strtotime("+3 days 4 hours 22 minutes");
$seconds = strtotime("now +3 days 4 hours 22
minutes"};
```

A strtotime-függvény jelenleg az amerikai hh/nn/éééé dátumformátumban (pl. 03/04/2002) használja az adatokat, és a következő példa szerint képes a dátumból számításokat végezni. A második sor egy a Unix időjelzést date()-tel formázó és az eredményt **print-tel** megjelenítő utasítást tartalmaz, míg az utolsó sorban az eredmény látható:

```
íseconds = strtotime{"03/04/2002 +3 days 4 hours 22
minutes"); print(date("1 F j, Y H:i:s", $seconds));
Thursday March 7, 2002 04:22:00
```

Húsvét

Ha az 1970 és 2037 közé eső évekre szeretnéd a húsvét dátumát megtudni, az easter_date()-függvény a tökéletes megoldás. A függvény a Unix időjelzést eredményezi, melyet például a dateQ dátumformázó függvénybe illeszthetsz be. Az eredményt megjelenítheted, miként azt következő példában látod. Az **easter_date()** függvény az adott dátum éjféljének idejét adja eredményül, és csak a Unix időjelzés által lefedett évekre használhatót

```
print(date ("F Y",
d, April 20, easter_date(2003)
2003
```

Az ettől eltérő évek esetén az easter_days() függvénnyel kapod meg a március 21. és húsvét közti napok számát. Ezután a megfelelő dátumfüggvény használatával kiszámíthatod a kívánt dátumot. A következő példában az **easter_days()** kiszámolja a március 21-től húsvétig hátralévő napok számát, a **gregoriantojd()** az ugyanezen év március 21-ének Julianus-napszámát adja meg, a következő sor összeadja ezeket, és az eredményt a **jdtogregorian()**-ba adja be, hogy megkapjuk a húsvét Gergely-naptár szerinti dátumát, ami 3/28/1703. Az eredményül nyert dátumformátumban nehéz megkülönböztetni a 4/1/2002-t és az 1/4/2002-t, amelyek közül az első az amerikai formátum az áprilisra, a második pedig az európai a januárra. Az ötödik sor összetevőkre bontja a Gergely-naptár dátumát. A hatodik sor egy tömböt a hónapok neveivel tölt fel úgy, hogy az első elemet üresen hagyja, így a hónapok neve illeszkedik a hónapok tömbbeli sorszámához. A kilencedik sorban a hónap számát a hónap nevére cseréljük a $m-tömbben. Az utolsó sor pedig a mindenhol érthető March 28, 1703-at jeleníti meg:

```
$e = easter_days(1703);
$j = gregoriantojd(3, 21, 1703);
$g = jdtogregorian($e + $j);
print("<br>" . $g);
list($m, $d, $y) = explode("/", $g);
$months = array("", "January", "February", "March", "April",
"May",
    "June", "July", "August", "September", "October", "November",
    "December"); $m = $months[$m]; print ("<br>
. $m . " " . $d .- ", " . $y);
```

3. fejezet

Tömbök

Áttekintés

A PHP-tömböket egyszerű használni, nagy rugalmasságot kínálnak, de néhány csapdát is rejtenek magukban. A PHP-tömbök úgy használhatók, mintha memóriában tárolt mini adatbázisok lennének, hiszen függvények széles skálája áll rendelkezésre a tömbök rendezéséhez és feldolgozásához. A szöveges dokumentumokban, nyomtatható fájlokban, XML-fájlokban és adatbázisokban tárolt adatok betölthetők tömbbe, és aztán feldolgozhatok anélkül, hogy az eredeti fájlba belenyúlnánk. Megmutatom az egyszerű és bonyolultabb tömbök építését, a listák feldolgozásának és rendezésének minden módját.

Néha nagyobb teljesítményt érhetsz el, ha a komplex SQL helyett egy egyszerű kiválasztást és egy kis memórián belüli feldolgozást végzel. Ha még nem használtál SQL-t, akkor az 5. fejezetben találsz példát a MySQL és a PostgreSQL használatára. Coriolos kiadta az *SQL Server 2000 Black Book-ot* az SQL-szervert használók számára, és ha még többet akarsz tudni az SQL-ről, akkor ott van a Groff és Weinburg: *SQL: The Complete Reference* (McGraw-Hill). Az SQL-ben rendkívül összetett műveleteket lehet elvégezni. Nagyon rég, mikor még fiatal, nó'tlen és bolond voltam, az IBM-nél dolgozott egy fiatal, független, gyönyörű, nőnemű adatbázis-guru, akinek küldtem egy romantikus verset, amit teljesen SQL-ben írtam. Az SQL működött, de szokatlan adatbázis-szerkezetet kívánt.

Most, hogy már kinőttem a fiatalságból és végeztem a nőtlenséggel, sokkal egyszerűbb SQL-t használok, és hagyom, hogy a PHP formázza az adatokat. Az SQL-ben minimális mennyiségű adatot lehet és kell felhasználni, hogy csökkenjen a lemez IO. Az adatokat sorrendbe is rendezheted. Ha csak egy sorrend szerint kellenek az adatok, index segítségével az adott sorrendben mentheted azokat az adatbázisba. Ha egy oldalon az adatok ismétlődnek és csak a sorrendjük változik, akkor lehetőség van az adatok egy tömbbe való beolvasására és újrarendezésére a PHP tömbrendező-függvényeinek használatával.

Használhatod az SQL-t az adatok átalakítására, és számításokat végezhetsz a különböző függvények segítségével, de a rendelkezésre álló függvények skálája adatbázisonként változó. Szintúgy, ha két adatbázis ugyanazt a függvényt használja is, lehet, hogy más a nevük és a paraméterek is más sorrendben vannak. Ez is magyarázza, miért kell az adatbázisból minimális SQL-lel kinyerni az adatokat és a PHP-val átalakítani azokat.

Egyszerű tömbök

Kezdjük az egydimenziós tömbökkel. Ugyanúgy működnek, mint a bevásárlólista, amit vásárláskor írsz, és a tömbfüggvényeket könnyen lehet alkalmazni az egydimenziós tömbökre. A „Többdimenziós tömbök" rész foglalkozik a többdimenziós tömbökkel; a többdimenziós tömbökre gondolhatunk úgyis, mint tömbök tömbje, és gyakran dolgozhatunk többdimenziós tömbökkel, úgy mintha beágyazott egydimenziós tömbök sorozatai lennének.

A világ legegyszerűbb tömbje

Kezdetnek itt van egy egyszerű tömb:

```
$a = "abc";
```

A PHP-ban a sztring-karakterek tömbje, amelyeket így lehet elérni:

```
print($a[O] . " " . $a[l] . " " . $a[2]};
```

A kapott sztring ebben az esetben így néz ki:

a b c

A tömb elemeit úgy lehet elérni, hogy az elem számát szögletes zárójelbe írjuk az elem neve után.

A tömböket 0-tól számozzuk

Más programnyelvekben a tömb elemeit 1-gyel kezdődően számozzák, vagy te választhatod meg a kezdőértéket. A PHP-ban alapértelmezésben 0-tól kezdődik a számozás, de mivel te is megválaszthatod a kezdő értéket, így bárhol kezdődhet. Létrehozhatsz új tömböt vagy új elemet egy már meglévő tömbben, mint az alábbi példa is mutatja:

```
$a[]    =    "fruit";
```

Ezután a tömböt így érheted el:

```
print($a[0]);
```

Ez az **eredménye:**

```
fruit
```

Létrehozhatsz tömböt 1-gyel kezdődően (vagy bármely más számmal kezdve) úgy, hogy beilleszted a kezdő értéket az alábbihoz hasonlóan:

```
$b[l]    =    "pear";
```

A következő elemet úgy is létrehozhatod, hogy a zárójeleket üresen hagyod, lásd lent. Az apple-érték a $b[2]- ben lesz:

```
$b[]    =    "apple";
```

A PHP csak azokat az adatokat menti el, amelyeket te hozol létre. Például ha egy gyorsétteremlánc a fióküzleteit 1-től kezdve számozza és sok üzletet nyit, majd be kell zárnia üzletei nagy részét, akkor a üzletlánc listája így nézne ki:

```
$branch[25]   =   "Boston";
$branch[42]   =   "Sydney";
$branch[98]   =   "London";
$branch[145] = "Wahroonga";
```

A nem létező elemekre nem pazarol helyet. Persze ez azt is jelenti, hogy nem tudsz egy egyszerű ciklussal végiglépegetni az elemeken 0-tól a maximális elemig, más körmönfontabb megoldást kell alkalmaznod, mint például a **whileQ**-ciklus, amit a „Tömbmutató" fejezetben ismertetek.

A tömb elemeit bárminek elnevezhetjük

A tömb elemeit számozás helyett el is nevezheted, így *asszociatív tömböt* hozhatsz létre. A gyorsétterem példa alapján a fióküzletek személyzetének számlálása így nézhet ki:

```
$branch_staff["Boston"] = 22;
$branch_staff["Sydney"] = 45;
$branch_staff["London"] = 18;
$branch_staff["Wahroonga"] = 72;
```

A kétféle tömböt vegyítheted is:

```
$silly_array[25]    =    "Boston";
$silly_array[42]    =    "Sydney";
$silly_array[98]    =    "London";
$silly_array[145]  =  "Wahroonga";
$silly_array["Boston"]    =    22;
$silly_array["Sydney"]    =    45;
$silly_array["London" ] = 18; $silly
array["Wahroonga"]    =    72;
```

Amikor egy adatbázisfüggvénnyel, mint például a **mysql_fetch_row(),** sima tömbhöz jutsz, akkor minden mezőben egy számozott elem jelzi a mező táblában elfoglalt helyét vagy az SQL kiválasztó utasítását. Ha a mysql_fetch_row()-ról a mysql_fetch_array()-re váltasz át, akkor asszociatív tömböt kapsz, amiben minden mező számozott adatként és a mező nevével létrehozott elemként is megjelenik.

Ha van egy „válaszd ki az országot, várost a városokból"-t tartalmazó SQL-utasításod, és az első elem Sydney, akkor az **mysql_fetch_row()** létrehozza azt a tömböt, amit az alábbiak alapján te is létrehoznál:

```
$array[0]    =    "Australia";
?array[l]    =    "Sydney";
```

Ugyanezzel az SQL-lel a **mysql_fetch_array()** a következő tömböt hozná létre. A számozott adatok különállóak a névvel ellátott adatoktól, így feldolgozhatóak és megváltoztathatók a névvel ellátott adatok megváltoztatása nélkül. Ha éppenséggel olyan SQL-t írsz, amely kettőzött mezőneveket eredményez, csak a mező legutolsó előfordulása lesz egy névvel ellátott tömbadat, minden más előfordulása számozott adat lesz:

```
$array[0]    =    "Australia";
$array["country"]    =
"Australia"; $array[l]    =
"Sydney"; $array["city"]    =
"Sydney";
```

Valójában minden PHP-tömb asszociatív tömb. A számozott adatok is asszociatív adatokként működnek. Az asszociatív tömbök annyira népszerűek és olyan könnyű velük dolgozni, hogy sok PHP-függvény egyre inkább az asszociatív tömböt tekinti a tömbök általános formájának. Például a **mysql_fetch_array()** kapott egy opcionális második paramétert, hogy neves adatokat hozzon létre számozott adatok nélkül, és ezt az új paramétert oly sokszor használták, hogy valaki hozzáadta a **mysql_fetch_assoc()-ot,** ami a második paraméter begépelése nélkül hoz létre neves adatokat.

Tömblétrehozó függvények

A array() listából hoz létre normál vagy asszociatív tömböt, az explodeQ határolt sztringből hoz létre tömböt. A tömblétrehozó függvények leírása a „Tömbök létrehozása[1] és a „Gyors megoldások" részekben található.

Lehetséges problémák

Én csak két problémát találtam a PHP-tömbökkel kapcsolatban: rendezés és a kis- és nagybetű megkülönböztetése. Az olyan rendezési problémák, mint például az adatok rendezési kulcsok nélküli rendezése, elkerülhetők úgy, hogy a rendezőfüggvény kiválasztása előtt példákon keresztül az összes rendezőfüggvényt megismered. Ezek leírása megtalálható ezen fejezet „Gyors megoldások" részében.

A kis- és nagybetű megkülönböztetésének problémája akkor jelentkezik, ha egy kis- és nagybetűt megkülönböztető nyelvből a PHP-ra váltasz. Néhány nyelv megkülönbözteti a kis-/nagybetűket, és lehetőséget ad arra, hogy szükség esetén kikapcsolhasd az érzékenységet; más nyelvek mindig megkülönböztetik a kis- és nagybetűket. A PHP egy érdekes keveréket alkalmaz, a függvények nem különböztetik meg a kis- és nagybetűt, de a változók nevei igen. A tömbökben az elemkulcsok megkülönböztetik a kis- és nagybetűket. Az alábbi tömb elemei mind különböző adatok, mert máshogy vannak írva.

```
$branch_staff["Boston"] = 22;
$branch_staff["boston"] = 45;
$branch_staff["bosTon"] = 18;
```

Ne higgy nekem, próbáld ki magad. Megnézheted a tömbök kis- és nagybetű-érzékenységét azáltal, hogy az alábbi kód használatával kinyomtatod a tömböt. A gyakorlat megtanít a kis- és nagybetű megkülönböztetésére, és egy könnyű módja a tömbök felfedezésének:

```
reset ($branch__staf f) ;
while (üst ($k, $v) = each($branch_staff))

   print("<br>key: " . $k . ", value: " . $v) ;
```

A kód a tömb elejére viszi a tömbmutatót (magyarázat a „Tömbmutató" részben), majd az each() végiglépked a tömb elemein, visszaadva az összes elemet, a **list()** pedig változókba menti az elem értékét és kulcsát a print()-utasításhoz. A következő lesz az eredmény:

```
key  Boston,  value:  22
key  boston,  value :  45
key  bosTon,  value:  1
```

Ha szükség van kis- és nagybetűket tartalmazó szavak, nevek tárolására, a PHP engedi ezt. Ha a kis-/nagybetű megkülönböztetése nem fontos és ez valószínűleg kettőződéshez vezet, azt javaslom, hogy az összes kulcsot alakítsuk át kisbetűssé a **strtolowerQ** használatával, vagy dokumentáljuk a kis-/nagybetű definiálását közvetlenül a tömb definiálása mellett. Például ha városneveket használunk és nem kerülnek megjelenítésre, akkor az összest kisbetűssé átalakítanám, ha megjelenítésre kerülnek, akkor dokumentálnám a nagy kezdőbetű fogalmát.

Többdimenziós tömbök

A PHP-tömbök bármennyi dimenziót elfogadnak, és az egyes dimenziók bármennyi elemet tartalmazhatnak. Valójában az a helyzet, hogy előbb futsz ki a memóriából, mint elérnéd a PHP korlátjait. A PHP konfigurációs fájlja, a php.ini egy beállítást tartalmaz, amit az ilyen hatalmas tömbök használatához növelned kell. Változtasd meg a **memory_limit-et** az alapértelmezett 8 MB-ról a legnagyobb tömböd méretének néhányszorosára.

Többdimenziós tömbök létrehozása és azokra hivatkozás történhet pusztán a szögletes zárójelek közé zárt extra referenciák hozzáadásával, amint azt a következő példa is mutatja. A $travel_comment-tömböt ország, állam, város indexálja. Létrehozok egy adatot Carmel, Calinforniának:

```
$travel_comment["OSA"]["California"]["Carmel"]     =
    "Scenic    town";
```

Akárhány dimenziót létrehozhatsz, amennyi csak kell az adatok szervezéséhez, így a példa az utazási információkat ország, állam, város szerint strukturálja. Hozzáadhatsz még kerületet, utcát, házszámot, éttermet és baristát, ha Carmel összes épületében, minden egyes éttermében a megfelelő kávéfőzőhöz akarod az embereket elvezetni, bár kétlem, hogy ez segítene ugyanolyan finom kávét találniuk mint, San Franciscóban vagy Sydneyben. (Ha nem tudod, mi az a barista, menj a **www.scaa.org/barista/** oldalra, és olvasd el az eszpresszógépek működtetését hitelesítő folyamat hivatalos leírását.)

Bármi lehet tömbelem, akár másik tömb is, így aszimmetrikus szerkezetű tömböt is építhetsz. Például létrehozhatsz egy városokból álló tömböt, ahol a nagyvárosok körzeteik alapján sorolódnak be, a kisebb városoknak csak egy adatuk van. Kilistázhatod a népességi adatokat körzet szerint, vagy város szerint, ha a városban nincsenek körzetek. A következő példa néhány fiktív adattal tölti meg a Sydney, Ausztrália tömböt:

```
$Sydney["Chatswood"] = 51,000;
$Sydney["Parramatta"] = 145,000;
$Sydney["Wahroonga"] = 12,000;
```

Adj hozzá körzet szerint osztályozott adatokat San Franciscóról:

```
$San_Francisco["Castro"] = 55,000;
$San_Francisco["Chinatown "] = 50,000;
```

Hogy **$Sydney** és **$San_Francisco** bekerüljenek egy népesség város szerinti tömbbe, a következőket kell beírnod:

```
$city["Dayuanjiadun"] = 5,000;
$city["Islip"] = 200;
$city["San    Francisco"]    =    $San_Francisco;
$city["Sydney"]    =    $Sydney;
$city["Ulaanbatar"]    =    38,000
```

Amikor hozzányúlsz a $city-tömbhöz, a feldolgozás egy kicsit trükkös, mert neked kell megnézned, hogy az értékek nem tömbök-e. Ha össze kell adnod az összes város népessé-

gét, rekurzív függvényt használj, mely összeadja a legmagasabb szintű számokat, és meghívja magát, ha az érték tömb. További információért nézd meg a „Ciklusok többdimenziós tömbökben" az „Gyors megoldások" részt.

A tömbmutató

A *tömbmutató* csak néhány nyelvben áll rendelkezésre, és az általam ismert 30 nyelv közül a PHP-ban található a legjobb megvalósítása. Segít meghatározni, hogy hol vagy, amikor egy tömbben mozogsz. Minden tömbnek megvan a saját külön mutatója, így teljesen függetlenül dolgozhatsz egyszerre több tömbben. Több függvény is támogatja a tömbmutató használatát, hogy segítse a tömbben való mozgásodat.

Amikor egy tömböt az adatokat hozzáadásával hozol létre, a mutató az utolsó hozzáadott adatra mutat, így a tömb végére további adatokat szúrhatsz be. Amikor a tömb készen van, vissza akarsz menni a tömb elejére, hogy átnézd.

A reset() a tömb elejére állítja vissza a mutatót. Ezt a parancsot használd, amikor egy ismeretlen állapotú tömbön mész végig. A tömb rendezése után a mutató visszaáll a tömb elejére, és egy tömbről készített másolatban a mutató annak az elején van. Például, ha $y egy tömb, és azt mondod $x = $y, akkor $x mutatója a tömb elejére fog mutatni.

Amikor tömböket adsz át függvényeknek, ez alapértelmezésben értékként történik, ami annyit tesz, hogy a függvény kap egy másolatot a tömbről, és a másolat tömbmutatója a tömb elején áll. Ha viszont a tömböt hivatkozással adod át a függvénynek, akkor a tömb az eredeti mutatót tartja meg, ismeretlen pozícióban.

Megjegyzés: Egy változó az & előtaggal megjelölve is átadható egy függvénynek. Erről bővebben a 10. fejezetben olvashatsz.

A mutató aktuális helyzetét a current()-függvénnyel tudhatod meg, és a next()-, prev()- és end()-függvényekkel mozgathatod azt.

Az each()-függvény megadja az aktuális adat kulcsát és értékét, és aztán átviszi a mutatót a következő adatra. Az each()-függvénnyel egy egyszerű while-ciklussal végig lehet a tömbön lépkedni. A while()-ciklus használata előtt győződj meg arról, hogy a mutató a tömb első adatán van, majd lépkedj végig az each()-függvénnyel. Az each()-függvény visszaadja az adat kulcsát és értékét, majd továbblépteti a mutatót a következő adatra, mint itt:

```
reset($array);
while(list($k,    $v)    =    each($array))
```

Ebben a példában a list()-függvény minden egyes értéket külön változóban ment el, így megkönnyíti az eredmények feldolgozását. Ha a tömbnek többszörös dimenziója van, akkor a $v változó egy tömb lesz, ami tartalmazza a $array részhalmazt, és a $v tömbmutatója a tömb elején lesz.

Tömbök rendezése

A PHP-tömböket függvényekkel tetszésed szerint rendezheted, amelyekk az alábbi elvek szerint rendeznek:

- érték szerint,
- kulcs szerint,
- abc-sorrendbe,
- számos „természetes" és speciális rendezés.

Megjegyzés: A természetes kifejezés azon rendezési elvekre utal, amelyek azt próbálják utánozni, ahogyan az emberek rendezik az egyes dolgokat olyan esetekben, amikor a közvetlen abc-sorrend nem a leglogikusabb eredményt adja. Egyben magába foglalja például azt a speciális eljárást, amikor a nevekhez illesztett számok alapján történik a rendezés. A sortQ, rsortQ, asortQ, arsortQ, ksortQ és natsortQ eljárásokra részletes példák találhatóak a „Gyors megoldások" részben.

Arra azonban ügyelni kell, hogy a függvények hogyan kezelik a kis- és nagybetűket. Az alaprendezések minden egyes karaktert különálló bináris bájtként kezelnek, amik bármit tartalmazhatnak; így megkülönböztetik a kis- és nagybetűket. A **natsort()-** és a **natcasesort()**-függvények másképp működnek, mert további kódot tartalmaznak, ami az adatokat rendezés előtt dolgozza fel. A **natcase()** extra kódja segít feldolgozni a néven belüli számokat, a **natcasesort()** a **natsort()-hoz** képest annyiban más, hogy nem különbözteti meg a kis- és nagybetűket. A legtöbb rendezőfüggvény a következőképpen rendez. Ez a lista:

a
A

b
B

c
C

rendezés után így néz ki:

A

B

C

a
b
c

Néhány tömbrendező függvény tömbbe rendezi az értékeket (elhagyva a hozzá tartozó kulcsokat és visszaállítva a numerikus kulcsok alapértelmezését), míg más függvények az értékeket a kulcs alapján rendezik, megtartva a tömbben a hozzátartozó kulcsokat. Ez számtalan lehetőséget nyújt az adatok tárolására és rendezésére, majdnem úgy, mint egy relációs adatbázisban. Tegyük fel, hogy szeretnél egy listát a barátaidról és rájuk vonatkozó információkról, és mindezt szeretnéd többféleképpen online megjeleníteni. Hogy ezt megtegyed, először is készítened kell egy listát a barátaidról (csak néhányat használj próbaképpen, de képzeld el azt, amikor listába szeretnéd szedni azt az 500 embert, akiket szüleid

hívtak meg az esküvődre, és azt 200-at, akiket te szeretnél meghívni):

```
$friend[]  =  "Cate
Blanchett"; $fríend[]  =
"Nicole  Kidman"; $friend[]  =
"Kate  Beckinsale";
```

Most add hozzá a telefonszámaikat. Az eredeti tömb nem mond sokat, de a barátok nevén alapuló asszociatív tömb már többet mond:

```
$friend["Cate  Blanchett"]  =  "555  111
1111"; $friend["Nicole  Kidman"]  =  "555  222
2222"; $friend["Kate  Beckinsale"]  =  "555
333  3333";
```

Most hozzá akarod adni a születésnapjukat, így kétdimenziós tömböt hozol létre:

```
$friend["Cate  Blanchett"]  =  array("number"  =>"555  111  1111",
    "birth"  =>  "1969-05-14");  $friend["Nicole  Kidman"]  =
array("number"  =>"555  222  2222",
    "birth"  =>  "1967-06-20");  $friend["Kate  Beckinsale"]  =
array("number"  =>"555  333  3333",
    "birth"  =>  "1973-07-26");
```

Ezután úgy döntesz, hogy a születésnapjuk alapján rendezed őket, hogy a megfelelő napon felhívhasd őket, és telefonkörzet alapján listázod (így csúcsidőn kívül tudod őket hívni). Miért állnál meg egy tömbnél? Betöltheted a tömböt többszörös tömbként, beprogramozhatod, hogy ossza az adatokat többszörös tömbökbe, vagy másolatokat készíthetsz az adatokról rendezés előtt. A példabeli kétdimenziós listát az alábbiak szerint lehet kettéosztani:

```
while (üst ($k,  $v)  =  each <$f riend) )
    {
$birthday[$v["birth"] ]  =  $k;
Snumber[$v["number"]]  =  $k;
```

Most van egy listád, amit születésnap szerint rendezhetsz, és van egy másik listád, amit telefonszám szerint. Létrehozhatsz kereszthivatkozásokat, hisz tudod, hogy a nevek teljesen egyezni fognak. A következő példában tételezzük fel, hogy tudod a telefonszámot, de nem tudod se a nevet, se a születésnapot (agglegény voltam egykor, és sokat gondolkodtam, mit tegyek a név nélküli telefonszámokkal). Ez a példa így fest:

```
$b  =  array_search($number["555  333  3333"],  $birthday);
```

A Snumber visszaad egy értéket, ami történetesen egy név. Az **array_search()** megkeresi a másik tömbben ezt az értéket, és visszaadja a kulcsot, ami a születésnap.

Ha egy listát sokféleképpen dolgozol fel egy oldalon, vedd figyelembe, mely mezők lesznek kulcsok a tömbökben, és azt is, hátha egyszerűbb lenne többszörös tömböt használni az egyes értékekhez. Vedd számításba, hogy hogyan akarsz a tömbökön végigmenni.

Push, Pop, Pad és Mérge

Sok új PHP-függvénnyel elég furcsa dolgokat lehet művelni a tömbökkel, illetve a tömbökön. Most következzen a legújabb függvények rövid összefoglalója.

Push és Pop

Az array_push() és az array_pop() parancsokat azoknak találták ki, akik a verem alapú kódot kedvelik. A tömböt használhatod memóriaveremként, ami azt jelenti, hogy az array_push() paranccssal berakhatsz dolgokat a verembe, és az array_pop()-pal kiveheted őket onnan. Felhasználási módként preferencia alapú rendszerek juthatnak eszünkbe, ahol az ember választ mondjuk egy úti célt és megadja az igényeit is. Ezután a kód kitörli azokat a lehető'segeket, amelyek nem felelnek meg a választási kritériumoknak. A kód például az összes Port Douglas-i repülőjáratot kiválasztja az ár alapján, majd a legolcsóbbtól kezdve kitörli azokat, amelyek túl sokszor állnak meg, majd azokat, amelyeken nem szolgálnak fel vegetáriánus menüt. A következő kódban a $a-tömb az adatok egyenkénti hozzáadásával jött létre, míg a $b-tömbhöz az array_push() paranccsal adtuk hozzá az adatokat, így ehhez kevesebb kódsor kellett. Mind a $a-, mind a $b-tömbnek ugyanaz a tartalma:

```
$a[]    =    "Báli";
$a[]    =   "Fiji";
$a[]              =
"Hawaii";

$b[]    =
"Báli";              'Fiji",
array_push <$b,    "Hawaii");
```

A következő példa azt mutatja meg, hogy kell az utolsó adatot visszaadatni, majd törölni hagyományos tömbparancsokkal és hogyan az array_pop() paranccsal. A $x-et egyenlővé tesszük a $a-tömb utolsó adatával az end()-függvény segítségével, és a key() visszaadja az utolsó adatot, majd az unsetQ kitörli az adatot a $a-tömbből. Az array_pop() ugyanezt az eredményt eléri egy kódsorral. $y végül meg fog egyezni $x értékével:

```
$x    =    end ($a);
unset($a[key($a) ] ) ;
$y    =    array_pop($b);
```

Nem kell először az array_push()-t majd utána az array_pop()-ot használni. Tömböt bárhogyan építhetsz, majd push vagy pop, aztán rendezheted, majd push vagy pop addig, amíg pontosan azt a tömböt nem kapod, amit szeretnél.

Pad

Tegyük fel, hogy van egy top 10-es listád valami fontos dologról, de csak kilenc adatod van, vagy csak hét vagy négy. Hogy garantálod, hogy 10 adatod legyen? Az array_pad()-függ-vény pont ezt teszi. Megadod az array_pad()-függvénynek a tömb nevét, az adatok számát, és az array_pad() által kreálandó adatok értékét. Ha a szexi férfiak top 10-es listája néhány adattal rövidebb a kelleténél, akkor:

```
$sexy_men    =    array_pad($sexy_men,    10,    "Péter    Moulding");
```

Miért használd az array_pad()-et a 10 darab (vagy bármely más számú) adat garantálásához? Képzelj el egy top 10-es listát, ami szavazási formátumban lesz (a 9. fejezetben leírt formátumkódok használatával). Szeretnéd berakni az öt vagy hat Oscar-jelölést a legjobb filmre, és hagyod, hogy az emberek a saját választásaikkal töltsék fel a listát. A manuális fo-

lyamat az, hogy a jelöléseket ciklussal beviszed a tömbbe, így létrehozol a formátumban minden jelölésnek egy-egy adatot, majd kiszámolod, hány üres hely kell, majd egy második ciklussal létrehozod az üres helyeket. A pad-függvénnyel mindez egy ciklussal megoldható.

Mérge

Az array_merge() két vagy több tömböt egyesít. Más-más technikát alkalmaz az asszociatív és a hagyományos tömbök esetén. A hagyományos tömböket, amelyek egész számokat tartalmaznak, egymásba fűzi és újra sorba állítja, míg az asszociatív tömböket adatról-adat-ra egyesíti, és az ugyanolyan kulcsú új adat felülírja a már meglévő adatot. A következő kód megmutatja, hogy az **array_merge()** hogyan egyesíti a **$friends** és a **$more** tömböket a $x-tömbbe, és hogyan egyesíti ezeket a hagyományos **while()**-ciklus a $y tömbbe. Mind a két módszer eredménye ugyanaz: mindkét esetben a második **Tom** felülírja az elsőt, de az array_merge() kevesebb idő alatt írja nieg és dolgozza fel:

```
$friends["Adam"] = "Hawaii";
$friends["Tom"] = "London";
$more["Alice"] = "Copenhagen";
$more["Tom"] = "Perth";

$x = array_merge($friends, $more);

while (üst ($k, $v) = each ($f riends) )
    {
    $y[$k] = $v;
    } while (üst ($k, $v) = each
($more) )
    {
    $y[$k] = $v;
```

Gyors megoldások

Tömbök létrehozása listából az array()-jel

Az **array** ()-függvény tömböt hoz létre egy listából. Egy egyszerű számozott tömböt valahogy így építhetünk:

```
$fruit = array("apple", "banana", "feijoa");
```

Asszociatív tömböt is létrehozhatunk az array()-függvénnyel. Figyeld meg a következőben, hogy rendeli a = > az értéket a kulcshoz.

```
$fruit_weight = array("apple" => 35, "banana" => 29, "feijoa" => 25);
```

Az **array()-t** és a fejezetben bemutatott függvényeket be lehet ágyazni, hogy többdimenziós tömböket hozzál létre, ahogy a következő péidában a **$carbohydrate** és a **Sprotein** mutatja:

```
$carbohydrate = array ("fruit" => array ("apple", "banana", "feijoa"),
    "vegetable" => array("carrot", "potato")); $protein = array("vegetable" => array("potato", "beans", "lentils"),
    "deadthings" => array ("cow", "sheep"));
```

Tömböket az **array**()-en belül is használhatsz többdimenziós tömböket létrehozására, mint azt a következő példa is mutatja:

```
$lunch = array ("carbohydrate" => $carbohydrate, "protein" => $protein) ;
```

így bármilyen listát beírhatsz tömbként, amely készen áll a feldolgozásra, és listákat többdimenziós tömbökké egyesíthetsz, amelyek tükrözik az adataid szerkezetét. Az 5. és 6. fejezetek megmutatják, hogyan olvashatod ki a listáidat az adatbázisból, és az utána következő „Gyors megoldások" hasznos segítséget nyújt a tömbök feldolgozásában.

Tömbök létrehozása sztringből az explode()-dal

Az expíodeQ-függvény feldarabolja a sztringet (a határoló jel alapján), és darabonként rakja be a tömbbe adatként. Ebben a példában vessző van a gyümölcsök nevei közt. Az explode()-függvény első paramétere a határoló jel, a második a sztring, mint ez lentebb látható. Az explode()-kódot egy másik kód követi, amely mutatja, mi kerül a tömbbe:

```
$fruit = explodel",", "apple,banana,feijoa");
```

```
$fruit[] = "apple"; $fruit[] = "banana"; $fruit[] = "feijoa";
```

Tömbök létrehozása sztringből az implode()-dal

Az **implode()** fordítottja az explode()-nak, a tömb adatait tetszés szerinti határoló jellel fűzi össze. A következő példában az SQL felépíthető' valamely tömb által tartalmazott mezőlistákból. Ez az eljárás tetszőleges karaktersorozatokra épülő lista szerkesztésére alkalmazható, beleértve a dinamikusan generált függvényeket, ábrák koordinátázott listáit, és a HTML- és XML-részeken belüli listákat.

Elemi SQL

Az SQL felépítésének illusztrálása céljából az elemi SQL legyen egy **select** parancs, melynek formáját alább mutatom be. Az egyes listáknak átadott részek a **select** utáni mezőlista, a where utáni feltételi lista, az **order by** utáni mezőlista és a **group by** utáni mezőlista lesznek.

```
select a, b, c, d from x where a = 'z' order by b, c group by b
```

Megjegyzés: Azok számára, akik nem ismerősek az SQL (Strukturált Kereső Nyelv) programnyelvben megjegyzem, hogy az SQL-adatok valamely táblázatból való kiválasztását teszi lehetővé. Ha a táblázat sok mezőt tartalmaz, és csak bizonyosakat akarsz kiválasztani, akkor szükséged van olyan parancsokra, mint például az „a mező", „b mező" kiválasztása stb. Ha a táblázat kevés mezőt tartalmaz, akkor ez ugyanolyan gyors, mint az összes mező kiválasztása a select segítségével. Ha nincsen where-paraméter, order by-paramé-ter vagy group by-paraméter, akkor az egészet kihagyhatod.

A mintatáblázat egy - belsőépítészeket kiszolgáló - helyi bolt bútor listáját tartalmazza. A vásárlás egyszerűsítése céljából a bolt szobákra van osztva. A táblázat neve **furniture** és a következő mezőket tartalmazza:

- szoba
- típus
- stílus
- szín
- márka

  magasság

  szélesség

  mélység

  súly

  anyag

  megrendelő

  szállítási idő

  összeszerelés ideje

  lehetőségek

select

Először is kell egy kiválasztási lista, hogy fel tudd építeni az SQL-ed select mezőlistáját. A lista tömb alakban kell a feldolgozás megkönnyítése érdekében. A következő példában használd a $select nevű tömböt:

```
$select []    "room";
Sselect[]     "color";
$select[]     "height";
$select []    "width";
$select[]     "depth[11];
```

where

Ezután kell a where-lista, hogy az adatbázis-táblázatból a megfelelő sorokat kiválaszthasd. Ha polcot akarsz venni, hogy legyen hol tartani a könyveidet, akkor könyvespolc kell neked, tehát hozd létre a $where nevű tömböt:

```
$where["type"]   =   "bookshelf";
```

order by

Szín, magasság és mélység szerint rendezd a listáját a könyvespolcoknak, hogy kiválaszthasd a fa megfelelő színét, és válassz magasabb és mélyebb könyvespolcot, hogy az igazán nagy könyveket is tárolhass. A $order tartalmazza a rendezési mezők listáját, és beépül a másik order by listába az SQL-ben:

```
$order[]    =   "color";
$order[]    =   "height";
$order[]    =   "depth";
```

group by

Nem kell a listát csoportokra osztani, így nincs group by-lista a könyvespolc választáskor az SQL-ben. Az SQL építő kódja ellenőrzi, hogy létezik-e $group-tömb, mielőtt használná a group by-lista elkészítéséhez.

SQL építése

Az SQL felépítésének első lépése az SQL select-részének felépítése a $sql-sztringen belül. Először azt ellenőrzi, hogy vajon létezik-e a select-hsta, és annak bejegyzéseit vesszővel és szóközzel választja el. Ha a select-hsta nem létezik, akkor alapértelmezés szerint a "*" jel használatával a táblázat mindegyik mezőjét ki kell választanod. Ha a $select nem tömb, akkor feltételezhető, hogy a kód más részében megszerkesztett előformázott lista.

```
$sql = "select ";
if(isset($select))
    {
    if (is array($select))
```

```
    $sql    .=    implode('',    '',
    Sselect); } else
        {
        $sql    .=    $select;
        }}
else {
    $sql.=    "*";
    }
```

A kód először definiálja, hogy a Ssql-sztring tartalmazza az SQL-t és beilleszti a select kulcsszót. Ez után ellenőrzi, hogy vajon a $select nevű tömb létezik-e, és beilleszti a "*" jelet, ha nincsen ilyen nevű tömb. Ha a $select létezik, akkor az is_array() használatával eldönti, hogy tömb-e, és ezután az implode() segítségével felépíti a kiválasztási listát.

Az implode() első paramétere az elválasztójel, amely tetszőleges sztnng lehet, így a nulla hosszúságú sztring is, arra az esetre, ha nincsen szükség elválasztó jelre. Az implodeQ második paramétere a tömb, amely akár standard, akár asszociatív tömb is lehet, ugyanis csak a tömb értékeit használja fel az eljárás.

A select parancsot az SQL from előírás követi. Az egyszerű példában a from előírás csak az itt felsorolt táblázatot nevezi meg.

```
$sql    .=    "    from    furniture";
```

A where-lista egy kicsit bonyolultabb, mert ez a lista asszociatív tömb és az SQL-nek mind a tömb értékére, mind a tömb kulcsára szüksége van. Szerencsére az összes érték az and segítségével egyesíthető, és ebben az esetben is használhatod az egyszerű implode() parancsot a többszörös where-értékek egyesítésére.

Ahhoz, hogy a select-kódhoz hasonlóan a kód egyszerű legyen, először az asszociatív értékpárból a where listatömböt egyszerű tömbbé kell átalakítanod, amely sztringek listáját tartalmazza, hogy ezáltal ezt a listát az implode() parancsban használhasd.

A következő lépésben a kód végigmegy a $where-tömbön, és a kulcsokat összekapcsolja az értékekkel. így egy új tömböt hoz létre, amelynek neve Swherelist. Először ellenőrzi, hogy a $where létezik-e, majd alapállapotba hozza azáltal, hogy a tömbmutatót a tömb kezdetére állítja be. Ezután végigmegy a tömbön. Ha a $where nem tömb, akkor azt kell feltételezned, hogy teljesen megszerkesztett where-sztring, és sztringként kell átadnod, mint alább:

```
if{isset(Swhere)) {
    if(is_array($where))
        { ~
        reset($where); while (üst ($k,    $v)
        =    each ($where) )
            {
            $where_list[]    =    $k    . " = ' " .    $v    . "' ";
```

```
else
    {
    $where üst = íwhere;
```

Ez a kód annyiban hasonló a **$select** utasítást megvalósító, korábban bemutatott kódhoz és az ezt követő **order** by- és **group** by-kódhoz, hogy a kód a változó használata előtt ellenőrzi, hogy a változó létezik-e. Erre az **is_array()** parancsot használja, ha a változó tömb, és a változót egyszerű sztringként kezeli, ha nem az. A fő különbség az, hogy a whileQ-ciklus soronként végigmegy a Swhere-tömbön és létrehozza **$where_list-et,** amely olyan tömb, amelynek elemei sztnngek, s ezek mindegyike egy SQL kiválasztási feltételt tartalmaz, amelyeket majd az SQL **where** részében **and** fog összekötni.

Megjegyzés: Számos adatbázis nem szereti az egész számok melletti egyszeres idézőjelet (Ő '), tehát előfordulhat, hogy a where kódodnak először ellenőrizni kell a mező típusát, mielőtt a $where_íist adatot létrehozná.

Most már átalakíthatod a $where_list-et SQL-lé. Ehhez a select-kódhoz hasonló kódot kell használnod. A korábbi select-hstáktól eltérően az SQL-ben most nincsen a hiányzó listát alapértelmezésben helyettesítő *. A where-lista alapértelmezése az, hogy kihagyja az SQL-ból hiányzó részt. A select-kódhoz hasonlóan ez a kód is feltételezi, hogy a **$where_list** teljesen megszerkesztett lista, ha a **$where_list** nem tömb, mint alább:

```
if (isset ($where__list) )
    {
    $sql .= " where ";
    if(is_array($where_list))
        {
        $sql .= implode(" and ", $where_list);
        }
    else
        {
        $sql .= $where_list;
```

A következő lépés az **order** by SQL létrehozása a Sorder-ből a $select-et végrehajtó kódhoz és a $where_list-kód utolsó részéhez hasonlóan. Az **order** by alapértelmezése az, hogy ez a rész hiányzik az SQL-ből. A select- és a where-kódhoz hasonlóan ez a kód is feltételezi, hogy a **$order** teljesen megszerkesztett lista, ha a **$order** nem tömb, mint alább:

```
if(isset($order)){
    $sql.=  "   order   by
    ";if(is    array($order))
        t ~
        $sql    .=   implode("r    ",    $order);
        }
    else
        {
        $sql    .=   $order;
        }
```

A group by SQL a **$group-ból** jön létre a **$order** esetében használt kóddal majdnem teljesen megegyező kód felhasználásával. A **group by** alapértelmezése az, hogy hiányzik ez a rész az SQL-ból. A korábbi kódhoz hasonlóan ez a kód is felteszi, hogy a **$group** teljesen megszerkesztett lista, hacsak a **$group** nem tömb, mint alább:

```
if(isset($group))

    $sql .= " group by "; if(is
    array($group))

        $sql .= implode (", ", $group);

    else {
        $sql .= $group; }
```

Amikor kinyomtatod a $sql-t, a következőt kell kapnod:

```
select room, color, height, width, depth from furniture where
type = 'bookshelf' order by color, height, depth
```

Az adatok tartományának kijelölése a range() segítségével

Ha címkék nyomtatásához vagy más megszokott eljáráshoz szükséged van arra, hogy rendelkezésedre álljon számoknak egy tömböt alkotó listája, akkor a range() a listát az első paraméterében megadott számtól kezdve és a második paraméterében megadottal befejezve építi fel számodra, amint azt a következő kód első sora mutatja. A kód többi része mutatja be azt, hogy mit kellett volna írnod akkor, ha nem állna rendelkezésre a rangeQ, amit tömb követ, amint a 3.1 ábra mutatja.

```
$labels = rangé(5, 10);

for($i = 5; $i <= 10;
    $i++) { $labels[] = $i;
```

Key	*Valuel*
0	**5.**
EJ	**1**
1?	**7**
[3 :	8
U	9
5	10

3.1 ábra Tőmbértékek létrehozása a range()-dzsel

Kulcstartomány kijelölése a range() és az array_flip() segítségével

Ha címkék nyomtatásához szükséged van arra, hogy rendelkezésedre álljon számoknak egy tömböt alkotó listája, mint például a nyilvántartási címkék esetében, akkor esetleg a tömbkulcsba akarod belerakni a kijelölt tartományt, hogy így később majd értékként hozzáadhass még valamit (például dátumot). Az **array_flip()** tömbön belül átvált az értékek és a kulcsok között, ezáltal a **rangé**() segítségével létrehozott számtartományt a kulcsba viszi át. A következő kód kibővíti „Az adatok tartományának kijelölése a **rangeQ** segítségével" részben szereplő kódot, először egy munkatömbben létrehozva a megfelelő tartományt, majd a **$labels** tömbbe rakja oly módon, hogy a tartomány a kulcsba kerüljön. A **range{)** és az **array_flip-kód** első két sora után az a kódlista következik, amelyet az **array_flip()** kitalálása előtt kellett volna írnod. A 3.2 ábra mutatja az eredményül kapott tömböt:

```
$numbers = rangé(5, 10); $labels
= array_flip($numbers);

while (üst(Sk, $v) = each ($numbers) )
    {
    $labels[$v] = $k;
```

	Value
5	ˇ
|6	L.. J
7	2
8	3
9	E
10	5

3.2 ábra Tömbértékek létrehozása a range()-dzsel és az array_flip()-el

Duplikát tömbértékek megszüntetése az array_flíp() segítségével

Ha valamely tömb listát tartalmaz és a duplikált értékeket meg akarod szüntetni, akkor figyelmedbe ajánlom az **array_flip()** kétszeri alkalmazását, amint a lenti kód mutatja. A mintatömbben a **keyboard** kétszer szerepel:

```
$labels[] = "keyboard";
$labels[] = "mouse"; $labels[]
= "keyboard"; $labels[] =
"monitor"; $x =
array^flip(\$labels) ;
```

```
$labels    =
array_flip($x);
ksort(\$labels);
```

A 3.3 ábra mutatja a tömböt a kétszeri átváltás előtt és után. Láthatod, hogy csak egyetlen keyboard maradt. Az első átváltás a kulcsot az értékbe, az értéket a kulcsba viszi át. Minthogy mindkét keyboardnak ugyanaz az értéke, ezért ugyanabba a kulcsba mennek át, a második keyboardérték pedig felülírja az első keyboardértéket. Amikor a második átváltás megtörténik, akkor a tömb az eredeti alakjába megy vissza, azonban az eredetitől eltérő sorrendben, ezért hozzáadtam a **ksort()** parancsot, hogy visszarendezze a tömböt az eredeti sorrendbe. A ksort() megtartja a kulcs és érték párokat, miközben a kulcs szerint rendez.

\Key\Value (o" Ikeyboard	fi--------'i ---------*___~~-
|l ;mouse	1 Imouse
[2	2
|3 [monitor	,3 imonitor

3.3 ábra A tömb a két array_flip() előtt (balra) és után (jobbra)

Tömb véletlenszerűsítése a shuffle() segítségével

Tegyük fel, hogy van egy CDéd, amit le akarsz játszani, de már unod mindig ugyanabban a sorrendben hallgatni. Hasonlóképpen, ha van egy weboldalad egy csomó bannerrel, talán szeretnéd, hogy a bannerek véletlenszerű sorrendben jelenjenek meg, hogy érdeklődést keltsenek. A **shuffleQ**-függvény az a varázslatos függvény, amivel véletlenszerű listát lehet létrehozni.

Először is kell egy lista, amit szeretnél megkeverni. Itt van egy blues-CD számainak listája:

```
$tracks[] = "Smokestack Lightnin'";
$tracks[] = "My Babé";
$tracks[] = "High Heeled Sneakers";
$tracks[] = "We're Gonna Make It";
$tracks[] = "I'm In The Mood";
$tracks[] = "Forty Four";
$tracks[] = "Help Me";
$tracks[] = "I'd Rather Go Blind";
$tracks[] = "Wang Dang Doodle";
$tracks[] = "I'm Your Hoochie Coochie Man";
$tracks[] = "Walkin' The Blues";
$tracks[] = "Walking By Myself";
$tracks[] = "When The Lights Go Out";
```

A számok összekeverése egyszerű, csak írd be ezt:

```
shuffle($tracks);
```

Most jelenítsd meg a tömb első adatát. Ez az adat bármelyik szám lehet.

Ha a számok sorrendje nem tűnik eléggé véletlenszerűnek, az azért van, mert a shuffle() a PHP véletlen szám generátorát használja, és ez a véletlen szám generátor az időn alapszik. Azt találtam, ha a Frissít gombot 2 másodpercenként nyomom meg rendszeresen, akkor minimális lesz a változás a számok sorrendjében, olyan nagy változások nélkül, amiket egy igazi véletlen szám generátortól elvárnánk. Bizonyos körülmények között ez nem biztos, hogy elég véletlenszerű, így az Unix időalapú (másodperc alapú) véletlen szám generátora helyett használd az **srand()** függvényt. A **microtimeQ** függvény jó helyettesítője az srand() **time()** függvényének, mert a **microtime()** mikromásodpercben számol, így nem valószínű, hogy sikerül kétszer ugyanazt az eredményt elérni. A részletek a **rand()** függvény kapcsán a 2. fejezetben megtalálhatóak.

A következő példa megmutatja az **srand{),** a **microtime(),** és a **shuffle()** együttes használatát, és a 3.4. ábra mutatja a számlista kétszeri gyors egymás utáni keverésének eredményét. A srand()-ot parancsfájlonként csak egyszer kell használni, függetlenül attól, hányszor használod a shuffleQ vagy a **rand()** függvényeket. A **microtime()** használata csak azért szükséges, hogy csökkentsük a parancsfájl lefuttatásának ismétlődését azáltal, hogy egész másodpercek helyett a másodperc töredékét vesszük alapul.

```
srand((double) microtime() * 1000000);
shuffle($tracks);
```

íJHigh Heeled Sneakers	Walking By Myself
Help Me	||Forty Four
I'd Raíher go Blind	|Wang Dang Doodle
|My Babé	!|I'm Your Hoochie Coochie Man
' Smokestack Lightnin'	**fWe're** Gonna Make **It**
FmüiTheMood	[When The Lights Go **Out**
We're Gonna Make It	jfWalkin' The Blues
||Fourty Four	High Heeled Sneakers
Wang Dang Doodle	I'm In The Mood
I'm Your Hoochie Coochie **Man**	Help Me
Walking By Myself	[i'd Rather Go Blind
Walkin' The Blues	**[MyBabe**
When The Lights Go Out	'Smokestack Lightnin'

3.4 ábra Lejátszó lista két véletlen sorrendben

Bannerek véletlenszerűsítése az array_rancl()-dal

Ha nem kell az egész listát összekeverned, és csak egy véletlenszerű adat kell (mint például, ha csak egy banner van az oldal tetején), akkor használd az array_rand()-ot. Az array_rand()-függvény alapértelmezésben adott számú adatból választ ki egy adatot. Majdnem tökéletes egy banner listából való kiválasztására.

Először is kell a reklámok listája. Talán nem hallottál az alábbi vállalatokról, de olyan termékeket gyártanak, amiket én kedvelek:

```
$banners[] = "Coriolis";
$banners[] = "Coopers";
```

```
$banners[] = "Penfolds"/
Sbanners[] = "Volvo";
Sbanners[] = "Compaq";
$banners[] = "AMD";
Sbanners[] = "PHP";
Sbanners[] = "Apache";
```

Bármelyik bannert kiválaszthatod és megjelenítheted az alábbi egyszerű kóddal. A srand()-ot csak egyszer kell lefuttatni a parancsállományban az array_rand() első futtatása előtt. A print() parancs élethűbb lesz, ha középre igazítod és kiemeled a bannert, ahogy ezt szokták:

```
srand((double) microtime() * 1000000);
$banner = array_rand(Sbanners);
print("<p align=\"center\"xfont color=A"339933\" size=\" + 4\">" .
    \5banners[\$banner] . "</font></p>");
```


Az array_rand() a választás kulcsát adja vissza, nem az értékét, így használhatod a választás kulcsát vagy az értékét is. Ha egy számot választasz ki véletlenszerűen egy CD-ről (mint azt az előző Gyors megoldásokbeli példában is láttuk), akkor lehet, hogy az érték kell (a szám címe) a megjelenítéshez, de lehet, hogy a kulcs kell (a szám sorszáma), hogy azt megadd a CD-t olvasó és lejátszó programnak. Ebben a banneros példában először megnézem a reklám nevét, amiben benne foglaltatik a .jpg kiterjesztés is.

A következő kódpélda véletlenszerűen választ ki egy bannert a $banners tömbből, majd megmutatja a banner kulcsát, hogy az array_rand() segítségével ellenőrizhesd a visszaadott értéket. Ezután megmutatja a reklám nevét. így ellenőrizheted, hogy a $banner-tömbből kapott érték helyes-e. Ezután megmutatja a banner fájlnevét annak megfelelő módon, ahogy azt egy HTML <img> tag-be raknád.

```
$banner = array_rand(Sbanners);
print("Banner number: " . Sbanner);
print("<br>Banner name: " . Sbanners[Sbanner]);
print ("<br>Banner file name: " . $banners[Sbanner] . ".jpg");
```

Esetleg szükséges lehet egy másik véletlen adatra. Egyes honlapok három mozgó bannert helyeznek el minden oldalra. Itt van az a kód, amely három véletlen bannert választ ki a $banners tömbből.

```
Sbanner_list = array^rand(Sbanners, 3);
```

Jegyezd meg, hogy az array_rand() a tömb adatának értékét adja vissza, ha egyetlen adatot választasz ki, és az értékek tömbjét adja vissza egynél több adat esetén. A reklámok listájának használatához a while-ciklushoz hasonló kódot használhatsz, amint alább látod. A kód az előző példákban használt a normál while(), Iist() és each() parancsokat használja és rendre a print-utasítást adja ki, amely a HTML <ímg> tag-et állítja elő a $banners tömbből vett fájlnevek felhasználásával.

```
while (üst ($k, $v) = each ($banner_list) ) { print ("<img src=\""
    . $banners[$v] . ".jpg\" border=\"0\">");
```

Mivel az array_rand() a tömb kulcsát adja vissza, ezért bonyolultabb tömböket is kezelhetsz vele. A bannertáblázatok esetében jó ok van arra, hogy a bannertáblázat bonyolultabb legyen. A következő kód, olyan többlet információval kiegészítve, amely elősegíti, hogy a banner hatásosabb legyen, megmutatja, hogy a Sbanners-tömb egy adatának kibővítése milyen lesz:

```
$banners[] = array("name"=>"Coriolis", "height"=>50,
    "width"=>200, "alt"=>"Check out their cool Black Books");
```

A következő kód a $banners-tömbből egyetlen bannert mutat, amelyet minden extra adattal kibővít abból a célból, hogy annak megjelenése sokkal jobb legyen. Az első sor ugyanaz a véletlen kiválasztás, mint az előző példában. A második sor a kiválasztás eredményét a $image-be rakja, mivel a választott érték most már nem egyetlen sztring, hanem sztringek tömbje. A print()-utasítás kiterjesztett formája szerepel, amely a $image-ben szereplő összes adatot felhasználja az <img> tag-en belüli megfelelő tag-ekben.

```
$banner = array_rand(Sbanners) ;
$image = $banners[$banner] ;
print("<img src=\"" . $image ["name"] . ".jpg\" border=\"0\""
    . " alt=\"" . $image["alt"] . "\""
    . " height=\"" . $image [ "height" ] . "V"
    . " width=\"" . $image["width^M] . "\">");
```

Amikor egy honlapot készítesz, az első dolgod az, hogy használat előtt ellenőrizd, vajon mindegyik tömbattribútum létezik-e, és néhány hasznos alapértelmezést beállíts. A kódot közvetlenül a Simage létrehozatala utáni sorok beiktatásával ki lehet bővíteni, és így már biztos lehetsz abban, hogy jó alapértelmezéseket kapsz. A következő példa gondoskodik arról, hogy a kép alt szövegének legyen értéke. A Simage ["alt"] értéke a kép neve lesz, ha a Simage ["alt"] hiányzik vagy üres sztringet tartalmaz.

```
$image = $banners[$banner];
if(!isset($imagefalt"] ) or !strlen($image["alt"]) )
    {
    $image["alt"] = $image ["name"];
```

Fájl tömbbe olvasása a file() segítségével

Egy fájlnak valamely tömbbe olvasását kezdd a test.txt-fájllal, amely néhány sor szöveget tartalmaz. Olvasd be ezt a fájl a file()-függvény segítségével egy tömbbe, mint alább:

```
$lines = fileftest.txt");
```

Egyszerű. Ez a fájl weboldal is lehet, mint ez:

```
$lines = file ("http://petermoulding.com/index.html");
```

A tömb mindegyik adata a \n-, ún. newline-karakterrel végződik, amely elválasztja a sorokat egymástól. Alkalmas sztring-függvény segítségével eltávolíthatod a newline-karaktert. A Windows operációs rendszerből beolvasott fájlok sorai return-t követő newline-nal is

végződhetnek, \r\n, az Apple Mac fájljai végződhetnek a \n, \r\n, \n\r bármelyikével. Szerencsére a **file()** parancs a newline-karaktert a helyén hagyja, így eldöntheted, mit teszel vele.

Nem mindegyik fájl tartalmaz olyan sorokat, amelyek a newline-karakterre végződnek. Azonban a **file()** ekkor is egy vagy esetleg néhány hatalmas elemet tartalmazó tömbként fogja a fájlt beolvasni, ahogy azt a JPEG-fájlok olvasása során tapasztaltam. A JPEG-ek tartalmaznak néhány olyan bájtot, amelyeknek ugyanaz az értéke, mint a **newline,** így tehát a **file()** néhány kisebb részre bontja fel a JPEG-eket, melyeket számba vehetsz és kezelhetsz. Ugyanakkor jobb módszerek is vannak az olyan bináris fájlok beolvasására, mint például a JPEG, amint az a 8. fejezetben szerepel.

Tömb rendezése érték szerint a sort() segítségével

Fényképezőgépbe való filmmárkák listájának növekvő sorrendbe rendezését egy standard tömb definiálásával kell kezdened, amely a márkaneveket adja meg, mint alább:

```
$film[]  = "Konika";
$film[]  = "Ilford";
$film[]  = "Kodak";
$film[]  = "Fuji";
$film[]  = "a_name_in_lower_case";
```

Ezután a sort()-függvény segítségével rendezed a listát az alábbi módon:

```
sort{$film);
```

A lista kinyomtatásához használd az alábbi kódot:

```
while (üst ($k,     $v)     =
    each($film)) {
    print("<br>"     .
    $v); }
```

Eredményül a következő listát látod majd:

```
Fuji
Ilford
Kodak
Konica
a_name_in_lower_case.
```

Megjegyzés: A kisbetűvel kezdődő adat nem illeszkedik az abc-sorrendbe, mivel a sortQ parancs megkülönbözteti a kis- és nagybetűket. Ha olyan rendezést akarsz, amely a kis-és nagybetűt olyan módon kezeli, ahogy azt elvárnád, akkor a „ Tömb értek szerinti természetes rendezése a natsortQ segítségével" részt nézd meg, amely később szerepel a „Gyors megoldások" részben.

Asszociatív tömb rendezése érték szerint az asort() segítségével

Ha egy asszociatív tömböt a **sort()** segítségével rendezel, akkor azt találod, hogy az összes kulcsból szám lett. Az asort() parancs az asszociatív tömböket úgy rendezi az érték szerint, hogy közben megtartja az eredeti társított kulcsokat. A következő példa filmek márkáit és a származási országot tartalmazó tömböt használja. Itt a mintatömb:

```
$film["Konika"] = "Japán";
$film["Ilford"] = "England";
$film["Kodak"] = "Mexico";
$film["Fuji"] = "Japán";
$film["a_name_in_lower_case"] = "anywhere";
```

Az asort() segítségével a filmek márkáit a származási ország szerint sorrendbe rendezheted. Ugyanúgy, mint a sort()-függvény, az **asort()** is a nagybetűk után sorolja a kisbetűket, mivel az **asort()** az egyes karakterek bináris értéke alapján rendez. A következő kód bemutatja, hogyan működik az **asort()** a **$film-tömbön:**

```
asort($film);
```

A következő segítségével kinyomtathatod a listát:

```
while (üst ($k, $v) = each < $ f ilm) )

   print("<br>" . $v $k);
```

Az eredmény az alábbi lista:

```
England Ilford
Japán Fuj i
Japán Konica
Mexico Kodak
anywhere a_name_in_lower_case
```

Tömb érték szerinti fordított sorrendbe rendezése az rsort() segítségével

Bizonyos helyeken esetleg arra van szükséged, hogy a tömböt az értékek szerint fordított sorrendbe rendezd. Filmcímek listájának fordított (csökkenő) sorrendbe rendezéséhez használd az **rsort()** parancsot. Ugyanúgy, mint a **sort(),** az **rsort()** sem látja a kis- és nagybetűket, hanem a bináris értékek alapján rendez. Kezdd először egy normál tömb definiálásával, amely az egyes filmcímeket tartalmazza, amelyek rendezni akarsz. így például:

```
$films[] = "Taxi Driver";
$films[] = "Heat";
$films[] = "Men of Honor";
$films[] = "The Adventures of Rocky & Bullwinkle" ;
$films[] = "Great Expectations" ;
```

```
$films[]  =  "Cape  Fear";
$films[]  =  "any  other
film";
```

Ezután rendezd a listát az rsort()-függvénnyel, mint alább:

```
rsort( $ f ilms);
```

A következő segítségével kinyomtathatod a listát:

```
while (Üst ($k,  $v)  =  each ( $f ilms) )

    print("<br>"  .
    $v); }
```

Az eredmény a következő listához lesz hasonló, a kisbetűvel kezdődő adatok a nagybetűsök előtt szerepelnek:

```
any  other  film
The  Adventures  of  Rocky  &  Bullwinkle
Taxi  Driver
Men  of  Honor
Heat
Great  Expectations
Cape  Fear
```

Asszociatív tömb érték szerinti fordított sorrendbe rendezése az arsort() segítségével

Filmcímek asszociatív listájának fordított (csökkenő) sorrendbe rendezéséhez használd az **arsortQ** parancsot. A **sort()** és **asort()** parancshoz hasonlóan az **arsortQ** is a bináris értékek alapján rendez. Kezdd egy asszociatív tömb definiálásával, megadva a filmek címét és főszereplőiket, mint alább:

```
$films["John  Travolta"]  =  "Swordfish";
$films["Mark  Wahlberg"]  =  "Planet  Of  The  Apes";
$films["Michael  J.  Fox"]  =  "Atlantis:  The  Lost
Empire"; $films["Alcatraz"]  =  "Escape  From  Alcatraz";
$films["anyone"]  =  "any  other  film";
```

Rendezd a listát a arsort()-függvény segítségével, majd nyomtasd ki az eredményt:

```
arsort{$films);
while (Üst ($k, $v) = each ($f ilms) )

    print("<br>" . $v . " " . $k);
   }
```

Az eredmény a következő lista:

```
any other film anyone
Swordfish John Travolta
Planet Of The Apes Mark Wahlberg
Escape From Alcatraz Alcatraz
Atlantis: The Lost Empire Michael J. Fox
```

Asszociatív tömb kulcs szerinti rendezése a ksort() segítségével

A ksortQ-függvény az asszociatív tömböt kulcs szerint rendezi. Az ideillő **krsort()** asszociatív tömböket kulcs szerinti fordított sorrendbe rendezi. A következő példa filmek és gyártók tömbjét használja. Itt a mintatömb:

```
$plastic["Sarán Wrap"] = "Dow Chemical";
$plastic["GLAD Wrap"] = "Clorox Australia Pty Limited";
$plastic["Multix"] = "Multix Pty Limited" ;
$plastic["generic plastic wrap"] = "just about anyone";
```

Kulcs szerinti rendezéshez rendezd az asszociatív tömböt a **ksort()** segítségével. így a műanyag csomagotómárkáknak a nevük szerint rendezett listáját kapod. Ugyanúgy, mint a **sort()** és az **asort(),** a ksort() is a nagybetűk után helyezi a kisbetűket. A következőképpen csinálhatod:

```
ksort($plastic);
```

A következővel kinyomtathatod a listát:

```
while (üst ($k, $v) = each
    ($plastic) ) {
    print("<br>" . $k . " - "
    . $v) ; }
```

Eredményül az alábbi listát kapod:

```
GLAD Wrap - Clorox Australia Pty Limited
Multix - Multix Pty Limited
Sarán Wrap - Dow Chemical
generic plastic wrap - just about anyone
```

Tömb érték szerinti természetes rendezése a natsort() segítségével

A natsort()-függvény a listát természetesebb növekvő sorrendbe rendezi, mint a **sort().** Számos definíciója van a természetes rendezésnek, és ezek a használt nyelv és karakterkészlet függvényében eltérnek egymástól. A **natsort()** történetesen jól működik olyan fájlnevek esetében, mint például a számozott képek.

Kezdd először a képfájlok nevét tartalmazó tömb definiálásával, mint alább:

```
Simage[] = "grease.jpg";
$image[] = "condensation.jpg";
$image[] = "greaseO3.jpg";
$image[] = "mildew4.jpg";
$image[] = "mildew04 0.jpg";
$image[] = "Mildew54.jpg";
$image[] = "MILDEW44.jpg";
```

```
$image[] = "slime.jpg";
$image[] = "butter.jpg";
$image[] = "rust 3 b.jpg";
$image[] = "rust3 a.jpg";
$image[] = "rust 3c.jpg";
$image[] = "rust3e.jpg";
```

Először rendezd a listát a sortQ segítségével, hogy lásd mi történik:

```
sort($image);
```

Nyomtasd ki a listát az alábbi módon:

```
whiledist ($k, $v) = each ($image) )

    print ("<br>" . $v);
```

Eredményként a következő listát kapod:

```
MILDEW44.jpg
Mildew54.jpg
butter.jpg
condensation.jpg
grease.jpg
greaseO3.jpg
mildewO40.jpg
mildew4.jpg rust
3 b.jpg rust
3c.jpg rust3
a.jpg rust3e.jpg
slime.jpg
```

Megjegyzés: Vedd észre, hogy a rust 3c előbb van a rendezésben, mint a rust3, mivel a szóközkarakter a 3 elé kerül a rendezésben.

Ezután rendezd a listát a natsort()-függvénnyel, a következő begépelésével:

```
natsort(Simage);
```

Az eredmény az alábbi lista:

```
MILDEW44.jpg
Mildew54.jpg
butter.jpg
condensation.jpg
grease.jpg
greaseO3.jpg
mildewO40.jpg
mildew4.jpg rust3
a.jpg rust 3
b.jpg rust 3c.jpg
rust3e.jpg
slime.jpg
```

Amint láthatod, az natsortQ parancs a különböző helyekre beszúrt szóköz ellenére képes volt a helyes sorrendbe rendezni a 3a, 3b, 3c és 3e rustképeket. A rustképek ugyanolyan sorrendben vannak, mintha nem lennének szóközök a nevükben. Ez a tulajdonság hasznos Windows, NT, nagyszámítógépek vagy Mac OS alatt szervezett fájlok esetén.

A mildewO4O, amint láthatod, még mindig a mildew4 előtt van, mivel a natsort() a számok kezelése során nem távolítja el a kezdő zérusokat. Bár nem áll rendelkezésre olyan lehetőség, hogy ezt a natsort() segítségével elvégezhesd, az usort() segítségével megcsinálhatod. Ez olyan rendező függvény, amely megengedi, hogy egy függvény segítségével megadd az összehasonlítási szabályokat. Ezt elemzi a következő rész.

A kis kezdőbetűs adatok a nagy kezdőbetűsök után vannak, mivel a natsort() nem különbözteti meg őket. A következő példa ugyanazt a tömböt mutatja, azonban a natcasesortQ-függvény segítségével történt rendezés után. Ez a függvény már kezeli a kis- és nagybetűket. Itt szerepel a kód, amely a natcasesort() használatát mutatja:

```
natcasesort($image);
```

A következő lista a natcasesort() eredményét mutatja, amely már teljes azáltal, hogy a "mildew"-ek a helyes sorrendben szerepelnek.

```
butter.jpg
condensation.jpg
grease.jpg
greaseO3.jpg
mildew040.jpg
mildew4.jpg
MILDEW44.jpg
Mildew54.jpg rust3
a.jpg rust 3
b.jpg rust
3c.jpg rust3e.jpg
slime.jpg
```

A példában az egyik fájlnév *4-rc* végződik, egy másik pedig *040-ra.*. Az ilyen helyzetekben a számítógépek alapértelmezése szerint a *040* előbb jön a sorrendben, mint a 4, azonban az emberek szerint a *4* van előbb, mintha ez a *004* helyett állna. Az európai betűk esete egy másik példa arra, a rendezés bonyolult lehet: *E* (ékezetes E) bináris értéke 200, amely jóval a standard abc után következik, azonban azt várhatnád, hogy az *E* betűhöz hasonló helyre kell rendezni.

Bizonyos helyzetekben szükséged lehet fonetikus rendezésre. A legegyszerűbb rendezési rendszer a Soundex-rendszer, amelyet a PHP soundex()-függvénye támogat. Pontosabb (és terjedelmesebb) eljárás a metafonok használata a metaphone()-függvény segítségével. A Soundex általában egy rögzített négykarakteres azonosítót használ, jóllehet a különböző adatbázisok - mint például az Oracle által támogatott MySQL — változó hosszúságú Soundex-értékeket használnak. A metafonok változó hosszúságúak, az angol nyelv kiejtési szabályaiból többet használnak fel, azonban túlságosan újak ahhoz, hogy széles körben támogatottak legyenek. A metaphone() és a soundex() használatának szabályait a 12. fejezet tartalmazza.

Tömb nem természetes rendezése az usort() segítségével

Néha szükséged lehet arra, hogy egy tömböt valamilyen speciális sorrendbe rendezzél, és a hagyományos rendező függvények nem adják meg erre a lehetőséget. Az usortQ-függvény megengedi, hogy mindent te intézz. Először is átveszi a rendezni kívánt tömböt, majd a tömb elemeinek összehasonlításához használt függvény nevét. Te szolgáltatod az összehasonlítási függvényt. Az általad megadott függvény két összehasonlítandó értéket bemenő adatként fogad, visszaad valamilyen értéket, amely utal arra, hogy a rendezésnek milyen műveletet kell elvégeznie. Visszaadhat tetszőleges pozitív egészet, amely azt jelzi, hogy a két érték nincsen sorrendben, a 0 jelzi, hogy a két érték egyenlő, a negatív szám jelzi, hogy a két érték sorrendben van.

A következő példa képfájlok nevét tartalmazó tömböt használ. Ebben a példában a képek különböző típusúak, a felhasználó által definiált rendezés a fájl típusa által fog rendezni. Egy típuson belül a fájlokat abc-sorrendbe rendezi, a neveket az összehasonlítás előtt kis kezdőbetűsre alakítva.

Egy lépést beiktatok, amely *a.jpeg-et* összehasonlítás előtt *jpg-re* cseréli. így a/jpeg mindkét fajta írásmódját együtt rendezi majd. A rustképek helyes rendezéséhez előbb eltávolítóm a szóközöket.

Itt a példatömb:

```
$image[] = "grease.jpg";
$image[] = "condensation.jpeg" ;
$image[] = "greaseO3.JPG";
$image[] = "mildew4.jpg";
$image[] = "mildewO4O.jpg";
$image[] = "Mildew54.jpg";
$image[] = "MILDEW44.jpg";
$image[] = "slime.gif";
$image[] = "butter.pgn";
$image[] = "rust 3 b.jpg";
$image[] = "rust3 a.jpg";
$image[] = "rust 3c.jpg";
$image[] = "rust3e.jpg";
```

Az **usort()** függvénynek szüksége van egy olyan függvényre, amely -1 értéket ad vissza, ha az értékek sorrendben vannak, és +1 értéket, ha nincsenek, 0-t, ha egyenlők. A következő függvény, az **unnatural()** éppen ezt teszi. Először nézd meg a függvényt, majd a következő példában a függvény használatát, végül az eredményt. Ezt a függvény egyes részeinek magyarázata követi:

```
function unnatural($x, $y)

    $x = strtolower($x);
    $y = strtolower($y);
    $x = str replace(" ", "" $x
    $y = str replace(" ", "" $y
    $p = strrpos($x, ".")
```

```
if($p === falsé)

    $xf = $x;
    $xt = "";

else
    {
    $xf = substr($x, 0, $p);
     $xt = substr($x, $p + 1);

$p = strrpos($y, ".");
if($p === falsé)
    {
    $yf = $y;
    $yt = "";

else

    $yf = substr($y, 0, $p);
    $yt = substr($y, $p + 1);

if($xt == "jpeg")

    $xt = "jpg";

if($yt == "jpeg")

    $yt = "jpg";

if($xt < $yt)

    return(-1);

elseif($xt > $yt)

    return(1);
    }
else
    {
    if($xf < $yf)
        {

        return(-1) ;

    elseif($xf > $yf)

        return(1);

    else

        return(0);
```

A listának az usort()-függvény és az unnaturalQ-függvény segítségével történő rendezéséhez írd:

```
usort($image,      "unnatural");
```

A kinyomtatáshoz használd a következőt:

```
while (üst ($k,      $v)      =      each ( $image)
    ) ; {print("<br>"      .      $v);
```

Az eredmény:

```
slime.gif
condensation.jpeg
grease.jpg
greaseO3.JPG
mildew040.jpg
mildew4.jpg
MILDEW44.jpg
Mildew54.jpg rust3
a.jpg rust   3   b.jpg
rust   3c.jpg
rust3e.jpg
butter.pgn
```

Vedd észre, hogy az eredményben a GIF-fájl jön legelőször, majd az összes JPEG-fájl, és végül a PNG-fájl. A JPEG-fájlok a helyes sorrendben vannak, függetlenül attól, hogy milyen a fájl kezdőbetűje, és hogy a JPEG hogyan van írva.

Az unnatural()-függvény két értéket fogad el, $x-et és $y-t, amelye(ke)t az ucase() ad át. Mindkét értéket azonnal átfordítja kisbetűkre a strtlowerQ-függvény, hogy a fájlnevek, attól függetlenül, hogy a gépíró milyen kezdőbetűvel írta, amikor létrehozta őket, ugyanolyan helyre kerüljenek.

A $x = str_replace(" ", "", $x); sor és a $y-ra vonatkozó megfelelő sor eltávolítja a szóközöket a fájlnevekből, hogy a rustképek logikus sorrendbe kerüljenek. Bár nem mindegyik operációs rendszer engedi meg a fájlnevekben a szóközöket, mindegyik operációs rendszer megenged valamilyen szóközhöz hasonló karaktereket, beleértve a kötőjelet és alulvonást, amelyeket a folyamat ezen pontján most eltávolíthatsz.

A következő rész, amelyet a $x esetében újra leírok, a strrpos()-függvény segítségével megtalálja a fájlnében az utolsó pontot (.), a substr()-függvény segítségével az utolsó pont pozíciójánál két részre bontja a fájlnevet, és a pont előtti szöveget a $xf változóba helyezi, a pont utáni szöveget a $xt változóba. A soron következő összehasonlításban a $xt, a típus szerepel. A $xf, a fájl neve csak akkor kerül felhasználásra, ha a típusok egyenlőek.

Ezt mutatja a következő kód:

```
    $p    =    strrpos($x,
    "."); if($p    ===    falsé)
```

```
        $xf  = $x;

        $xt  = "";
    else
    {
        ($xf  substr ($x. 0, $P);
        $xt   substr ($x, $P +
```

A kód két következő része, amint alább látható, egyszerűen csak ellenőrzi, hogy $xt vagy $yt értéke *jpeg-e, és* kicseréli *ajpg-értékre.* Mivel a fájl típusának teljes neve *a jpeg,* ugyanakkor a legtöbb ember csak *jpg-nek* írja, a kettőt azonossá teheted a fájltípusok összehasonlítása előtt, mint itt:

```
if($xt  ==  "jpeg")

        $xt  =  "jpg";

if($yt  ==  "jpeg")
        $yt  =  "jpg";
```

A következő kód összehasonlítja a fájltípusokat, $xt és $yt értékét, és -1-et ad vissza, ha sorrendben vannak, 1-et, ha nincsenek sorrendben. Az utolsó else a fájltípusok megegyezése esetén szükséges eljárásra vezet át:

```
if($xt  <  $yt)

    return(-1);

    elseif($xt > $yt)

    return (1);

    else
```

A fájlnevekre vonatkozó eljárás a fájltípusokra vonatkozó eljárás másolata azzal az eltéréssel, hogy $xt helyett $xf, $yt helyett $yf szerepel. Ha a fájlnevek megegyeznek, akkor a kód 0 értéket ad vissza, amely a teljes megegyezésre utal.

Bár a kód hosszú, gyorsan és könnyen lehet megjegyzésekkel dokumentálni. Azok, akik sorokat akarnak megtakarítani, a str_replace() parancsot a strtolower() belsejébe helyezhetik, sok helyet takarítva meg ezáltal.

Amikor én adatbázisok segítségével kezelek fájlokat képkezelő vagy más hasonló alkalmazásokban, gyakran már az eljárás korai szakaszában levágom a fájltípusokat, és külön mezőben tárolom el őket. így az SQL segítségével kiválaszthatók és rendezhetők, ahelyett, hogy az usortQ-függvényt kelljen használnom.

Fordított ciklus a tömbelemeken

Egy tömb elemein fordított sorrendben is végigmehetsz, ha előbb a tömböt fordított sorrendbe rendezed, majd előrehaladsz. Azonban ha csak néhány utolsó elemet akarsz gyorsan megtalálni, ahhoz ez a tömbelemek kezelésének nagyon lassú módja lenne. Jelen szakasz elmagyarázza, hogyan lehet egyszerűen a végén kezdeni és visszafelé haladni.

Először is vegyük az alábbi mintatömböt:

```
$mineral      "gold";
$mineral      "palládium";
$mineral      "iridium";
$mineral      "platinum";
$mineral      "ruby";
$mineral      "rhodium";
$mineral      "ruthenium";
$mineral      "indium";
$mineral      "tantalum";
```

Kezdd a tömb utolsó elemének kiválasztásával. Erre az **end()**-függvényt használd, amely az utolsó adatot adja vissza, mint alább:

```
print(end($mineral));
```

Ezután a következőt kell kapnod:

```
tantalum
```

A tömb előző adatát a prev()-függvény segítségével érheted el, amely a tömb fájlmutatóját az előző adatra helyezi, és visszaadja annak értékét:

```
print(prev{$mineral));
```

Ezután a következőt kell kapnod:

```
indium
```

Visszafele lépegethetsz a tömbön a **whíle()** és **prev()** használatával, mivel a **prev()** a hamis értéket adja vissza, amikor a tömb kezdetén túlszalad, mint alább:

```
while{$x    =    prev($míneral))
    {
    print($x) ;
```

Van azonban egy probléma a prev()-függvénnyel. Az üres tömbelemek esetén is hamis értéket ad vissza. Alternatívaként felmerül, hogy először fordítsd meg a tömböt, majd az **each()**-függvényt használd a tömb elemein való előrehaladáshoz, mivel az **eachQ** az üres

elemek esetén nem hamis értéket ad vissza. Bár a tömb megfordítása időigényes, elkerüli az üres elemek problémáját. A következő kód megfordítja a tömböt, és azután a fordított tömbön lép végig. Ehhez az array_reverse()-függvényt használja, amely az átfordítást végzi, és a whileQ-függvényt, amely végigmegy a megfordított tömb elemein.

```
íreversed = array_reverse($mmeral) ;
while(list($k, $v) = each
($reversed))
    {
    prínt ("<br>" . $v) ;
```

A tömb elemein a for()-ciklus segítségével is végigmehetsz, amely a tömbön alkalmazott count()-függvényen alapul, azonban a count()-függvény értéke elromlik, ha a ciklus belsejében valamit csinálsz, hozzáadsz vagy elveszel tömbelemeket. A prev()-függvény a tömbelemeken való visszafelé haladásnak a leggyorsabb módja, ha biztosítani tudod, hogy nincsenek üres elemek. Gyorsaság szempontjából a for()-függvény a következő választás, ha a cikluson belül nem adsz hozzá vagy nem veszel el elemeket. Az tömbelemeken való lépege-tésnek a legbiztonságosabb módja az each{)-függvény, mivel az each() üres elemekkel, törlés és hozzáadás esetén is működik.

Ciklus a többdimenziós tömbökön

Ez a példa egy különböző termékeket tartalmazó tömb esetén összeadja az egyes tételek súlyát. Egyes termékek más termékekből állnak össze, amelyek ismét más termékeket alkothatnak, így a tömb többdimenziós. Minthogy nem tudom előre, hogy hány dimenziós, és a dimenziók történetesen nem szimmetrikusak, a tömbelemeken való végiglépegetéshez nem használhatom ciklusok egyszerűen egymásba ágyazott halmazát. Szerencsére a PHP-ben egyszerű eljárás áll rendelkezésre, amely még a legkülönösebb tömbstruktúrák kezelé-

lkai

sere is a mas.

Először kezdjük azzal, hogy néhány példaterméket és azok súlyát egy egydimenziós tömbbe rakjuk, amelynek neve Sweight. A példa csak a súlyokat használja, azonban egy valódi raktárrendszer mennyiségeket és más értékeket is tartalmazna:

```
$weight["nut"]    =   50;
$weight["bolt"]   =   550;
$weight["flange"]   =   300;
$weight["gasket"]   =   30;
```

A $weight>tömbben néhány terméket más termékekből szerelnek össze, és ahelyett, hogy a súlyuk szerepelne, az a lista áll rendelkezésre, hogy milyen termékekből állnak össze. Ilyen például a kütyü, amely csavaranyából, karimából és csavarból készül.

```
$weight["widget"]    =    array($weight["nut"],
    $weight["flange"], $weight ["bolt"]) ;
```

A ketyerét gyártó cég zászlóshajója a WidMAX, amelyet a ketyeréből és további csavarokból szerelnek össze. A következő példa a Sweight tömbben mutatja a WidMAX bejegyzést, amely a ketyerére való utalást is tartalmazza, amely utóbbi más termékekből áll össze:

```
$weight["WidMAX"]    =    array($weight["widget"],
    $weight["widget"], $weight["bolt"],    $weight["bolt"],
    $weight["bolt"]);
```

Tegyük fel, hogy van egy teherautód, és megkérlek arra, hogy az alaszkabeli Kotlikból a texasi Brownsville-be szállítsd át az árukészletemet. Hogyan számolod ki az elszállítandó teljes súlyt? Kezdd az alábbi egyszerű függvénnyel, amely összeadja az egy egydimenziós tömbben lévő számokat (ahhoz, hogy a Sweight**["WidMAX"]** adattal dolgozni tudjál, a kódban szükség van egy összeadásra is, amint azt a következő bekezdésekből láthatod):

```
function    add_weights($weights)
    {
    $x   =   0;
    while (üst ($k, $v) = each ($weights) }
        {
        $x += $v;
        }
    return($x);
```

Ez az egyszerű tömb megkapja a $v-ből az értékeket, és hozzáadja a $x-hez. A többdimenziós tömbnél meg kell nézned, hogy a $v tömb-e, és valami különlegeset tenni, ha az. A legegyszerűbb dolog meghívni az add_weights()-függvényt a $v-hez, hogy a tömb szerkezetében egy szinttel lejjebb ismétlődjön a folyamat. A következő példában egy egydimenziós függvény korlátlan dimenziók kezelésére való kiterjesztését láthatod:

```
function    add_weights($weights)
    {
    $x   =   0;
    while (Üst ($k,    $v)   =   each
        ($weights) ) {
        if(is   array($v)) { $x   +=
            add  weights($v) ;
            }
        else
            {
        $x   +=   $v; } }
    return ($x) ;
    }
```

Az előző $x + = $v; felcserélődött az **if(is_array($v))** kezdetű kódra, ami meghívja az add_weights()-függvényt, ha $v tömb, és ugyanazt a $x += $v;-t használja, ha $v nem tömb. Az add_weights() a tömb részhalmazával kezd, ami maga is tömb, és visszaadja a részhalmaz súlyát. Ha a részhalmaz elemei tömbök, akkor ezen elemek feldolgozása újra meghívja az add_weights()-függvényt.

Ötlet: Figyelembe kell venni az ilyen újrameghívásoknál a végtelen ciklus lehetőségét. Ha csak egy kis gépelési hibát is ejtesz a kód írásakor, akkor lehet, hogy mindig ugyanarra a pontra visz vissza a ciklus addig, amíg a php.ini-ben beállított időkorlátot el nem éri. A hi baüzenet nem fogja megmondani, hol tartott a kódban, amikor kifutott az időből. Az ilyen kódok teszteléséhez futtasd le egyszer az újrameghívót a $x+=add_weights($v) sor kiikta tásával.

4. fejezet

Bankkártyák

Áttekintés

Némely vásárló paranoiássá válik attól, ha egy weboldalon a hitelkártyaszámát kérik, viszont ugyanezt a számot bármiféle óvintézkedés nélkül nyugodt körülmények között beolvassa a telefonba. Más vásárlók a szállítási feltételek és kikötések elolvasása nélkül megadják ezt, majd panaszkodnak, amikor egy nem várt tétel bukkan fel a számlakivonaton. Más weboldalak tovább rontanak a helyzeten azzal, hogy megtévesztően mutatják be az árut azáltal, hogy egy linkkel a gyártó honlapjára mutatnak, de más terméket vagy más felszereltséggel szállítanak, mint az a linkelt oldalon látható volt. A te dolgod a weboldaladat láthatóan biztonságosabbá és megbízhatóbbá tenni a versenytársaid honlapjainál.

A paranoiásoknak jó okuk van azt hinni, hogy a weboldalak azért vannak, hogy megko-passzák a vásárlókat. Nehéz lesz olyannak bármit is eladnod, aki egyszer már ráfázott az online vásárlásra, így fontold meg a szállítás utáni fizetést. Ha zöldséget, gyümölcsöt szállítasz, a vevő otthon lesz a kézbesítésnél, így nyugodtan fizethetnek kézbesítéskor.

Fontold meg az alábbi, a valós világból vett problémákat:

1. Vásároltam bort egy online-aukción, és megadtam a hitelkártyaszámomat, hogy gyor sabban kézbesítsék az árut, de a licitem több mint kétszeresével terhelték meg a számlámat. Ez a számlakivonat egy hónappal későbbi érkezéséig nem derült ki. így aztán nem fogok olyan online-aukción vásárolni, ahol a kézbesítés előtt kérik a hitel kártyaszámot, és az újabb online-aukciók üzemeltetőinek nagyon nehéz lesz meggyőzni engem az aukciós weboldaluk biztonságáról. Mi ment rosszul az online-aukción? A weboldalt a fejlesztői úgy építették, hogy az automatikus licitemelő funk ció segítségével az eladónak a lehető legmagasabb árat biztosítsák. Majd a fejlesztők alapértelmezésben beállították az automatikus emelést, és amikor az oldalt megláto gatjuk, ez még akkor is bekapcsolódik, ha korábban kikapcsoltuk is. Nem tudom, hogy a fejlesztők figyelmetlenek, kretének, őrülten kapzsiak, vagy csak gonoszak vol tak. A vége az lett, hogy az automatikus bid bekapcsolódott, hogy többet fizettem, mint kellett volna, hogy soha nem fogok olyan aukcióra menni, ahol van automatikus licit, és hogy szólni fogok másoknak, hogy kerüljék el ezeket a weboldalakat.
2. Egy házhoz szállító cég csak egyszer kéri el a hitelkártyaszámot, majd mindig azt ter heli, így nagyon nehézzé teszi a kártyaszám megváltoztatását. Visszavontam a régi kártyaszámom és megadtam az újat, de a régire terheltek. Ezen a cégen keresztül nem fogok vásárolni, mert nem bízom meg abban, ahogy a számlázást lebonyolítják.
3. Olyan szállítótól veszem a lemezmeghajtóimat és más tételeket, amely mindenegyes tételnél felsorolja, hogy hány van még raktáron belőle, hányat hívtak vissza, és mikor van a következő szállítás. Amikor valami sürgősen kell, csak olyat veszek meg, ami raktáron van. Egy nap egy hasonló kinézésű honlapról vettem egy mobiltelefonhoz való akkumulátort. Az oldal azt sugallta, hogy a szállító nagy és megbízható. Volt egy kis probléma az online rendeléssel, így telefonon rendeltem. Pár hét várakozás és jó néhány telefon után kiderült, hogy a cég online értékesítési részlege egy fiatal srác egy telefon mellett, és hogy a cégnek nincs raktára. A bizalomra nem méltó szállító

végül is talált valakit, aki nagyon olcsón leszállítja neki az akkumulátort, miközben én teljes árat fizettem, és az akkumulátor gyorsan tönkrement, mert öreg volt, valószínűleg használt vagy bemutató darab. Ezek után csak olyan szállítótól vásárolok, akinek van raktára, a rendelés beérkezésekor lefoglalja a kért cikket, és az árut eredeti csomagolásban szállítja le.

Ha még mindig elég erőt érzel magadban a problémák legyűrésére, olvass tovább. Túljuthatsz a szoftveres problémákon, az ügyfélszolgálat sok problémáján, és leküzdheted az online vásárlásban már csalódottak félelmét. Elfogadhatsz hitelkártyát a rendelésekhez. Csak a megfelelő irányba kell haladnod, és minden lépést alaposan tesztelned kell.

M

Kereskedői bankszámla

Ahhoz, hogy hitelkártyát fogadj el, kereskedői bankszámlára van szükséged. Ez a bankszámla teszi lehetővé, hogy a banktól vagy más cégektől, akik a hitelkártyapanaszokat feldolgozzák, visszakérd a pénzed. Számlanyitási díjat, havi minimumdíjat, minimális tranzakciós költséget és a forgalom után százalékot kell nekik fizetned. Kis ügyfelek forgalmuk 6%-át is kifizethetik, nagy ügyfelek pedig akár csak 0,2%-ot (ezt hívják kereskedői díjnak, avagy egyszerűen díjnak). A tranzakciós költségek 10 és 20 cent között mozognak. A havi díj papír alapú számla esetén körülbelül 10$, de elektronikus számla esetén, amellyel az ügyintézései akár interneten vagy telefonon keresztül is történhet, esetleg 35$-ra is nőhet.

Csak késedelemmel fizetnek, és a fizetés három hónapig visszavonható. Ellenőrizd a részleteket a bankodnál, és nézz jól körül, hogy az értékesítésed volumenéhez a legmegfelelőbb üzletet kössd.

Ha elfogadsz online fizetést és a következő napon szállítasz, akkor a problémáidat csökkentheted azzal, hogy a fizetés jóváírását a szállítás napjára halasztód. Gondolj csak annak a szállítónak a problémájára, aminek több hétbe telt egy akkumulátort találnia. Ha online fogadta volna el a fizetést, és én az első késedelem után lemondtam volna a rendelést, a szállítónak vissza kellett volna térítenie a pénzemet és visszatérítési díjat kellett volna fizetnie. Ha a szállítónak sokszor kell visszatérítenie, a kereskedői díja nőni fog.

Egy ausztrál bolt hagyományos Visa/MasterCard kereskedői bankszámlát használ a helyi értékesítéshez, de csak American Expresst fogad el nemzetközi értékesítés esetén, mert az American Express nagyobb biztonságot nyújt a hitelkártya-azonosítás terén, mint egy helyi bank. Drága árucikkek kereskedői valamilyen módon nyomon követik az értékesítést, hogy egy árucikk addigi eladásait megnézhessék az adás-vétel teljesítése előtt. Nyomon követheted vásárló szerint, ország szerint, és árucikk szerint is. A lopott hitelkártyával fizetők gyakran gyorsan újraértékesíthető árukat vesznek, mint például Sony Walkman, és több darabot rendelnek. Vigyázz, ha kiskereskedelmi oldalad van és egy új vásárló 10 Walkmant rendel. Egy online értékesítő elmondta, hogy sok megrendelést kap indonéziai turistahelyek közeléből, bár ő nem ad el semmi olyat, amit egy turista venne vakáción. A Visa bejelentette, hogy az amerikai pornóoldalak sok országból nem fogadnak el Visa kártyát, mert számtalanszor előfordult, hogy az emberek feliratkoztak egy hónap nézelődésre, majd vitatták a számlakivonatnak ezt az elemét.

Sok ellenérv van az online fizetés ellen, de ez nem zárja ki a hitelkártya-használatot. Ha az oldalad adományokat fogad alapítványok számára, semmit nem vesztesz az online fizetéssel. Amikor online szoftvert adsz el, az azonnali értékesítés lehetősége fontosabb, mint az, hogy egy távoli országból esetleg nem fizetnek. Hozzáadhatsz egy kódot, hogy észleld a gyanús megrendeléseket, és késleltesd a szállítást, hogy a rendelést ellenőrizd. Ha egy online zöldséges 1,000$ értékű chipszről és kóláról kap megrendelést, késleltetheti a szállítást, és a rendelés visszaigazolása miatt felhívhatja a vásárlót.

Gyerekek

Ha az oldalad vonzza a gyerekeket, gyerekek fognak a szülők kártyáival vásárolni, és a szülők vitatni fogják a számlát. A problémák egy részét megelőzheted azzal, hogy az új megrendelők szállítását addig késlelteted, amíg nem ellenőrizted őket, nagyobb megrendeléseknél például felhívhatod a vevőt.

Számlázási név

Ha az oldalad címe **custom-software-delevopment.com,** viszont a vevő számlakivonatán a internationalservices.com jelenik meg, néhány vevő vitatni fogja a számlát, hisz nem tudja, mi az a **internationalservices.com.**

A tanulság az, hogy a vevőnek tudnia kell, mire számíthat az weboldaladtól és a számlakivonatától, hogy a problémák felderítéséhez az értékesítést nyomon kell követned, és hogy kérd a kereskedői számládat vezető pénzintézet segítségét a problémák megelőzéséhez.

Biztonság

A tranzakcióknál a minimális biztonsági elvárás a 128 bites Secure Socket Layer (SSL). A szervert biztonságosan kell tárolnod, nehogy a hitelkártyára vonatkozó információkat ellopják. Az SSL-ről további információt a nyílt forráskódú SSL-weboldalon (www.openssl.org/), az SSL Apache-webszerveren való megvalósításáról pedig a **www.openssl.org/** címen találsz.

Egyes kereskedői bankszámla szolgáltatók olyan jóváírási rendszert használnak, amelyben a hitelkártyára vonatkozó információkat a webszervered soha nem tárolja el. Az információt csak átadja a kereskedői bankszámla szolgáltatónak, azonban a webszervereden jelentkező rést a tömegtájékoztatás nem fogja ártalmatlannak minősíteni. Ha a webszerveredet akár csak egyszer is megemlíti a sajtó, akkor a nevedet vádak és hitelkártya-veszteségekre vonatkozó utalások lengik majd körül.

Szerver

Ha a webszervered RedHat vagy Mandrake Linux, minden egyes változtatás után futtasd le a Bastille biztonsági utasítássorozatot. A többi operációs rendszer esetén találd meg az ezzel ekvivalens programot. A Bastille **(www.bastille-Iinux.org/)** olyan utasítássorozat,

amely végigfut a webszervereden és az operációs rendszereden kikapcsolja azokat az opciókat, amelyek veszélyeztetik a biztonságot. A Linux egyes változatai a biztonság magasabb fokát biztosítják alapértelmezésben, és beépített biztonsági beállításai miatt sokan inkább a freeBSD-rendszert választják. A webszerver-szoftvered az első védelmi vonalad. Ha valaki a webszervereden keresztül betör, akkor az operációs rendszernek kell megakadályoznia, hogy a betörő átvegye a teljes szerver felett az irányítást.

A webszerverre tűzfalat is telepíthetsz a szerver ellen intézett támadások bizonyos típusainak kiszűrésére. Miközben gépelek, a hálózati routeremen a fények villognak code-red-tá-madásokat jelezve, de a router tűzfala a támadó csomagokat visszadobja anélkül, hogy egyetlen támadás is elérné a lokális hálózatomat és szeveremet. A code-red-féreg a Microsoft IIS rendszerét támadja meg és az HS-szervereket használja támaszpontként más szerverek megtámadásához. A féreg véletlenül választott IP-címeket támad meg, és a code-red hármas verziója már elég HS-szerveren megtalálható ahhoz, hogy minden második másodpercben támadást intézzen a hálózatom ellen.

Hash-elés

A *zagyvalék (hash)* az, amit akkor csinálok, ha a főzés során az étel katasztrofálisan sikerül. Ugyanakkor a hash-nek a számítástechnikában speciális jelentései vannak, és ezek közül nem mindegyik jelenti azt, hogy a kódom arra a sorsra jut, mint a főztöm. A hash olyan kód, rendszerint szám, amely valamely sztringből jön létre, és hosszú sztringeknek rövid számsorozattal való azonosítására lehet használni. Ha az adatbázisod recepteket tartalmaz, akkor a hash-t a többi hozzávalók alapján is elkészítheted, például 1 a burgonya helyett, 2 a só helyett, és a 9 jelzi a sütést mint a főzés egy módját. Ekkor a tócsni (hagymás burgonya olajban sütve, rendszerint sózott) hash-száma 129 lesz, ugyanakkor a sómentes tócsni vegyítése 19 lenne. Az adatbázisod a sós tócsnira vonatkozó recepteket a 129-es rekordban tárolja, az egészségesebb változatot pedig a 19-es rekordban. Amikor valaki olyan recepteket keres, amelyekben burgonya, sütés, só nélkül a fő jellemzők, akkor azonnal kiszámolhatod, hogy a hash értéke 19 és az adatbázisban való keresés nélkül előveheted a 19-es rekordot. A kriptográfiában és más területeken különböző hash-számítási módokat használnak. Ezek közül némelyik kétirányú - az eredeti érték meghatározható a hash-értékből — ezek azonban a legtöbb egyirányú - az eredeti érték nem határozható meg a hash-értékbŐl.

A PHP-ban mhash_-függvények állnak rendelkezésre, ezek olyan különböző értékeket számítanak ki, amelyeket kissé pontatlanul *hash-értékeknek* fogunk nevezni. Az **mhash_**-függvények működését a Gyors megoldások „Adatok hash-elése" része mutatja be.

A hash-eket ellenőrzésre és visszaigazolásra, nem pedig titkosításra használják. A hash-ek általában egyirányúak, tehát nem tudod vissza-hash-elni az eredeti információt. A hash-ek lehetőséget adnak neked arra, hogy az adatokat visszaigazold anélkül, hogy az adat két teljes másolatát küldd el. Az általad használt hash típusa a követelményektől függ. A CRC-ket arra tervezték, hogy egy fájl vagy rekord mindegyik bitjét ellenőrizzed, ugyanakkor más hash-értékek esetleg csak a sztring első részét használják. A CRC32 (amt a 8. fejezet ír le) 4 milliárd értéket tárolhat, tehát felhasználható egy fájlnak ezrek közül történő azonosí-

tására. (Nekem olyan könyvtáram van, amely 49 000 hasonló méretű és tartalmú fájlt tartalmaz, és mégis mindegyiknek egyedi CRC-száma van.) Ha el akarsz egy sztringet, például egy jelszót rejteni, majd később elő akarod venni, akkor a hash helyett titkosításra van szükséged.

- A *CRC32* egy 32 bites Ciklikus Redundancia Ellenőrzés, amely a lemez hibáját deríti fel, amikor hálózaton továbbítanak adatot, és fájlokat (például a ZIP fájlokat) ellen őriz. A mhash_-függvény támogatja **MHASH_CRC32-t** (Etherneten használják) és a MHASH_CRC32B-t (ZlP-fájlokra használják). A CRC32 értékek megjeleníthetők hexadecimális és egész számként is. A PHP néha szétvagdalja az egész szám változa tot, mert 32 bites előjeles egész számoknak tekinti őket, viszont azok valójában 32 bi tes előjel nélküli egész számok. Ha a PHP negatív előjelű egész számot jelenít meg, akkor 4 milliárdot kell hozzáadnod, hogy a helyes értéket kapd meg.
- Az *MD5* algoritmust Ron Rivest találta fel az RSA-nál, és az RFC 1321-ben le van ír va **(www.faqs/rfcs/rfcl321.html).** Az MD5-öt használják a jelszó verziójának átadásá ra a böngészőből a szervernek, és lemezen való tárolására. Az MD5 nem titkosító, te hát nem kell dekódolni az MD5-öt, és nem lehet megszerezni a jelszavakat az MD5-től. Az MD5 úgy működik, hogy elmenti a jelszót MD5-verzióban, majd átalakítja a bejövő jelszavakat MD5-be, és összehasonlítja az MD5-értékeket, hogy ellenőrizze a jelszót. Ha egy vásárló elfelejti a jelszavát, akkor azt te sem tudod neki megmondani; vissza kell a jelszót állítanod alapbeállításra, majd a vásárlónak kell megváltoztatnia.
- *MD4* - Az MD4-re tekints úgy, mint a titkosítás Apolló-13-jára, olyan hősies próbál kozásra, ami elbukott. Használd inkább az MD5-öt.
- *SHAl* — A SHA-algoritmust az NIST (Országos Szabvány- és Technológiai Intézet [www.nist.org/]) használja a Digitális Aláírás Szabványaként. A **mhash** a **MHASH_SHA1**-algoritmust hívja meg.

Megjegyzés: Egy érdekes történet: a NIST-F1, a NIST egyik cézium alapú órája másodpercenként 9 billiót rezeg, és 1 másodpercet téved 20 millió év alatt. Ez igazából elavult, mert a NIST új merkúrium alapú órája 1 kvadrilliót rezed másodpercenként és 1 másodpercet téved 20 000 millió év alatt. A webszervereden tényleg a jó idő van beállítva?

- *HAVAL* - A HAVAL az MD5 változata. Többféle hosszúságú értéket enged meg, és az mhash-ben definiálja, mint **MHASH_HAVAL256, MHASHHAVAL224, MHASH_HAVAL192** és **MHASH_HAVAL160.**
- *RIPEMD-160* - A RIPEMD-160 az MD4, MD5 és a RIPEMD 160 bites helyettesítője, amit Hans Dobbertin, Antoon Bosselaers és Bárt Preneel terveztek. A RIPEMD-et az EU RIPE projektjének részeként fejlesztették ki. Az **mhash** ezt **MAHSHRIPEMD160** algoritmusnak hívja.
- *Tiger - A* Tiger, amit Eli Biham és Ross Anderson terveztek, állítólag nagyon gyors 64 bites számítógépeken. Az mhash-ben ez az algoritmus **MHASH_TIGER192, MHASHTIGER160** és **MHASH_TIGER128-ként** van definiálva.
- *GOST* - A GOST az 256 bites orosz digitális aláírási szabvány. Az **mhash** ezt az algo ritmust **MHASH_GOST-nak** hívja.

Titkosítás

A titkosítás félig tudomány, félig művészet. A tudomány ellenőrzi, hogy egy titkosítás működik-e vagy sem. Viszont a titkosítás megfelelő megvalósítása a weboldaladon az már művészet. Valami olyanra van szükséged, ami használható abból a szempontból, hogy egy gazdaságos webszerveren is elfogadható időn belül lehet elvégezni a titkosítást, mégis feltörhetetlen még egy olyan hackernek is, aki akár egy cégnél, akár egy egyetemen pár ezer hálózatba kötött géphez is hozzáfér. A hacker mondhatja azt, hogy törvényes kutatást végez a cég megbízásából, de nem említi, hogy fel akarja törni a helyi bank jelszavát vagy egy koncertjegyeket árusító oldalt.

A PHP tartalmazza az mcrypt-függvényt, amely a **http://mcrypt.hellug.gr/** oldalról kéri le az mcrypt-szoftvert, így lehetővé teszi a BLOWFISH, TWOFISH, DES, TripleDES, 3-WAY, SAFER, LOKI97, GOST, RC2, RC6, MARS, IDEA, RIJNDAEL, SERPENT, CAST, ARCFOUR és WAKE titkosító algoritmusok használatát.

Az mcrypt telepítése

Az mcrypt installálásához kövesd az alábbi lépéseket.

Unix

Ha Unix alatt installálod:

1. Töltsd le a libmcrypt-x.x.tar.gz-fájlt az **ftp://mcrypt.hellug.gr/pub/mcrypt/libmcrypt/** oldalról.
2. Kövesd a telepítési utasításokat.
3. Fordítsd be a libmcrypt-et a következő opcióval: — **dísable-posix-threads.** Superuserként konfiguráld a libmcrypt-et, hogy elkerüld a jogosultságból fakadó problémákat.
4. Fordítsd be a PHP-t a következő opcióval: **—with-mcrypt.**

Windows és NT

A PHP 4.0.6 Win32 bináris állományai nem tartalmazzák a php_mcrypt.dll-t. Remélem, egy későbbi kiadása már tartalmazni fogja.

mcrypt-függvények

A következő rész néhány példával illusztrálva sorolja fel az mcrypt_-függvényeket.

mcrypt_get_cipher_name()

A titkosító algoritmus (cipher) nevét az alábbi kód használatával kaphatod meg egy algoritmus azonosítójából. Ha az azonosító hibás, a függvény hamisat ad vissza.

```
print("<br>" . mcrypt_get_cipher_name(MCRYPT_DES)) ;
```

A példa eredménye:

```
DES
```

mcrypt_list_algorithms()

Az alábbi kód felsorolja az mcrypt könyvtárban található összes titkosító algoritmust. Megadhatsz egy opcionális könyvtárnevet, hogy megtalálja a könyvtárban található összes titkosító algoritmust.

```
íarray = mcrypt_üst_algorithms () ;
while (üst ($k, $v) = each ( $array) )
    í
    print("<br>" . $v);
```

mcrypt_list_modes()

Ez a függvény felsorolja az mcrypt könyvtárban fellelhető összes módozatot, így megnézheted, hogy az aktuális könyvtárban megvan-e az a módozat, amit akarsz. További könyvtárne-vet is megadhatsz, hogy a megadott könyvtárban lévő összes módozatot felleld. A titkosítási módozatok tartalmazzák a CBC-t (cipher block chaining, titkosító algoritmus blokk lánco-lás), CFB-t (cipher feedback, titkosító algoritmus visszacsatolás), ECB-t (electronic codebook, elektronikus kódkönyv), OFB-t (output feedback with 8bit data, output visszacsatolás 8 bites adattal), nOFB-t (output feedback with nbit data, output visszacsatolás n-bites adattal), és STREAM-et (A PHP jelenleg még nem támogatja a stream módot). Az alábbi kód felsorolja az **mcrypt_list_modes()** által visszaadott módokat:

```
Smodes = mcrypt list_modes();
while (üst ($k, $v) = each ($modes)
)
    (
    print("<br>" . $v);
```

mcrypt_get_block_size ()

Ez a program megmondja egy adott algoritmus blokkméretét. A PHP jelenleg sztringek titkosítását támogatja, és nem stream-ek vagy más bonyolultabb dolgokét, így a blokkméretre csak azért van szükséged, hogy olyan dolgokat beállíthass, mint például a táblázatok mezőméretei:

```
print("<br>" . mcrypt_get_block_size(MCRYPT_DES, cbc) ) ;
```

mcrypt_get_key_size()

A mcrypt_get_key_sizeQ-parancsot használd, ha egy speciális titkosító eljárás kulcsának méretére van szükséged:

```
print ("<br>" . mcrypt_get_key_size(MCRYPT_DES,cbc) ) ;
```

mcrypt_module_open()

Az **mcrypt_module_open()** használat céljából megnyit egy titkosító modult. Ezt az **mcrypt_create_iv()-** vagy a titkosító és a visszafejtő függvény előtt le kell futtatni. Az eredmény olyan forrásazonosító lesz, amelyet más függvények használnak. A modulnév annak a modulnak a neve, amely a speciális titkosító eljárást megvalósítja. A Blowfisf-algorit-mus modulja mint a "blowfish"-sztring is bevihető, vagy az **MCRYPT_BLOWFISH** előre

definiált név használatával. A modulok a php.ini **mcrypt.algorithms_dir** paramétere által kijelölt könyvtárban vannak. Alapértelmezésben ez az /usr/local/lib/libmcrypt. Ha már nyitva van, akkor bezárhatod a modult az **mcrypt_module_close()** vagy **mcrypt_generic_end()** alkalmazásával. Az **mcrypt_module_close()** nincsen a jelenlegi dokumentációban, kihagyható. A következő példa megnyitja a DES-t, a php.im-ben alapértelmezésként beállított könyvtárat, valamint a CBC módot használja, és a CBC-t az **/usr/lib/mcrypt-modes** könyvtárból veszi:

```
$cipher = mcrypt_module_open(MCRYPT_DES, "",
    MCRYPT_MODE_CBC,"/usr/lib/mcrypt-modes") ;
```

mcrypt_createjv()

A titkosítás valamilyen formájának megkezdése előtt egy *inicializáló vektorra* (IV) van szükséged. A **mcrypt_create_iv()** létrehozza ezt. Add meg az IV méretét és az IV-hez felhasználandó véletlen forrását mint alább:

```
srand {(double) microtime() *
1000000); $iv = mcrypt_create_iv(32,
MCRYPT_RAND);
```

Az **MCRYPT_RAND** előtt használd az srandQ-parancsot (amelyet a 2. fejezetben mutattam be). Az első paraméter az mcrypt_get_block_size()-parancs által visszaadott blokkméret. A második paraméter lehet az **MCRYPT_DEV_RANDOM,** amely a /dev/random-ból olvas be adatot, vagy az **MCRYPT_DEV_URANDOM,** amely a /dev/urandom-ból.

mcrypt_get_iv_size()

A mcrypt_get_iv_size() az algoritmushoz tartozó IV méretét nyeri ki:

```
print ("<br>" . mcrypt_get_iv_size(MCRYPT_DES, cbc)};
```

mcrypt_cbc()

A mcrypt_cbc() a titkosító algoritmus blokk láncolás módszer szerint tikosít, amely fájlok esetén megfelelő:

```
$key = "Do not teli anyone the contents of this string";
$string = mcrypt_cbc(MCRYPT_DES, $key, $file_or_string, MCRYPT_ENCRYPT,_ 5iv)
    ;
```

Az első paraméter a titkosító, a második a kulcs. Mivel a kulcs titkos kell legyen, a kulcsot nem mutatja a példa. A kulcsnak a titkosítóhoz kell alkalmazkodnia. Némely eljárás erősebb lesz, ha a kulcs-sztring hosszú, ugyanakkor más eljárások csak korlátozott sztringhosszúsá-got használnak. A következő paraméter a titkosítandó vagy visszafejtendő adat, amely vagy sztring vagy pedig egy fájl, amelyet a **file()**-utasítás segítségével sztringbe olvastál be. A negyedik paraméter a mód, titkosítás vagy visszafejtés. Az opcionális ötödik paraméter egy IV lehet, amelyet az **mcrypt_create_iv()** ad **át. Az mcrypt_creat_Ív()** első paramétere a CBC számára az **mcrypt_get_block_size**() által visszaadott blokkméret kell hogy legyen.

mcrypt_cfb()

A **mcrypt_cfb()** ugyanazokat a paramétereket fogadja el, mint az **mcrypt_cbc().** A titkosító algoritmus visszacsatolás módban titkosít, amely a legjobb választás a bájtonként történő titkosítás esetén. A következő példa egy sztringet fejt vissza:

```
$string = mcrypt_cfb(MCRYPT_DES, $key, $file_or_string,
MCRYPT_DECRYPT);
```

mcrypt_ecb()

A **mcrypt_ecb()** ugyanazokat a paramétereket fogadja el, mint az **mcrypt_cbc(),** és elektronikus kódkönyv módban titkosít, amely a legjobb választás rövid sztringek, így például kulcsok esetén. A következő példa egy sztringet fejt vissza:

```
$string = mcrypt_ecb(MCRYPT_DES, $key, $file_or_string,
MCRYPT_DECRYPT);
```

*

mcrypt_ofb()

A **mcrypt_ofb()** ugyanazokat a paramétereket fogadja el, mint **mcrypt_cbc(),** és output visszacsatolás módban titkosít. Ez egy 8 bites eljárás, amely nem biztonságos és nem javasolt. A következő példa egy sztringet titkosít:

```
$string = mcrypt_ofb(MCRYPT_DES, $key, $file_or_string,
MCRYPT_ENCRYPT);
```

mcrypt_encrypt()

A **mcrypt_encrypt{)** majdnem ugyanazokat a paramétereket fogadja eí, mint **mcrypt_cbc(),** és a negyedik paraméter által megadott módban titkosít. Az opcionális ötödik paraméter IV értéket szolgáltat azon eljárások számára, amelyek igénylik. A következő példa **egy** sztringet titkosít:

```
$string = mcrypt_encrypt(MCRYPT_DES, $key,
    $file_or_string, MCRYPT_MODE_CBC);
```

mcrypt_decrypt()

A **mcrypt_decrypt()** ugyanazokat a paramétereket fogadja el, mint **mcrypt_encrypt(),** és visszafejt a negyedik paraméter által jelzett mód szerint. Az opcionális ötödik paraméter IV értéket szolgáltat azon eljárások számára, amelyek igénylik. A következő példa visszafejt egy sztringet:

```
$string = mcrypt_decrypt(MCRYPT_DES, $key,
    $file_or_string, MCRYPT_MODE_CBC);
```

mcryptjgeneric_init()

A mcrypt_generic_init() a titkosító eljárás forrásazonosítóját fogadja el, amelyet az **mcrypt_module_open()** ad vissza. Ez egy kulcs és egy IV A függvény a titkosításhoz használt összes puffért inicializálja. Ha hiba adódik, a függvény az -1 értéket adja vissza:

```
$int = racrypt_generic_init($cipher, $key, $iv);
```

mcrypt_generic()

A **mcrypt_generic()** a titkosító eljárás forrásazonosítóját fogadja el, amelyet az **mcrypt_module_open()** ad vissza; és egy titkosítandó sztringet. A **mcrypt_generic()** titkosított sztringet adja vissza. Az **mcrypt_generic_init()-parancsot** az **mcrypt_generic()** meghívása előtt kell kiadni:

```
Sencrypted = mcrypt_generic($cipher, $string) ;
```

mdecrypt_generic()

A **mdecrypt_generic()** a titkosító forrás azonosítóját fogadja el, amelyet az **mcrypt_module_open()** ad át, és egy titkosított sztringet. A **mdecrypt_generic()** visszafejtett sztringet adja vissza. Az **mcrypt_generic_init()-parancsot** az **mcrypt_generic()** meghívása előtt kell kiadni:

```
$string = mdecrypt_generic($cipher, $encrypted);
```

mcrypt_generic_end()

A mcrypt_generic_end() bezárja az **mcrypt_generic_init()** által megnyitott puffereket és az **mcrypt_module_open{)** által megnyitott modult.

```
if(!mcrypt_generic_end($cipher)) {
    print("<br>mcrypt_generic_end() failed.");
```

mcrypt_enc_self_test()

A mcrypt_enc_self_test() tesztet futtat le az **mcrypt_module_open()** által megnyitott modulon, és 0 értéket ad vissza, ha az eredmény megfelelő, ha pedig a modul nem megy át a teszten, akkor 1-et:

```
Lí(mcrypt_enc_self_test($cipher))
    {
    print("<br>module failed.");
    }
```

Ha ezt a tesztet használod, akkor azelőtt használd, hogy bármit titkosítottál volna. A PHP 4.0.6 verziójában, az **mcrypt_enc_self_test()** CGI-modulként és hagyományos Apache-modulként működött, azonban Apache/Unix DSO- (Dynamic Shared Object, Dinamikus Megosztott Objektum) modulként nem.

mcrypt_enc_is_block_algorithm_mode()

A **mcrypt_enc_is_block_algorithm_mode()** az aktuális módon futtat le tesztet. 1 értéket ad vissza, ha az eljárás működik a blokk algoritmussal, ha pedig a mód streamekre alkalmas, akkor 0-t.

```
if(mcrypt_enc_is_block_algorithm_mode($cipher))
    {
   print("<br>mode is for block algorithms.");
    }
else
```

```
    print("<br>mode is for stream algorithms.");
```

mcrypt_enc_is_block_algorithm()

A **mcrypt_enc_is_block_algorithm()** az aktuális algoritmuson futtat le tesztet, és 1 értéket ad vissza, ha az algoritmus blokkokkal dolgozik, 0 értéket, ha az algoritmus streamekre alkalmas.

```
if(mcrypt enc is block_algorithm($cipher))

    print("<br>block algorithm.");

else

    print("<br>stream algorithm.");
```

m cryp t_e n cjs_b lo ck_mode ()

A **mcrypt_enc_is_block_mode()** lefuttat egy tesztet a megnyitási módon, és 1 értéket eredményez, ha a mód blokkokat ad vissza, 0 értéket, ha bájtfolyamot.

```
if(mcrypt_enc    is_block_mode($cipher))

print("<br>mode    is    for    blocks.");

else

    print("<br>mode    is    for
    streams."); }
```

mcrypt_enc_get_block_size()

A **mcrypt_enc_get_block_size()** a megnyitott algoritmusban használt blokkméretet adja vissza:

```
$size = mcrypt enc_get_block size($cipher);
```

mcrypt_enc_get_key_size ()

A mcrypt_enc_get_key_size() a megnyitási mód által használt kulcs maximális méretét adja vissza. Azon módok esetében, amelyek változó méretű kulcsot támogatnak, tetszőleges, a maximálisnál nem nagyobb kulcsméret elfogadható:

```
$size    =    mcrypt    enc    get_key    size ($cipher);
```

mcrypt_enc_get_supported_key_sizes()

A mcrypt_enc_get_supported_key_sizes() az aktuális algoritmus által elfogadott kulcsméreteket tartalmazó tömböt adja vissza. Ha a tömb üres, akkor 1-től a maximális méretig minden kulcsméret elfogadott. A következő kód a kulcsméretek listáját nyomtatja ki, ha ez a lista létezik:

```
$size = mcrypt enc get
supported_key_sizes($cipher); while(list($k, $v) =
each($size))
    {
    print("<br>" . $v);
```

mcrypt_enc_get_iv_size ()

A mcrypt_enc_get_iv_size() az aktuális eljárás IV-je méretét adja vissza. Ha a függvény nulla értéket ad vissza, akkor az eljárás nem használ IV-t:

```
$size = mcrypt_enc_get_iv_size($cipher) ;
```

mcrypt_enc_get_algorithms_name()

A mcrypt_enc_get_algorithms_name() az aktuális eljárás nevét adja vissza:

```
$name = mcrypt_enc_get_algorithms_name($cipher) ;
```

mcrypt_enc_get_modes_name()

A mcrypt_enc_get_modes_name() a megnyitási mód nevét adja vissza:

```
$name = mcrypt_enc_get_modes_name($cipher) ;
```

mcrypt_module_self_test()

A mcrypt_module_self_test() egy modult tesztel, és igaz értéket ad vissza sikeres, hamis értéket sikertelen teszt esetén. Az opcionális második paraméter a modult tartalmazó könyvtár nevét fogadja el:

```
if(!mcrypt_module_self_test($module)) {
    print("<br>module " . $module . "
    failed.");
```

mcrypt_module_is_block_algorithm_mode ()

A **mcrypt_module_is_block_algorithrn_mode()** a megadott modulon futtat le tesztet, és 1 értéket ad vissza, ha a modul blokk-algoritmussal dolgozik, 0 értéket, ha a modul stream algoritmus-móddal dolgozik. Az opcionális második paraméter a modult tartalmazó könyvtár nevét fogadja el:

```
if(mcrypt_module is block_algorithm_mode($module))
    {
    print ("<br>block algorithm modes.");
    }

else
    {
    print ( "<br>stream algorithm modes.");
```

mcrypt_module_is_block_algorithm()

A mcrypt_module_is_block_algorithm() a megadott eljáráson futtat le tesztet, és 1 értéket ad vissza, ha az eljárás blokkokkal dolgozik, vagy 0 értéket, ha az eljárás streamekkel

dolgozik. Az opcionális második paraméter az eljárásmodult tartalmazó könyvtár nevét fogadja el:

```
if(mcrypt_module_is_block_algorithm($module))

    print("<br>block.");

else

    print("<br>stream.");
```

mcrypt_module_is_block_mode()

A **mcrypt_module_is_block_mode()** a megadott modulon tesztet futtat le, és 1 értéket ad vissza, ha a modul blokkokat, 0 értéket, ha a modul bájtokat ad vissza. Az opcionális második paraméter a modult tartalmazó könyvtár nevét fogadja el:

```
if{mcrypt_module_is_block_mode($module))

    print("<br>block.");

else

    print ("<br>byte.");
```

mcrypt_module_get_algo_block_size()

A **mcrypt_module_get_algo_block_size()** a megadott eljárás által használt blokk méretét adja vissza. Az opcionális második paraméter az eljárást tartalmazó könyvtár nevét fogadja **el:**

```
$size = mcrypt modulé get algo block size(Smodule);
```

mcrypt_module_get_algo_key_size()

A **mcrypt_module_get_algo_key_size()** a megadott eljárás által támogatott maximális kulcsméretet adja vissza. Az opcionális második paraméter az eljárást tartalmazó könyvtár nevét fogadja el.

```
$size = mcrypt_module_get_algo_key_size($module);
```

mcrypt_module_get_algo_supported_key_sizes()

A mcrypt_module_get_algo_supported_key_sizes() a PHP 4.0.6 utáni verziók által tartalmazott új függvény lesz majd, amely a megadott eljárás által elfogadott kulcsméretek tömbjét adja vissza. Ha a tömb üres, akkor 1-től a maximális méretig minden kulcsméret elfogadott. Az opcionális második paraméter az eljárást tartalmazó könyvtár nevét fogadja el. A következő kód a kulcsméretek listáját nyomtatja ki, ha ilyen lista létezik:

```
$size = mcrypt_module_get_algo_supported_key_sizes($module);
while {üst ($k, $v) = each(Ssíze) )
    {                                                    ' .
    print("<br>" . $v);
    }
```

Elektronikus fizetési szoftverek

Ha belevágsz az online fizetés lebonyolításába, szükséged lesz egy szoftverre, amelynek segítségével a kereskedői számlád szolgáltatójával kommunikálhatsz. Ezt a szoftvert adhatja **a** kereskedelmi számlád szolgáltatója, az internet-szolgáltatód vagy harmadik fél. Itt van néhány ezek közül, amelyeket a PHP függvényei közvetlenül támogatnak.

CyberCash

A CyberCash jelenleg a VeriSign tulajdonában van (www.verisign.com), és a CyberCash szoftvert eladták a First Data Merchant Services-nek **(www.firstdata.com/).** (A First Data weboldala nem PHP-t, hanem JSB-t használ, miért is akarnál tehát velük üzletelni?)

A CyberCash Unix alá való installálásához fordítsd be a PHP-t a — with-cybercash-sel. A CyberCash Windows NT4 vagy Windows 2000 alatti telepítéséhez az alábbiakat kell tenned:

1. A c:/Program Files/php/extensions könyvtárból másold át a php_cybercash.dll-t a c:/winnt/system32-be.
2. Változtass meg egy sort a c:/wínnt/php.ini-ben a **;extension** = **php_cybercash.dll-ről extension = php_cybercash.dll-re.**
3. Indítsd újra a webszerveredet (ha a PHP modulként fut).

A Windows más verzióiban a c:/winnt/system32 helyett a c:/windows/system-be másold.

A Payflow a CyberCash egy alternatívája. A Payflow is a VeriSign tulajdonában van, tehát ugyanazé a cégé, mint a CyberCash, így valószínűleg a CyberCash meg fog szűnni, vagy **a** forráskódját kicserélik a Payflow-ra.

A CyberCash használatához a CyberCash weboldalán regisztrálnod kell magad, majd egy felhasználói azonosítót, jelszót, kereskedői azonosítót és titkosító kulcsot kapsz. Ha egy fizetést fel akarsz dolgozni, titkosítsd az üzenetet a CyberCash számára a **cybercash_encr()** használatával, kódold a titkosított üzenetet **a** cybercash_base64_encode()-dal, majd küld az üzenetet a CyberCash-nek az fopen()-nel, illetve egy CGI parancsállománnyal. Az **fopen()** a tranzakció eredményét adja vissza. Az eredményt fel kell osztanod, majd a cybercash_base64_decode()-dal dekódolnod, és a **cybercash_decr()-tel kell visszafejtened.**

A CyberCash dokumentációja a **www.cybercash.com/cashregister/support/docs/** oldalon található. Könnyebbé teheted az életedet, ha a 17. fejezetben utánanézel a objektumoknak, majd letöltőd a Nathan Cassano's CyberClasst a **www.cjhunter.com/~nathan/ /class.cyberclass.txt** oldalról. Az **fopen()** leírása **a** 8. fejezetben található.

cybercash_encr()

A **cybercash_encr()** háromszoros DES titkosítást használ. Az üzenet formátumával kapcsolatban olvasd el a CyberCash dokumentációját. A kereskedői kulcs **a** kereskedői azonosítóval fog megérkezni. A session-kulcs egy session-önként egyedire állítható azonosító. Lehet például egy sztring, ami a session azonosítóját és az aktuális dátumot tartalmazza:

```
$session_key = session_id() . " " . date("Y-m-d H:i:s"); $encrypted =
cybercash_encr($merchant_key, $session_key, $rnessage) if
($encrypted["errcode"] === falsé)
    {
   print("<br>output buffer: " . $encrypted["outbuff"] .
       "<br>output length: " . $encrypted["outLth"] .
       "<br>mac buffer: " . $encrypted["macbuff" ] ) ;
    }

else
    {
   print("<br>error code: " . $encrypted["errcode"]);
```

cybercash_base64_encode()

A cybercash_base64_encode()az URL-be való foglalásra kódolja a titkosított üzenetet:

```
$outbuff = cybercash_base64_encode($encrypted["outbuff" ]);
$macbuff = cybercash_base64_encode($encrypted["macbuff"]);
```

fopenQ

A **cybercash_base64_encode()** és a **cybercash_base64_decode()** között a kódolt sztringet egy URL-be kell beépítened, meg kell adnod az URL-t az fopen()-nel a CyberCash-nél lévő CGl-parancsfájlnak, fel kell darabolnod az eredményeket, majd a végeredményt a dekódolásba kell táplálnod. Szükség lesz hibaellenőrzésre és a a hálózati időtúllépés kezelésére alkalmas kódra. (Az **fopen()** leírása a 8. fejezetben található.) A következő kód egy példát mutat be (az URL meg fog változni, ahogy a VenSign újracímkézi a CyberCash termékeket):

```
$file = fopen("http://cr.cybercash.com/cgi-bin/", "r");
```

cybercash_base64_decode()

A **cybercash_base64_decode()** dekódolja a base64 kódolt üzeneteket:

```
$output = cybercash_base64_decode($result_output);
$mac = cybercash base64_decode($result mac);
```

cybercash_decr()

A cybercash_decr()a kereskedői és session-kulcsok segítségével visszafejti a CyberCash-től kapott üzeneteket:

```
$decrypted = cybercash_decr($merchant key, $session key,
$output) ; if($decrypted["errcode"] === falsé)
    {
   print ("<br>output buffer: " . $decrypted["outbuff"]
       . "<br>mac buffer: " . $decrypted["macbuff"]); }
else    {    print("<br>error    code:    "    .
   $decrypted["errcode"]);
```

Payflow

A VeriSign **(www.verisign.com/payment)** Payflow Pro-ja a pénzügyi tranzakcióknak széles választékát nyújtja, a hitelkártya-elszámolást is beleértve. A Payflow Unix alatti telepítéséhez fordítsd be a PHP-t a ~with-pfpro-vaí, és a VeriSign-tól töltsd le a szoftverfejlesztő csomagot (SDK) (először regisztráltatnod kell magad a VeriSign-nál). A Payflow mintha hiányozna a PHP 4.0.6 Win32 bináris állományokból.

pfproJnitQ

A Payflow-folyamatot a **pfpro_init()** indítja el. Ha elfelejtenéd ezt a függvényt, **a** következő Payflow-függvény automatikusan elindítja a folyamatot:

```
pfpro_init();
```

pfpro_version()

A pfpro_version()a Payflow-könyvtár verziószámát adja vissza, és arra használható, hogy figyelmeztesse a rendszergazdádat, ha megfelelő tesztelés nélkül telepítettek újabb verziót:

```
if (pfpro_version()      !=    "L211")
    {
    print("<br>Warning, payflow library wrong version.");
    i
```

pfpro_process()

A **pfpro_process()** feldolgozza a tranzakciót, és tömbként adja vissza az eredményt. A következő kód egy tranzakciót dolgoz fel, és a **print_r()-rel** kinyomtatja az eredményt:

```
$test    =    arrayC'USER"    =>    "fredsmith",    "PWD"    =>
    "fsl23", "TRXTYPE"    =>    "S",    "AMT"    =>    49.95,
    "TENDER"    =>    "C","ACCT"    =>    "1234123412341234",
    "EXPDATE"    =>    "0904");
$server    =    "test-payflow.verisign.com";
$port    =    44 3;
$timeout    =    30;
$sslproxy_host    =    "192.168.32.45";
$sslproxy_port    =    123;
$sslproxy_logon    =    "securetran";
$sslproxy_password    =    "secl23";
if ($array    =    pfpro    process($test,    $server,    $port,    $timeout,
    $sslproxy_host,    $sslproxy_port,    $sslproxy_logon,
    $sslproxy_password))
    {
    print    r($array);
    }
else    {    print("<br>pfpro_process()
    failed.");
```

Az első paraméter mindenképpen szükséges, a többi opcionális, de a szervert úgyis majdnem mindig rneg fogod határozni. A proxy beállításáról kérdezd meg a rendszergazdát.

pfpro_process_ra w()

A **pfpro_process_raw()** ugyanazt csinálja, mint a **pfpro_process(), két** különbséggel. Az első megadott paraméter egy sztring az összes kulcsértékkel, amelyek az URL kódoláshoz hasonlóan vannak kódolva, és a kijövő adat is sztring. A **pfpro_process()-t** könnyebb használni, és megbízhatóbb eredményeket ad, mert a sztring kódolásánál nem fogsz hibát ejteni.

pfpro_cleanup()

A **pfpro_cleanup()** fejezi be a Payflow-folyamatot. Ha elfelejted ezt a függvényt, akkor a következő függvény automatikasan elindítja ezt a folyamatot a parancsállomány végén:

```
pfpro_cleanup();
```

t

CCVS

A Red Hat Hitelkártya Ellenőrző Rendszerről (CCVS) a **www.redhat.com/products/ /software/ecommerce/ccvs** oldalon olvashatsz. A hozzáférési kulcs és a fenntartás nem ingyenes. Az ingyen teszt helyett 95$-ért lehetőség van 30 napos kipróbálásra. A kulcs egy kereskedő egy szerverére szól, így ahogy terjeszkedsz, egyre többe kerül. Sajnos ez az extra költség nincs a Red Hat honlapján feltüntetve.

A CCVS-szoftver a weboldalad és a hitelkártya klíringház között helyezkedik el. A Red Hat állítása szerint a CCVS együttműködik a „legtöbb" klíringház protokollal, így nézd meg, kinél nyitod a kereskedői számlád, ők kivel végzik a klíringet, és nézd meg, hogy a CCVS ismeri-e a megfelelő protokollt. Ha nem ismeri, akkor valószínűleg használhatod a PHP hálózati és titkosítási függvényeit a tetszésed szerinti klíringházzal való kommunikáláshoz.

A Red Hat állítása szerint a CCVS jól együttműködik az olyan valós idejű online klíringekkel, amelyek több batch-fájl átvitelével egyidőben történnek. Ez azt jelenti, hogy az ügyfeleid új előfizetéseket vásárolhatnak, miközben te épp az előfizetések havi frissítését végzed, és ezt bizonyos hitelkártyaelszámoló-rendszereknél nehéz megcsinálni.

A CCVS működik az USA-ban, Kanadában és néhány más országban. A CCVS a Visa 2nd Generation K Formát protokollját használja, így keresd ezt a helyi bankodnál vagy kereskedői számlaszolgáltatódnál.

A dokumentáció azt sugallja, hogy a protokollt a modemes csatlakozáshoz írták, és hogy kevésbé kifinomult, mint a PHP. Ha a Red Hat nyerővé akarja tenni a termékét, akkor jó kis átalakításra van szükség a PHP4-hez.

CCVS telepítése Unix alatt

A CCSV Unix alatti telepítéséhez kövesd az alábbi lépéseket:

1. Töltsd le a **www.redhat.com/products/software/ecommerce/ccvs/-rol** és telepítsd a CCVS-t.
2. Mutasson a PHP a — with-ccvs-sel a CCVS könyvtárára.
3. Indítsd el a ccvsd-folyamatot, hogy konfigurálhasd a PHP-val való használatot.
4. Állítsd be, hogy a PHP-folyamatok ugyanazzal az azonosítóval fussanak, mint a **ccvsd.**

Gyors megoldások

Adatok hash-elése

Ez a megoldás az mhash-függvényt használja. Az **mhash()** az, amely az igazi munkát végzi, a maradék adminisztrációs szolgáltatásokkal használható, ahol a rendelkezésre álló hash-ekről szeretnél információt.

mhashQ

Az mhash() több hash-t is előállít. A hash-ek különböző típusú adatokkal való használhatóságát tesztelve végezzük el az alábbi próbát, amely sztringgel, egész- és lebegőpontos számmal, speciális és nemnyomtatható karakterekkel vizsgálja ezt:

```
$data[] = "gfgfgfdgfsdgs";
$data[] = 125;
$data[] = 125.986;
$data[] = "special áé,";
$data[] = "non print" . chr(13) . chr(lO);
```

CRC32

A CRC32 egy normál 32 bites CRC, amit hálózathoz és lemezekhez használnak, míg a CRC32B-t speciálisan a ZlP-fájlokhoz. A következő kód végigmegy a teszt adaton, és minden egyes adatból CRC32- és CRC32B-sztringeket hoz létre. A könnyebb összehasonlítás végett a bevitt és a kapott adatok egy táblázatba kerülnek. Mivel a CRC-k 32 bites bináris adatok, így a kód a CRC-ket hexadecimálisán jeleníti meg a **bin2hex()** segítségével:

```
reset($data);
print ("<tablextrxtd>Data</tdxtd>   </td>"
    .   "<td>CRC32</tdxtd>    </tdxtd>CRC32B</tdx/tr>") ;
while (üst ($k,   $v)   =   each($data))
    {
    $hash  =  mhash(MHASH_CRC32,  $v);
    $hashb = mhash(MHASH_CRC32B, $v) ;
    if($hash === falsé)

    print("<br>made a hash of }        htmlentities($v)
    else                              );

       print ("<tr><td>" . htmlentities ($v) . "</tdxtd>Snbsp; </td>"  .
                   bin2hex($hash) . "</tdxtd> </td>"  .
                   bin2hex($hashb) . "</tdx/tr>") ;


print{"</table>");
```

Az eredményeket a következő táblázat mutatja, amely mindig 32 bites és egyenletesen oszlik el a 32 bites értékek tartományán. A CRC-k leírása a 8. fejezetben található.

Data	CRC32	CRC32B
gfgfgfdgfsdgs	7478bl78	20859599
125	29432b3c	e7c62b61
125.986	b50d8790	b8cf7900
special aé,	46243921	f5cbe007
non print	f5cdl30c	7dd5c4c7

Hivatkozás: oldal:
CRC-számolás fájlokra **286**

MD5

Ha nem a biztonságon van a hangsúly, akkor az MD5-tel gyorsan lehet jelszavakat menteni. Az alábbi kód a CRC32 MD5 alatti változata:

```
reset($data);
print ("<tablextrxtd>Data</tdxtd>   </tdxtd>MD5</tdx/tr>") ;
while(list($k,     $v)     =     each($data))

    $hash = mhash(MHASH_MD5, $v);
    if{$hash === falsé)

       print("<br>Error with " . htmlentities($v));

    else

       print ("<tr><td>" . htmlentities($v) . "</td><td> </td>" >"
                 . bin2hex ($hash) . "-


print("</table>");
```

Az első változtatás a CRC32 fejlécet MD5-re cseréli ki. A második a két mhash()-függ-vényt egy olyan mhash()-re cseréli ki, ami tartalmazza a MHASH_MD5-öt. A *hash* szót kicseréli a hibaüzenetben, majd a végső print()-et kicseréli arra, hogy csak egy eredményt nyomtasson ki az mhash()-ből.

A következő lista mutatja az eredményt hexadecimális, 16 bites MD5-sztring formájában:

Data	Hash
gfgfgfdgfsdgs	d8b32ce7 3 0bd0d54 63a8 5dlc2a4 4b7 0 8
12 5	3defl84ad8f4 755ff2 698 62ea77393dd
12 5.98 6	f3b22c8513875add521ef724a53a5c7 9
special áé,	ff5e791248822257f8a279c4b72c88f4
non print	80b5da917ee85d4e797831(

A lista olyan nemmegjeleníthető karaktereket is mutat, mint például Od és 08. Amikor egy adatbázisban MD5-öt mentesz el, akkor bináris-biztos mezőt vagy megjelenítést, például

hexadecimálist kell használnod. Amikor URL-en vagy e-mail-en keresztül MD5-öt küldesz, akkor base64-es kódolást kell használnod.

És a *maradék*

Fussunk át gyorsan még néhány **mhash()** hash-en. Az alábbi kódot használtam, csak a kiemelt kódrészben a *SHA1* sztringet más hash-ek neveivel cseréltem ki:

```
print ("<table><trXtd>Hash</td><td>    </td><td>Result</td>"

print ("<tr><td>SHAl</td><td> </td>"
    .  "<td>"  .  bin2hex(mhash(MHASH_SHAl,  [0])) . "</td></tr>");
 $datá print("</table>");
```

A következő lett az eredmény (a sorok a könyv formátuma miatt törve vannak):

```
Hash          Result
SHA1       3 6c02 4 4 3ad02 211bb93 8 25e2ab0 57 0fe7 5 9 9b0bf
HAVAL2 5 6   b0e7 77d66725 4 41baaf19dff51d8e4e94b56be0ad61605d8d814 05b2c6
             a777eO
HAVAL22 4   323323aele7550f323527ac6891888 5557 4a9d94clcl46518 8e2 8c2 0
HAVAL192   7d4dbefa3bl5b4 5ea4dd9158 0301dfle8 8f2bla6b0bla3db HAVALI 60
dfbbab634 9 67ab93ab8 68223de90ec2c83d70c63 RIPEMD160  631b5 94ffdb75af2
0el4a62 5 85f8d4 00a02 0 964 2 GOST
bcl0c57f8931759eafe871e68f72f248273bda56018d98732a0b7bla48cc
             6394 TIGER      424e4 610167e4dlaa5 87eff90d7 6ccbel4
87 74 24a0ddecb0
```

mhash_get_hash_name()

Ha van hash-azonosítód, mint például a **MHASH_CRC32,** és meg akarod kapni a hash hivatalos nevét, ezt a kódot használd:

```
!
print {"<br>"    .   mhash_get_hash_name(MHASH_CRC32) } ;                    j
```

Az eredmény:

```
CRC32
```

mhash_get_block_size()

Ha megvan a hash azonosítója és ki akarod számolni, hogy mennyi helyre van szükséged, a következő kódot használd:

```
print("<br>" . mhash_get_block_size(MHASH_CRC32) ) ;
```

Az eredmény (lásd alább) az adott típusú hash-hez szükséges hossz. Használhatod ezt a számolást, amikor automatikusan építesz fel egy adatbázis-táblázatot, és ki kell számolnod a mezők méretét:

mhash_count() *

A mhash_count() a hash-hez rendelt legmagasabb értékű azonosítót adja vissza, és lehetővé teszi, hogy olyan kódot írj, amellyel az összes rendelkezésre álló azonosítón végiglépkedhetsz. A következő kód végiglépked az első hash-azonosítótól, a 0-tól a mhash_count()-tól kapott értékig, és mind a nevet, mind a méretet egy helyes kis táblázatba foglalja:

```
$hashes = mhash_count{);
print ("<table><tr><td>Hash</tct><td> </td>"
    . "<td>Block Size</tdx/tr>")
; for($h = 0; $h <= $hashes;
$h++)
    t
    print("<tr><td>" . mhash_get_hash_name($h) .
                "</td><td>  . mhash_get_block_size($h) .
                "</tdx/tr>") ;

print ("</table>");
```


Itt van az eredmény, amely azokat az azonosítókat is tartalmazza, amelyeknek nmcs nevük vagy tömbméretük. Az eredmények változni fognak, ahogy új hash-típusok hozzáadódnak, és a régi idejétmúltak törlődnek (üres bejegyzéseket hagyva hátra):

```
Hash            Block Size
CRC32           4
MD5             16
SHA1            20
HAVAL2 5 6      32
                0
RIPEMD160       20
                0
TIGER           24
GOST            32
CRC32B          4
HAVAL22 4       28
HAVAL192        24
HAVAL160        20
```

mhash_keygen_s2k()

Az **mhash_keygen_s2k()** segít a felhasználó által megadott jelszóból egy salt-sztringgel kódolt S2K-algoritmussal kulcsot generálni, mely algoritmus specifikációi az OpenPGP RFC2440-ban találhatók. A függvénynek szüksége van a hash nevére, a jelszó adataira, a salt-sztring értékére és a hosszra. A következő kód az MD5-öt használja a hash-hez, a $data-érték a jelszó, a nem túl véletlenszerű **turkey** a salt-sztring, és 32 a sztring hossza.

```
$hash = mhash_keygen_s2k{MHASH_MD5, $data[0], "turkey",
32); print("<br>" . htmlentities($data[0] ) . " " .
bin2hex($hash));
```

Megjegyzés: A salt-sztring értéke a kezdő érték, amely elindítja a folyamatot. Lehet véletlenszerű vagy akár minden alkalommal ugyanaz. Ha mindig ugyanazt a salt-sztringet és jelszót használod, akkor mindig ugyanazt a kulcsot fogod kapni. De ha más salt-

sztring-értéket használsz, akkor még ugyanazzal a jelszóval is más kulcsot fogsz kapni. Ha hagyod, hogy a felhasználó adja meg a jelszót, amelyből a kulcsot generálod, lehet, hogy ezt a jelszót más weboldalon is megadják, tehát ahhoz, hogy a kulcs egyedi legyen, a weboldaladnak egyedi salt-sztríng-értéket kell választanod.

Itt a példaként felhozott kód eredménye. A 32 bites eredményt, amely hexadecimálisán jelenítődik meg, tördeltem, hogy a könyv formátumának megfeleljen:

```
gfgfgfdgfsdgs
b2alflba22 3195 92b7 3 9557aell8c2ee7 4 7c0 5 37 32 005c33 5c43a0 0 8eb
        b0e0e8
```

A salt-sztring-mező csak egy véletlenszerű kezdőpontja a generálás folyamatának. Tedd a salt-sztringet annyira véletlenszerűvé, hogy az emberek ne találhassák ki. Az **mhash()** létrehozhat egy hash-t egy sztringből, amit például a **microtimeQ** ad eredményül, és a hash válhat a salt-sztring-gé. A salt-sztring-re szükséged lesz, amikor a kulcsokat ellenőrződ, tehát mentsd el a salt-sztring-et is, amikor a kulcsot mented.

A salt-sztring 8 bitre korlátozott, és az alábbi hibaüzenetet kapod, ha a salt-sztring túllépi a 8 bitet. A PHP 4.0.7dev szintúgy kilép, amikor túlméretezett salt-sztring-mezőt próbál feldolgozni:

```
Warning: The specified salt [13] is more bytes than the required by the
         algorithm [8]
```

5. *fejezet*

MySQL és PostgreSQL

Áttekintés

A MySQL és a PostgreSQL egyaránt nyílt forráskódú adatbázisok, és ideálisak kis és közepes méretű honlapokhoz. Vannak hasonlóságaik, amelyek mindkettőt egyaránt jó választássá teszik a legtöbb weboldal számára, és vannak különleges funkcióik, amelyek előtérbe helyezhetik valamelyiket egy bizonyos projekt szempontjából. A fejezet első része áttekinti a hasonlóságokat és a különbségeket, ez megkönnyíti a megfelelő adatbázis kiválasztását, míg a második rész, a „Gyors megoldások" megmutatja, hogyan oldhatjuk meg a leggyakrabban felmerülő feladatokat MySQL-lel és PostgreSQL-lel. Sőt, olyan kód megírására törekedtem, amely mindkét adatbázishoz megfelelő, vagyis minimális átalakításra lesz szükséged, ha át kell térned egyikről a másikra.

A MySQL már Windows és Windows NT alatt is elérhető, vagyis mindkét operációs rendszer használói elkezdhetnek MySQL-lel dolgozni. Ha PostgreSQL-t szeretnél használni Windows alatt, akkor jelenleg számos termék bonyolult telepítésével és fordításával kell szembenézned. A Unix és a Linux különböző verzióin mind a MySQL, mind a PostgreSQL egyaránt elérhető, egyaránt könnyű (vagy nehéz) telepíteni őket, és nagyjából ugyanolyan szintű terméktámogatást kapunk hozzájuk.

Régebben Windows NT alatt készítettem weblapokat, melyek igény szerint Windows NT-n, Solaris-on, Linuxon vagy FreeBSD-n kerültek megjelenítésre, ezért mindig MySQL-lel kezdenék, és akkor használnék PostgreSQL-t, ha a PostgreSQL valamelyik speciális funkciójára lenne szükségem. A Linux a Mandrake 8 disztribúcióval hasonlóan felhasználóbarát telepítést és kezelőfelületet biztosít, mint a Windows NT, ezért lehet, hogy teljesen áttérek Linuxra.

A MySQL korábban gyorsabb teljesítményt nyújtott a számos weblapon használt egyszerű, olvasásigényes adatbázisokkal, mert nem végzett erőforrás-igényes tranzakció-feldolgozásokat. Most már a MySQL is képes tranzakció-feldolgozásra és a PostgreSQL felveszi vele a versenyt a frissítésigényes adatbázisok területén, tehát a teljesítmény már nem képezheti igazán a választás alapját, hacsak nem tervezed egy nagyméretű és különleges szerkezetű adatbázis használatát.

Történet

Az adatbázisok evolúciójában a nyílt forráskódú fejlődésnek két fő ága volt: a nagy és lassú buldózer, amelyik bárhová eljut és bármit megcsinál (Postgres), és a kicsi, de gyors Porsche (MySQL). A Postgres egy a Berkeley egyetemen folyó projektből nőtte ki magát, ami látszólag az Oracle egyes funkcióval akart versenyezni.

Queenslandben, Ausztráliában a Bond Universityn a programozó Dávid Hughes megpróbált a Postgres-szel összekapcsolni egy alkalmazást SQL-en keresztül (a Postgres-ben ek-

r
kor még nem volt SQL). Dávid megírta a saját SQL-feldolgozó programját, és miniSQL-
|
nek nevezte. Dávid úgy találta, hogy a Postgres túl lassú, ezért egy saját egyszerű és gyors adatbázist készített a miniSQL-hez, amit mSQL-nek nevezett.

A MySQL később felváltotta az mSQL-t, de egyszerű és gyors maradt. A Postgres-ből végül kifejlődött a PostgreSQL.

Az emberek többsége eddig a Porschét részesítette előnyben a traktorral szemben.

MySQL

Az alábbi idézet a MySQL dokumentációjából származik és a MySQL tulajdonjogait magyarázza meg: „A MySQL AB az a svéd vállalat, amelynek tulajdonosai és működtetői a MySQL alapítói és fő fejlesztői. Célkitűzésünk a MySQL fejlesztése és az adatbázisunk elterjesztése az új felhasználók körében. A MySQL AB tulajdonában van a szerver forráskód szerzői joga és a MySQL márkanév. A szolgáltatásainkból származó bevételekből jelentős összegeket fordítunk a MySQL fejlesztésére." A MySQL-nek egy fura szoftverfelhasználói szerződése volt, de mára a standard GNU General Public License (GPL) alatt forgalmazzák. A fejlesztők az mSQL vizsgálatával kezdték, majd elhatározták, hogy egy olyan adatbázist írnak, amely jobban illeszkedik nagyobb weblapok kiszolgálásához. Monty Widenius, a MySQL fejlesztésvezető'je egyértelműen azt a célt tűzte ki, hogy egy praktikus, mindennapi használatra alkalmas adatbázist fejlesszenek, amely biztosítja a MySQL további sikerességét.

PostgreSQL

A PostgreSQL a berkeley-i University of Californián 1986-ban megkezdett Postgres szoftverfejlesztésből származó Postgres95 továbbfejlesztett változata. A Postgres kereskedelmi változatát az Illustna Information Technologies fejlesztette ki; az Informix megvásárolta az Iílustriát, majd az IBM megvásárolta az Informíxt, így a Postgres-ből néhány elem átszivároghat az IBM DB2-jébe. A PostgreSQL 7. verziója óta számos MySQL felhasználó a PostgreSQL jó teljesítményéről és magbízhatóságáról számolt be a közepes méretű weblapok területén, tehát számíthatunk a PostgreSQL terjedésére, amint az emberek kellő gyakorlatot szereznek az adatbázisok területén ahhoz, hogy kihasználják a PostgreSQL plusz szolgáltatásait.

A Unix környezettől eltekintve a PostgreSQL telepítése még mindig nehézkes, ami a Windows-t használó webfejlesztők millióit arra sarkallja, hogy előbb a MySQL-t sajátítsák el, ezért a MySQL tartja az előnyét. Mielőtt PostgreSQL-t telepítenél Windows-ra, számos Unix komponenst kell telepítened, mindegyikük külön telepítési művelet. Megpróbáltam felkutatni egy időszerű, átfogó és pontos oktató anyagot a PostgreSQL Windows-os és Windows NT-s telepítéséről, de nem jártam sikerrel. A Windows-os problémák arra engednek következtetni, hogy más platformok elérése hasonló nehézségekbe fog ütközni, és a MySQL egyszerű, megbízható felépítése révén inkább meghódítja az új platformokat.

A fejezetben található PostgreSQL kódokat 7.1.l-es verzión teszteltük, Windows NT Service Pack 6a platformon, cygwin 1.3.1-et **(http://cygwin.com)** és cygpic 1.09-2-t (www.neuro.gatech.edu/users/cwilson/cygutils/Vl.1/cygipc/) használva. A cygpic és a cygwin nélkülözhetetlen, ha PostgreSQL-t akarsz használni Windows NT-n. E két program egy kiterjesztett POSIX-környezetet biztosít, kiegészítve a Unix bash shell-jével. Ha a bash shell és hasonló kifejezések összezavarnak, akkor ne próbálkozz a PostgreSQL Windows NT-s telepítésével. Először próbálj Linux-gyakorlatra szert tenni, vagy kérj meg egy gyakorlott Linuxost, hogy vezessen végig a telepítési és konfigurálási procedúrán.

Néhány különbség

A MySQL és a PostgreSQL közel áll az ANSI/ISO SQL92 szabványához és a PostgreSQL az SQL99-ből is átvett néhány dolgot. Mindkettő folyamatosan fejlődik, ezért nincs sok értelme a szabványoknak való megfelelés alapján választani közülük. Mindössze arra van szükség, hogy tisztában legyél a különbségekkel, és ezeket dokumentáld is a kódodban. A főbb funkciók közti eltérések sokkal többet nyomnak a latban, mint a szabványokhoz való igazodás; a szabványoktól való eltérések nagy részét és a funkciók különbségeit az alábbiakban felsoroljuk, egyes különbségeket, mint pl. az adattípusok, a fejezet hátralévő részében tárgyaljuk.

A MySQL-ben vannak olyan SQL programozási megoldások, amelyek remélem, egy napon végleg eltűnnek. Az SQL is csak egy programnyelv, a maga különcségeivel, amelyeket én nem-szabványosnak neveznék, más esetekben a MySQL fejlesztői jobban ragaszkodnak a szabványokhoz, mint az SQL-t fejlesztők. Példa erre a megjegyzéseket jelző karakterek: a programozási nyelvekben legáltalánosabban elterjedt megközelítés a #, vagyis a kettős kereszt szimbólum a sor elején. Az ANSI/ISO/SQL létrehozói a ~ szimbólumot választották a megjegyzések jelölésére. A MySQL a #-et választotta és a — szimbólumot is támogatja, míg a PostgreSQL fejlesztői ragaszkodtak a — szimbólumhoz. Azt hiszem, a legtöbb programozó számára egyszerűbb volna, ha a sor elején kettős kereszttel jelölhetnék a megjegyzéseket, és minden programnyelv könnyebben megtanulható volna a jövőbeli programozó nemzedékek számára, ha egységes megjegyzés-határoló szabványt használnának.

A MySQL nem szabványos SQL-je engedélyezi a kettős keresztet a megjegyzésekhez (de szintén elfogadja a szabványos — szimbólumot), a " szimbólumot használja a rendszer-azonosítók meghatározásához (amit a legtöbb programozó ritkán használ), a " szimbólummal jelöli a sztringeket (de szintén elfogadja a szabványos ' szimbólumot is, amit ebben a könyvben a példaprogramokban használok), és a | | szimbólum jelentése VAGY, nem pedig sztringek összefűzése. Elkerülheted a | | karakter téves használatát, ha mindig or-t írsz, ha a logikai VAGY-ra van szükséged és mindig kiírod az ilyen kifejezéseket, mint az or és az **and.** A legtöbb programnyelvben ez a legbiztosabb módja a programozásnak.

Dátumok

A PostgreSQL és a MySQL majdnem ugyanúgy kezeli a dátumokat, de itt megint csak különbségek vannak az adatbázisok közt; a legnépszerűbb kereskedelmi adatbázisok a legrosszabbak, mivel az amerikai dátumformátumhoz, a hónap/nap/évhez ragaszkodnak, amely túl könnyen összekeverhető az európai nap/hónap/évvel. A 02/06/2002 és a 02/06/2002 megkülönböztethetetlen egymástól: az egyik az új-zélandi Waitangi Day (február 6.), a másik az olasz Köztársaság Napja (június 2.). Az SQL-szabvány az év - hónap -nap, amit mindkét adatbázis elfogad.

Kis- és nagybetűk megkülönböztetése

A PostgreSQL a sztringek összehasonlításakor megkülönbözteti a kis- és nagybetűket, és biztosít olyan utasítást, amely a kis- és nagybetűkre érzéketlen összehasonlítást végez, míg

a MySQL a formai érzéketlenségtől indul, és be lehet állítani, hogy figyelembe vegye a kis-és nagybetűk közti különbséget. A MySQL formailag érzékeny összehasonlítást használ a **char, text** és **varchar** sztring-típusokra, míg a blob-típusra bináris összehasonlítást végez (ami nem különbözteti meg a kis- és nagybetűket). A **char-, text-** és varchar-típusok is formailag érzéketlenné tehetők, ha a **binary** tulajdonsággal definiáljuk őket. A MySQL olyan utasítással is rendelkezik, amely szöveges mezőket kis- vagy nagybetűssé alakít, ezáltal kézzel is elvégezhető az összehasonlítás.

A PostgreSQL formailag érzéketlen módon kezeli a mezők, táblák és adatbázisok neveit, de megadhatsz egy meghatározott formát, ha szeretnéd megbonyolítani az életedet. A MySQL-adatbázis definíciói könyvtárakként szerepelnek a fájlrendszerben, míg a táblanevek egyedi fájlok, és mindkettőt befolyásolja az operációs rendszer formai érzékenysége. Ha a MySQL-ben kezdetben beállítod a **lower_case_table_names** = **1** paramétert, akkor minden táblaneved kisbetűs lesz, ezáltal elkerülheted a formai érzékenységből adódó csapdákat. A MySQL különálló könyvtárakban tárolja az adatbázisokat és külön fájlokban a táblákat, ami rugalmasságot biztosít, de az operációs rendszer fájlnevekre vonatkozó megszorításai komoly gondot okozhatnak, ezért csak kisbetűket és számokat használj.

Tranzakciók

A PostgreSQL-ben mindig is voltak tranzakciók, és a MySQL nemrég vezetett be egy új táblatípust, amely lehetővé teszi a tranzakciókat, ami logikus lépés volt, mivel a tranzakciók mégiscsak csökkentik a teljesítményt. Ha egy weblapon nagyméretű, csak olvasható táblákat akarsz használni a referencia-információk számára, akkor a MySQL standard, tranzakció nélküli tábláit válaszd, és a frissítendő táblákhoz használd a MySQL új típusú tábláit vagy a PostgreSQL-t. Semmi nem gátol meg abban, hogy párhuzamosan futtasd a kettőt. Egy több szervert használó honlapon az elsődleges referencia-adatbázison engedélyezheted a tranzakciókat, hogy garantáld a hibamentes frissítést, majd egy tranzakciómentes másodpéldányt készíthetsz a referenciaszerverről — ez ideális megoldás a több ezer szerverből álló keresőmotorok esetében.

Tárolt eljárások

Húsz évvel ezelőtt a tárolt eljárások az adatbázisok megmentői voltak, mivel lehetővé tették, hogy az üzleti logikát az adatok definíciójába ágyazzák, ahelyett, hogy számos különböző programnyelvben megírt kódba kellett volna eltemetni. A tárolt eljárások sokkal gyorsabban futnak le, mivel a feldolgozást az adatbázis kódja hajtja végre, ami csökkenti a forgalmat az adatbázis és az alkalmazás közt. A Star Trek világa egyre kevésbé tűnik hihetetlennek.

A tárolt eljárások a több programnyelv együttes használatának problémáját oldották meg, mikor az emberek az eljárásokat COBOL-ban írták, a jelentéseket RPG-ben, az adattípusok közti konvertálást Assembly nyelven, de adódott egy új probléma, ugyanis még egy programnyelvet meg kellett tanulni, a tárolt eljárások nyelvét, ami csak az adott adatbázisra korlátozódott. Ma a programokat, jelentéseket és a konvertálást egyaránt megírhatod PHP-ban, ami azt jelenti, hogy egy üzleti logikát egyetlen helyen tárolhatsz, egy függvényben

vagy objektumban, pl. egy iizletilogika01.html include-fájlban, és bármely szkriptbe beágyazhatod, amelyik hozzáfér az adatbázishoz. A tárolt eljárásoknak nem kell többé megoldaniuk a több programnyelv együttes használatából fakadó problémákat. A tárolt eljárások valójában egy újabb problémát jelentenek, azáltal, hogy a PHP-n kívül egy másik nyelvet is meg kell tanulnunk.

Ha olyan rendszert használsz, ahol az adatbázis egy külön adatbázisszerveren van, míg a PHP a webszerveren, akkor a klasszikus kliens/szerver hálózati forgalom problémával szembesülsz, amit jó SQL-tervezéssel lehet megoldani. A tárolt eljárások megelőzhetnek néhány balesetet, amelyek az SQL-tervezés hibáiból adódhatnak, de a tárolt eljáráskora hagyatkozás növelheti a problémákat, ha egyvalaki kifinomult tárolt eljárásokat fejleszt, míg a többi fejlesztőnek nincs kellő tapasztalata az SQL-tervezésben.

A PHP kódbeágyazása, egyszerű függvény- és objektumhasználata és a PHP4 sebessége révén feleslegessé teszi a tárolt eljárások használatát. Egy kisebb fejlesztői csapatban megspórolhatod annak a költségét, hogy az egyik fejlesztőnek specializálódnia kell még egy nyelvre. A tárolt eljárások arra valók, hogy megvédjenek egy adatbázist, amelyet különböző nyelveken megírt alkalmazások használnak.

Triggerek

A triggerek a tranzakciókhoz hasonlóan csökkentik a teljesítményt, de olyan funkciókat biztosítanak, amelyeket nehezen tudnál leprogramozni; tehát, ha triggerekre van szükséged, merülj el a PostgreSQL-ben, mert a MySQL-ben nem találod meg a megfelelőjét. Alaposan vizsgáld meg az igényeidet mielőtt bármit implementálnál, mert a weblapok kliens/szerver felépítése feleslegessé teszi a triggerek néhány hagyományos alkalmazását. A triggerek adatbázisfüggőek, ha még soha nem használtad volna őket; ha szükséged van rájuk tanulmányozd a PostgreSQL dokumentációját és oktató anyagait a **postgresql.org-ol-**dalon.

Nézetek

A legjobb kezdeti nézet a sima, egyszerű nézet, amelyet nem befolyásolnak tárolt eljárások vagy egyéb, az adatbázis-kezelőbe beépített trükkök. Ha már megértetted az adatok struktúráját és tartalmát, akkor hasznosíthatod a korlátozott nézeteket, amelyek pl. lehetővé teszik, hogy a dolgozók megnézzék egymás nevét és telefonszámát, de nem engedik, hogy lássák egymás fizetését. Tovább is fokozhatod az ellenőrzést, ha megengeded a dolgozóknak, hogy lássák egymás születési napját és hónapját, vagyis megünnepelhetik a születésnapokat, de nem engeded, hogy lássák egymás születési évét, így az a dolgozó, aki 20 ezer **dollárt** költött plasztikai műtétre, megőrizheti a titkát. SQL-ben is megírhatod a korlátozott nézeteket, de ha a következő programozó hozzáad egy új mezőt, az SQL-nézet hibás lesz. A PostgreSQL-nézet funkciója lehetővé teszi, hogy a nézetet mint új táblát rögzítsd az adatbázisba, és a PostgreSQL biztonsági megoldását használva ellenőrizheted a hozzáféréseket a nézetekhez. Ha egy adatbázis-adminisztrátorod van és számos programozód, és nézeteket használsz, akkor az adminisztrátor az egyedüli ember, aki kikémlelheti a dolgozók fizetését.

Adattípusok

Ha adatokat tárolsz, akkor biteket, bájtokat, egész számokat, sztringeket, bináris objektumokat és nagyon nagy bináris objektumokat akarsz tárolni. Minden mást az előbbi adattípusok valamelyikében tárolsz, vagy egy külső fájlban, és ekkor egy hivatkozást tárolsz valamelyik adattípusban. Hogyan viszonyul egymáshoz a PostgreSQL és a MySQL az adattípusok tekintetében?

Bitek

A MySQL set-típusa egy speciális bináris objektum, amely opciók felsorolását tartalmazza. A PostgreSQL-ben van egy SQL99-sz.abvány **boolean-típus,** amelyet használhatunk arra, hogy eltároljuk a set-opcióit, és van két **bit** sztring-típus is, amelyek úgy viselkednek, mint a char-mezők, de mindössze egyeseket és nullákat tárolnak. Noha a MySQL set-típusával bizonyos adatok esetén megspórolhatunk egy részt a programozásból, mégis a PostgreSQL boolean-adattípusa a legjobb választás a jövőre nézve, és a MySQL fejlesztői is megígérték, hogy beteszik a **boolean-t** a következő verzióba.

A PostgreSQL **bit-** és **bit** varying-mezői egyaránt egy nullákból és egyesekből álló sztringet tárolnak, a tartalomra vonatkozó bármilyen megkötés valamint, bármilyen paritást jelző vagy egyéb jelentéssel kitüntetett bit nélkül. A **bit** rögzített hosszúságú - meghatározod a hosszát, és nullával kerül kitöltésre. A **bit varying** változó hosszúságú, és ha nem adod meg a maximális hosszát, akkor tetszőlegesen hosszú lehet.

Ha hosszabb távra írok alkalmazást és MySQL-t használok, akkor a boolean adatokat függvények segítségével **tinyint** vagy **set** típusú mezőkben tárolom el, tehát valamikor a jövőben könnyedén lecserélhetem a **tinyint-** és set-mezőimet **boolean** típusúakra. Az SQL-szabványokhoz való alkalmazkodás hosszú távon fontosabb, mint egy kis adatfeldolgozási többlet vagy egy kis elveszített tárterület, bár sokkal valószínűbb, hogy az operációs rendszer alapján vagy a tranzakciók használata miatt, esetleg a hivatkozási integritás érdekében választasz operációs rendszert.

Egész számok

A PostgreSQL és a MySQL egész típusai az 5.1 táblázatban vannak felsorolva. Mindkét adatbázis közel áll az ANSI/ISO SQL92 numerikus adatokra vonatkozó szabványához, és semmi okot nem látok arra, hogy valaki az egész számok miatt válassza az egyiket a másik rovására. A MySQL opcionális unsigned-paramétert biztosít a nagy pozitív egészek tárolására és ez az opció főleg az int-típussal használatos, ha 32-bites jelöletlen egészeket akarsz eltárolni pl. CRC32-es mezőkben és ha **int autoíncrementet** használsz a kulcs-mezőkhöz (lásd a fejezet későbbi részét).

A PostgreSQL **bigint-je** nem biztos, hogy minden platformon elérhető, tehát lehet, hogy más formátumot kell használnod; továbbá a **money** is elavult, vagyis válassz egy másik formátumot, pl. a **decímal-t.** A MySQL **dec-e** a **decimai** alternatív neve, hasonlóan az **int** is használható az integer helyett.

Decimai és a numeric

A decimai és a numeric sztringként tárolja a számokat, tehát akármilyen hosszú számot eltárolhatsz. Mind a MySQL, mind a PostgreSQL korlátozásokat tartalmaz arra vonatkozóan, hogyan kezeli ezeket az adattípusokat, tehát a BCMath-ben vagy GMP-ben használt tetszőlegesen nagy számokat inkább hagyományos sztringben tároljuk. Ha mégis használod ezeket az adattípusokat, meghatározhatod a maximális hosszt és a számolási pontosságot, pl, decimal(ll,2).

Serial

A PostgreSQL serial-je és a MySQL int autoincrement-je a numerikus azonosítók automatikus növelésére szolgál, és pl. számlák azonosítójához vagy tranzakciószámokhoz használható. A MySQL megengedi, hogy az egész mezőkhöz unsigned-paramétert adjunk, annak érdekben, hogy megduplázzuk a tartományukat, ha automatikus növelésű mezőként használjuk őket. A későbbi „Azonosítók beillesztése" részben tárgyaljuk az automatikus növelésű egészek azonosító kulcsként való használatát.

5.1 táblázat Egész adattípusok a MySQL-ben és PostgreSQL-ben.

Típus	Hossz (bájt)	Értéktatomány	MySQL	PostgreSQL
tinyint	1	-128-tól + 127-ig	Igen	Nem
smallint	2	-32768-tól +32767-ig	Igen	Igen
mediumint	3	-8388608-tól +8388607-ig	Igen	Nem
integer	4	-2147483648-tól +2147483647-ig	Igen	Igen
bigint	8	17-től 18 számjegyig	Igen	Igen
decimai	Változó	Nincs korlátozva	Igen	Igen
numeric	Változó	Nincs korlátozva	Igen	Igen
serial	4	O-tól +2147483647-ig	Nem	Igen
money	4	-21,474,836.48 dollártól +21,474,836.47dollárig	Nem	Igen

Lebegőpontos számok

A MySQL és a PostgreSQL azonos négy és nyolc bájtos lebegőpontos számokkal rendelkezik, utóbbi megegyezik a PHP belső nyolcbájtos formátumával - vagyis csak akkor térj el a duoble precision használatától, ha a számítási eredmény garantáltan elfér a kisebb mezőben, vagy ha nem bánod, ha az adatbáziskezelő megcsonkítja a túl nagy számokat. A lebegőpontos típusok felsorolása az 5.2 táblázatban található.

A MySQL double-ja a double precision alternatív neve. A MySQL float típusához tetszőleges pontosságot lehet beállítani, ami nem SQL92 szabvány; mivel ehhez az adattípushoz ritkán szükséges a számítási pontosság meghatározása, a kompatibilitás érdekében inkább kerüld el a float használatát.

Sztringek

A rövid sztringek tárolására a MySQL átfogóbb körű típusválasztékot kínál, azaz a legjobb választás lehet olyan alkalmazásokhoz, amelyek jó hasznát veszik a rövid szöveges mezőknek. Ahogy csökken a merev lemezes tárolókapacitás ára, egyre kevesebb értelme van néhány sztring-mező'nként néhány bájtot megtakarítani. A sztnng-típusokat az 5.3 táblázat sorolja fel.

5.2 táblázat Lebegőpontos adattípusok a MySQL-ben és PostgreSQL-ben

float	4	6 tízes helyi érték	Igen	Nem
reál	4	6 tízes helyi érték	Igen	Igen
double precision	8	15 tízes helyi érték	Igen	Igen

5.3 táblázat Sztring adattípusok a MySQL-ben és PostgreSQL-ben

Típus	Hossz mező (bájt)	Adat	MySQL	PostgreSQL
char	0	1-től 255-ig	Igen	Nem
varchar	1	1-től 255-ig	Igen	Nem
tinytext	1	1-től 255-ig	Igen	Nem
text	2	1-től 65535-ig	Igen	Nem
mediumtext	3	1-től 16777216-ig	Igen	Nem
longtext	4	1-től 4294967295-ig	Igen	Nem
text	4	1-től 4294967295-ig	Igen	Igen
character	4	1-től 4294967295-ig	Nem	Igen
character varying	4	1-től 4294967295-ig	Nem	Igen

A MySQL varchar-, tinytext-, text-, mediumtext- és longtext-típusai változó hosszúságú mezők, amelyek a bennük eltárolt adatokon kívül egy, a hosszukat leíró mezőt tartalmaznak, mint az 5.3 táblázat mutatja. A PostgreSQL text-je megegyezik a MySQL longtext-jével, a character varying-je pedig a MySQL maximum korláttal ellátott text-tí-pusával, vagy a 255-ről 4294967295-re növelt maximális hosszúságú varchar-típussal. A character varying másik neve a varchar. A PostgreSQL character-e vagy más néven char, megegyezik a meghatározott hosszúságban szóközökkel kitöltött character varying PostgreSQL-típussal, vagy a MySQL char típusával, ha annak jóval hosszabb maximális hosszt állítunk be.

Enum

A MySQL enum-típusában egy számot tárolhatsz, amely egy kulcsszót képvisel, amit az enum-mező megadásakor definiáltál, és tökéletesen használható a rádiógombokhoz olyan űrlapokon, ahol a látogató több lehetőség közül csak egyet választhat. Tegyük fel, hogy a

honlapodon szavazatokat tárolsz a világ tíz legnépszerűbb emberére vonatkozóan, és ezúttal tényleg szeretnél takarékoskodni a hellyel, mert mind a 800.000.000 internetező szavazni fog. Definiálhatsz egy enum-mezőt a jelöltek neveivel, és egy szavazathoz csak egy bájtot fogsz használni, ahelyett, hogy a név minden karakterét eltárolnád, mivel a neveket csak egyszer tárolod a tábla mező'definícióiban és a mezők csak a névlistára vonatkozó indexet tartalmaznak. 1 és 255 közötti nevet tartalmazó listákhoz tartozó index számára elegendő 1 bájt, 65 ezer nevet tartalmazó lista két bájtban fér el, és az adatokat be lehet vinni az **enum** definíciójában használt névvel vagy közvetlenül az indexszámmal. A következő" kód az **enum** definícióját példázza; a mező neve **person** (személy), amely a top 10-es listán szereplő neveket tartalmazza a **votes-** (szavazatok) táblázatban.

```
create table votes {person enum('Nicole Kidman[1], 'Péter Moulding',
'Hannibál Lecter', 'Janet Jackson', ' Kevin Spacey',
'Cuba Gooding Jr.', 'Rasmus Lerdorf', 'Madonna', 'Yoda ',
' _ insert your name here _ '))
```

Set

A set révén egy bináris sztringet tárolhatsz, úgy hogy a megadott kulcsszavakat egy-egy bittel ki-be kapcsolhatod. Tegyük fel, hogy a szavazatokat fogadó weblapon a látogatók egy listán kipipálhatják azokat a tulajdonságokat, amelyek miatt a kérdéses jelöltre szavaztak, pl. „gondos", „jóképű", és „tenyésztésre alkalmas DNS-sel rendelkezik". A következő kód azt mutatja, hogy definiálunk egy **features** nevű mezőt, amely a tulajdonságok listáját tárolja. A features-ben minden tulajdonsághoz tartozik egy 0 vagy egy l-es, és mivel a set-mezők maximum 8 bájt (vagyis 64 bit) hosszúak lehetnek, egy set-mezőhöz 64 különböző tulajdonságot határozhatsz meg. A set-típus a HTML-űrlapok jelölőnégyzetéhez illeszkedik, ahol az emberek a megadott lehetőségek közül többet is kiválaszthatnak. A set mezőnként 64 lehetőséget fed le, és csökkenti az igényt a PostgreSQL **boolean** mezőtípusára (a MySQL megígérte, hogy a **boolean** szerepel a következő verzióban):

```
altér table votes add column features [1]handsome
    set('intelligent', 'caring[1], 'sharing', 'DNA')
```

Blobok

A Nagyméretű bináris objektum (Bmary Large Objects, **blob)** képek vagy más bináris adatok adatbázisban történő tárolására alkalmas, ami néha előnyös, de nem mindig. Nagyon jó példa erre a titkosított jelszó, ami csak 32 bájtot foglal az adatbázisban, de egy különálló fájlként a Linux ext2 fájlrendszerében 8192 bájt lemezterületet fogyasztana el. Teljes képeket az adatbázisban tárolni ennek épp az ellentettje: egy 100 000 bájtos képet a fájlrendszerben kell tárolni, és az adatbázisban csak egy hivatkozást tartunk.

Dátum és idő

A PostgreSQL-ben és a MySQL-ben hasonló, a legáltalánosabb dátum és időformátumok eltárolására alkalmas mezők vannak. Mindkettő elfogadja az egyszerű inputot, és egyszerű outputot szolgáltat, ha az egyik adatbázishoz írt PHP-kódodat a másikra alkalmazod az SQL átalakítása nélkül. Az 5.4 ábra a legnagyobb hasonlóságot mutató mezőtípusokat sorolja fel.

A MySQL **timestamp-je** a szabványos Unix időjelzés megfelelője, ami 1970-től 2037-ig érvényes, egy másodperces pontossággal, míg a PostgreSQL **timestamp-je** 1903-tól számol. A MySQL-é akkor frissül, ha beillesztesz egy rekordot, vagy megváltoztatsz egy mezőt egy rekordban, vagy **null** értéket adsz egy mezőnek. A PostgreSQL úgy tűnik, nem frissíti a timestamp-jét, vagyis egy tárolt eljárást kell írnod, vagy a PostgreSQL valamely másik különleges funkcióját kell használnod a feladat magoldásához. A PostgreSQL-ben időzónákat is megadhatunk, ami lehetővé teszi, hogy a helyi időt is átalakítsuk GMT/UCT időre.

5.4 táblázat Dátum és idő adattípusok a MySQL-ben és a PostgreSQL-ben

Type		ormatum		ostgre.
date	3	eeee-hh-nn	Igen	Igen
datetime	8	éééé-hh-nn óó:pp:mm	Igen	Igen
timestamp	4	ééééhhnnóóppmm	Igen	Igen
time	3	óó:pp:mm	Igen	Igen-
year	1	1901-től 2125-ig	Igen	Nem
interval	12	-178000000-tól +178000000-ig	Nem	Igen

A PostgreSQL year-típusa Kr. e. 4137-től 32 767-ig tart, míg a MySQL a 0000-tól 9999-ig tartó időszakot fedi le, és mind a **month-,** mind day-mezőkre megengedi a nulla értéket, vagyis részleges dátumokat is eltárolhatsz, pl. 0000-11-04 (megadja egy személy születésnapját, de nem árulja el a korát), vagy 1857-00-00 (pl. tudod, melyik évben született az egyik ősöd, de nem tudod, hogy mely hónapban, melyik napon).

A PostgreSQL **interval-ja** a Umxos időformátum kiterjesztése egy olyan tartományba, amely csillagászati számításokra alkalmas, vagy esetleg az adóhivatal is használhatja. Az interval 365 millió évet tárol másodperces felbontásban, és mivel az egy évre jutó másodpercek száma csökken (a föld lassul), az **intervaí** közelítő értéket használ a másodpercek eves szamara.

A PostgreSQL különleges típusai

A PostgreSQL-ben van néhány különleges formátumú mező, amelyeknek jó hasznát veheted, ám a speciális formátumú mezők csapdát is jelenthetnek. Ausztráliában négyjegyű irányítószámokat használnak, és a Posta minden évben felveti a hatjegyűre való áttérést; ezért Ausztráliában valóban csapdát jelenthet egy speciális adatformátumot használni az irányítószámokra. A PostgreSQL különleges típusait az 5.5 táblázat tartalmazza.

A **cidr** 4-es verziójú Internet Protocol (IPv4) címeket tárol *x.x.x.x/y* formátumban, ahol az *y* a címhez használt alhálózati maszkban lévő bitek száma. (Az internetes közösségben már bevezetés alatt áll az IPv6.) Az **inét** a **cidr** másik változata, amely minden cím minden helyén megenged nem nulla értékeket, vagyis egyedi gazdaszámítógép címek tárolására is alkalmas. A **cidr** a hálózatod előtagjának tárolására használható, pl. 192.168.3/8, és az **inet**-ben tárolhatod az egyedi számítógépek címet, pl. 192.168.3.4. A **macaddr** különböző for-

mátumú média acces control (MAC, médiahozzáférési vezérlés) címeket fogad el, pl. 10:03:44:0a:f3:92. A betűk lehetnek kis- és nagybetűk és az inputban a kettőspontot (:) kötőjel (-) is helyettesítheti, az outputban viszont szigorúan kettőspont (:) szerepel.

5.5 táblázat A PostgreSQL egyedi adattípusai

Típus		
cidr	12	Bármilyen IPv4 hálózati cím
inét	12	Bármilyen IPv4 hálózati vagy gazdacím
macaddr	6	Bármilyen hálózati illesztő MAC címe
point	16	Egy pont a kétdimenziós térben
line	32	Egy végtelen vonal a kétdimenziós térben
lseg	32	Egy véges hosszú vonal a kétdimenziós térben
box	32	Egy téglalap a kétdimenziós térben
path	változó	Egy nyitott vagy zárt út a kétdimenziós térben
polygon	32	Egy poligon a kétdimenziós térben
circle	24	A kör a kétdimenziós térben

A point egy kétdimenziós geometriában használatos *x,y* koordinátapárt tárol; a **line** két pár koordinátát tárol egy végtelen hosszú vonal számára; az **lseg** egy véges hosszú vonal kezdő-és végpontjának koordinátáit tartalmazza; a box-ban egy téglalap két szemközti csúcsa tárolható; a **path** több *x,y* koordinátapárt tartalmaz, amelyek egy nyílt vagy zárt utat adnak meg (a **polygon** párja); a polygon-ban tárolt *x,y* koordináták egy poligont írnak le; és a **circle** egy kör középpontjának koordinátáit és sugarát tartalmazza. Mindegyik mező 8-bájtos lebegőpontos számokat tárol, ezért biztosítani kell, hogy a megadott értékek a megfelelő határok közé essenek.

Azonosítók beillesztése

Mikor olyan rekordokat illesztesz be, amelyek **auto_increment-** (automatikus növelésű) mezőt (a PostgreSQL-ben **serial)** tartalmaznak, lehet, hogy szeretnéd megkapni az automatikusan növelt mező értékét. Ha egy számlát adsz az adatbázishoz, szeretnéd megkapni a számla automatikusan növelt azonosítószámát a számla tételeinek beillesztéséhez. Az automatikusan növelt értéket különféle módokon elérheted. A MySQL-ben benne van a **last_insert_id()** (utoljára beillesztett azonosító) SQL-függvény, a PHP pedig tartalmazza a **mysql_insert_id()** MySQL-utasítást. Ha a számlát egy számlákat tartalmazó táblába illesztted be, ezt az SQL-lekérdezést futtasd:

```
select last_ínsert id{) from invoices limit 1
```

A MySQL-hez ezt a PHP-kódot is használhatod:

```
$invoicenumber = mysql__insert_id {) ;
```

A mysql_insert_id() az auto_increment-mező legfrissebb értékét adja az aktuális szálban, ezért a mysql_insert_id()-utasítást közvetlenül a sikeres insert (beillesztés parancs) után kell tenni, mielőtt bármilyen más hozzáférés auto_increment-utasítást futtatna az adatbázison. Ha automatikusan növelt értéket nyersz ki egy adatbázisból, azonnal kérd le az értéket, és nézz utána az adatbázis dokumentációjában, hogy mi történik ha, több szkript egyidejűleg illeszt sorokat ugyanabba a táblázatba. Néhány adatbázis esetében ez csak úgy lehetséges, hogy frissítésre zárolod a táblát, visszanyered az azonosítót, majd megszünteted a zárolást.

A következőkben a PostgreSQL megfelelő utasítását, a pg_getlastoid()-ot mutatjuk be, amely egy kicsit különbözik a mysql_insert_id()-tó'l, mert a PostgreSQL belső azonosítóját, az oid-ot adja vissza. Az oid-ot nem használhatod hosszú távon, mert bizonyos dolgok hatására, pl. táblák újra betöltésére törlődik. A kód azt csinálja, hogy lekéri az oid-ot, és arra használja, hogy lekérje a jelen esetben invoice nevű sorrendmezőt az oid által meghatározott sorból. A pg_exec()-parancs a mysql_query() megfelelője; argumentumában egy adatbázis link azonosító és egy SQL-állítás szerepel, visszatérési értéke az eredmény azonosítója. Az eredmény azonosítóját a pg_fetch_array()-utasításban használjuk, hogy visszanyerjük az oid-nak megfelelő sort. Egy sort felfoghatunk egy tömbnek is, amelynek elemeit a mezők nevei azonosítják, hasonlóan a mysql_fetch_array()-hoz, ennek révén tudjuk megadni a $inviocenumber változónak a megfelelő értéket. Vannak más módok is a PostgreSQL-ben ennek a műveletnek az elvégzésére, pl. a sorrendmezőt kinyerhetjük a beillesztés előtt is, de ez áll legközelebb más adatbázisok módszereihez, és ez fog a legkevesebb problémát okozni, ha több, PostgreSQL-ben járatlan programozó dolgozik a kódon:

```
$oid = pg_getlastoid($result);
$result = pg_exec($link, "select invoice from invoices"
. " where~oid = " .
$oid) ; $row =
pg_fetch_array(Sresult) ;
$invoicenumber =
$row["invoice"];
```

A PHP-ban kétmilliárdig növelheted az egész számokat. Ha az automatikus növekedés miatt túlléped ezt az értéket, akkor a PHP hosszú numerikus (long numeric) mezőre vált. A MySQL és más adatbázisok nagyon nagy értékeket is megengednek az automatikusan növelt mezők számára, de nem biztos, hogy hiba nélkül átkonvertálják a PHP long numeric formátumára. Megkerülheted a problémát, ha az SQL-t használva nyered vissza az azonosítót, és az SQL-lekérdezés eredményének formájában adod át a PHP-nek.

Előfordulhatnak-e egyáltalán ilyen nagy számok? Képzeld el, hogy beüzemelsz egy rendszert, amellyel a fogyasztók elemezhetik a 25 leglátogatottabb honlap nézettségét. Minden oldalletöltés egy rekord, és minden oldalon több banner található, vagyis minden oldalletöltéshez több rekord is tartozik. Webhelyenként oldalletöltések tízmillióit regisztrálhatod, ami egy nap összesen akár százmillió rekordot is jelenthet az adatbázisban. Az automatikusan növelt számláló értéke akkor is tovább növekszik, ha folyamatosan törlőd a 7 napnál régebbi rekordokat.

Platformfüggetlenség

A PostgreSQL a Unix és a Linux minden, általam ismert verzióján elérhető'. A MySQL szintén, valamint Windows-on és Windows NT-n is. Ezért a MySQL platformfüggetlenebb és jobb választás olyan alkalmazások számára, melyeket több különböző szerveren akarsz használni. A MySQL úgy tűnik, minden platformon elérhető, amelyre Apache telepíthető, ami azt jelenti, hogy mindenütt elérhető, ahol PHP-t használhatsz.

A MySQL-nek egyetlen hátulütője van az eltérő platformokon: az adatbázisok és táblák nevei könyvtár- és fájlnevekké válnak, ezért a jelenlegi és a jövőben használni tervezett operációs rendszerek fájlnevekre vonatkozó megszorításait figyelembe kell venni. Azt tanácsolom, csak kisbetűs fájlneveket használj, mellőzd a szóközöket, aláhúzásokat, kötőjeleket és egyéb különleges karaktereket, és ne alkalmazz hosszú fájlneveket, hogy minden operációs rendszer értelmezni tudja az elnevezéseket.

A platformfüggetlenségbe az SQL is beletartozik, amely mindenhol fut, valamint a tetszőleges adatformátum olvasása és visszaadása. Mindkét adatbázis kezeli a nagyméretű bináris objektumokat **(blob),** ami problémákat okozhat egyes számítógépeken, valamint mindkettő hatékonyan tárolja az adatokat, ami gondot okoz a korlátozott tárhellyel bíró kis komputerek számára.

Adatbázis nyers adatokból

Az SQL különféle módokat kínál a külső fájlok beolvasására az insert-utasításon keresztül és a PHP tökéletesen alkalmas a fájlok átformázására, ha az input-fájlok nem illeszkednek tökéletesen az SQL insert-utasításának feldolgozásához. A szabványos SQL insert-et a „Rekordok beillesztése" című Gyors megoldásokban ismertetjük. A **mysql.com-on** és **postgresql.org-on** elérhető dokumentációban megtalálható, milyen fájlformátumokat tud olvasni az insert-utasításuk, és a 8. fejezetben ismertetjük, hogyan lehet a PHP-vel manuálisan formázni az adatokat az SQL insert-utasítása számára.

Adatbázisok és tömbök

A 3. fejezetben bemutatott PHP-tömbök széles körben kínálnak tárolási formátumokat és kulcsstruktúrákat, amelyek révén tetszőleges relációs adatot betölthetsz a memóriába és feldolgozhatsz a memóriából, ami ideális, ha egy szkript ugyanazt az adatot többször használja. Ahelyett, hogy ugyanazokat az adatokat több különböző formátumban vagy sorrendben olvasnád be, olvasd be egyszer, és végezd el a rendezést vagy a feldolgozást a tömbökből.

Indexelni vagy nem indexelni

Az adatbázis indexelése meggyorsíthatja az olvasást, mert az adatbázis-kezelő szoftver egyből a kért adatra tud ugrani, viszont lassíthatja az írást, mert új indexeket is fel kell írni

a lemezre. A legtöbb adatbázis biztosít valamilyen módot arra, hogy megállapítsuk, egy index segít-e egy adott lekérdezést valamint, lehetővé teszi az indexek hozzáadását és eltávolítását. A PHP a MySQL és a PostgreSQL összes különleges funkcióját támogatja. Ahhoz, hogy olyan kérdésekre válaszoljunk, mint pl. „Melyik lekérdezés a leggyakrabban használatos?", a naplófájlok mélyére kell ásni, és a PHP szinte bármilyen formátumot képes olvasni, beleértve a naplófájlokat. Ki kell számolnod az olvasási sebesség növekedéséből fakadó előnyöket és az írásból fakadó hátrányokat, és a PHP segít a számítások automatizálásában, ha egyszer eldöntötted, melyek a legfontosabb mutatószámok a saját honlapod számára. Mind a MySQL-lel, mind a PostgreSQL-lel könnyedén létrehozhatsz indexeket, megvizsgálhatod a honlap teljesítményét és törölheted az indexeket.

Kapcsolatok

A PostgreSQL oly módon támogatja a kapcsolatokat, hogy az adatbázist igazi relációs adatbázissá teszi - ez rendkívül hasznos bizonyos helyzetekben. A relációk összekapcsolják a táblákat, ezáltal összhangban tartják az adatokat, de egy rossz relációs szabály igencsak megnehezíti az adatbevitelt. Ha csak most ismerkedsz az adatbázisokkal, vagy teljesen új adatokkal dolgozol, akkor kezdetben hanyagold a relációkat, olvasd be az adatokat táblákba, ellenőrizd, hogy az adatok megfelelnek-e minden kritériumnak, és ezután kísérletezz relációk hozzáadásával. Ha már átlátod az adataidat és van valamekkora tapasztalatod a relációk tervezésében, akkor megítélheted, hogy a többletköltségek megérik-e a megnövelt adatintegritást a PostgreSQL-ben.

Az adatok betöltésének és vizsgálatának legegyszerűbb módja az előre megírt phpMyAdmin (letölthető a www.phpwizard.net-ről) és pgMyAdmin (letölthető a www.greatbridge.org-ról) használata. Mindkettő előre megírt elemeket tartalmaz és ha ezek nem passzolnak az igényeidhez, akkor nyers SQL-parancsokat is begépelhetsz.

ODBC

Az Object Database Connectivity (ODBC) valamelyes adatbázis-függetlenséget biztosít, ha különböző típusú adatbázisokkal dolgozol; ennek ára az egyes hozzáférések viszonylagos lassúsága. Ha egy adatbázissal dolgozol, pl. MySQL-lel, és átkonvertálod egy másik típusra, például PostgreSQL-re, akkor a legjobb megközelítés, ha mindkét adatbázishoz a PHP saját utasításait használod, és gondosan ellenőrződ, hogy az egyes hozzáférések optimális sebességgel működnek-e. Ha többnyire egy adatbázissal dolgozol és alkalmanként más típusú adatbázisokat is használsz, akkor ezekhez célszerű lehet ODBC-vel kapcsolódni, mert ez egyszerű, és az ODBC hátrányai nem befolyásolják a fő adatbázisodat.

<m

Gyors megoldások

Kapcsolódás az adatbázishoz

Az első példa a minimális kódot mutatja, amivel egy helyi MySQL-adatbázishoz kapcsolódhatsz. A további sorok egy hálózaton belüli másik szerverhez való kapcsolódást mutatnak. Egy szerverhez a mysql_connect()-utasítással kapcsolódhatsz, amelynek argumentumában a szerver neve az első érték; ha ezt nem adod meg, akkor alapértelmezett **localhost**-ot használja. A szerver neve mellett a portot is meghatározhatod az alábbi formában: sqlserver:3306. A mysqlconnectQ alapértelmezése 3306, amit nem szükséges megadni, hacsak a rendszergazda nem állít be egy másik portot. A felhasználónév alapértéke a webszerverfolyamat tulajdonosának felhasználóneve, a jelszó alapértéke pedig üres, tehát el is hagyhatod, ha a MySQL ugyanazon a gépen fut, és a honlapod az egyedüli honlap a gépen. Az értékek a $database-tömbben vannak összegyűjtve, vagyis egy sima fájlból ís be lehet tölteni őket.

A @ jel a **mysql_connect()** előtt egyszerűen kikapcsolja az alapértelmezett hibaüzenetet, így elcsípheted a hibát, és olyan hibaüzenetet jeleníthetsz meg, amely sokkal hasznosabb számodra, és a honlapod látogatói is jobban megértik. A bemutatott alapvető hibaüzenet csak illusztrációs célokat szolgál. A gyakorlatban kapcsolatfelvételt lehetővé tévő információk szerepelhetnének, vagy átirányítás egy másik oldalra, esetleg egy levél a weboldal fenntartójának.

```
$connect = mysql_connect();

$database["server"] = "sqlserver";
?database["user"] = "sqluser";
$database["password"] = "tx04dd";
if (!$connect = @mysql__connect ($database [ "server" ],
    $database["user"],
    $database["password"]))
    {
    print("MySQL problem. Connect failed to server: " .
        $database["server"]);
```

Az előző kód csatlakozott egy MySQL-szerverhez; a következő kód révén a MySQL csatlakozik egy adatbázishoz a mysql_select_db()-utasítással. Az adatbázis neve **movies,** ez az első paraméter a **mysql_select_db()**-utasítás argumentumában, a második pedig az opcionális kapcsolatazonosító, ami a mysql_connect()-parancs visszatérési értéke. A kapcsolatazo-nosító sok MySQL-utasításban opcionális paraméter, és ha nem adod meg az értékét, akkor az utasítások az éppen aktuális kapcsolatot használják - mindez kielégíti az igényeidet, ha a szkriptjeid csak egy kapcsolatot használnak. Ha nem sikerült létrehozni kapcsolatot, a **mysql_select_db()** visszatérési értéke hamis, vagyis hasznos hibaüzenetet készíthetsz:

```
$database["database "] = "movies";
if(!@mysql_select_db($database["database"] , Sconnect))
    {
   print("MySQL problem. Failed to select " . $database["database "]);
```

A PostgreSQL-hez van egy utasítás, a pg_connect(), ami a mysql_connect() és a mysql_selest_db() kombinációjának felel meg: argumentumában az összes paraméter egy kapcsolódási sztringként szerepel. A következő példa a helyi szerveren lévő templatel nevű adatbázishoz kapcsolódik (a templatel-et a PostgreSQL telepítője hozza létre) peter felhasználónévvel, meghatározott host-on, passworddel és porton. Mikor ezt teszteltem egy PostgreSQL-szerveren, amely ugyanazon a gépen futott, mint az Apache webszerver, akkor meg kellett adnom a host-ot, de nem kellett megadni a passwordöt és a portot:

```
$database["database"] = "templatel";
$database["host"] = "localhost";
$database["port"] = "5432";
$database["user"] = "peter";
$database["password"] = "tx04dd";
$connect = pg connect("dbname=" . $database["database"]
    . " host=" . $database["host"]
    . " port=" . $database["port"]
    . " user=" . $database["user"]
    .   "  password="   .   $database["password"] ) ;
```

Vedd észre hogy kihagytam a @-ot a pg_connect() elől és a hibaellenőrzés is elhagytam. A @ kikapcsolja a PostgreSQL hibaüzeneteit. Abban a verzióban, amelyben ezt a kódot kipróbáltam, a pg_errormessage() utasítás nem adott vissza semmilyen hibaüzenetet, ami azt jelenti, hogy nem győződhetsz meg arról, hogy a PostrgeSQL kapcsolatod száz százalékosan tökéletes és lehet hogy a felhasználók alkalmanként ronda üzeneteket kapnak, ha nem jön létre a PostgreSQL kapcsolat. Milyen rondákat? Az 5.1 ábrán látható egy hibaüzenet. A pg_errormessage() akkor működik, ha már létrejött a kapcsolat, és a kapcsolat legutolsó hibaüzenetét rögzíti.

Waniing: Unable to connect to PostgreSQL server: connectDBStartQ — socketQ failed: errno=0

5.1 ábra Sikertelen pg_connect() ronda hibaüzenete

Mindkét adatbázis támogatja az állandó kapcsolatokat, amelyek lehetővé teszik a szkript számára, hogy a korábbi kapcsolatot használja, ugyanavval a felhasználóval, jelszóval és szerverrel, ezáltal csökkentik az erőforrás-igényes műveleteket. A MySQL állandó kapcsolatai teljesen megbízhatóak, de a PostgreSQL-lel felmerülhetnek problémák, ha újraindítod anélkül, hogy az Apache-ot is újraindítanád. A MySQL-ben a mysql_connect()-et cseréld le mysql_pconnect()-re, PostgreSQL-ben pedig a pg_connect()-et pg_pconnect()-re.

Az adatbázisok listázása

Ha szeretnéd megtudni, milyen adatbázisok elérhetők a szervereden, a MySQL támogatja az SQL **show** database-parancsát (lásd a példakód első sorát), amely megmutatja azokat az adatbázisokat, amelyekhez van jogosultságod. Ebbe beletartozhatnak azok az adatbázisok is, amelyeket jogosult vagy látni, de nincs jogosultságod olvasni őket. A kód feltételezi, hogy már kapcsolódtál az adatbázishoz az előző, „Kapcsolódás az adatbázishoz" című részben leírt módon:

```
$database["sql"]    =    "show    databases";
if($result    =    @mysql_query($database["sql"] )
)
    {
    while($row = mysql_fetch_row($result) )
        {
        print("<br>" . $row[0]);


else

    print ("Query error: " . mysql_errno() . ", " . mysql_error(} .
        " using sql " . $databaset"sql"] ) ;
```

A kód az SQL-utasításokat beteszi egy általános adatbázis-elérési tömbbe, majd továbbadja azt a **mysql_query()**-függvénynek. A szóban forgó függvény egy eredményindikátort ad vissza, aminek értéke hiba esetén hamis. A mysql_fetch_row()-utasítás egyszerre egy sort olvas ki az eredményből és hamisat ad vissza, miután az utolsó sort is elolvasta. Ha hiba lép fel az adatbázis elérésében, a kód egy hibaüzenetet ad; de ha az adatbázis elérése sikeres és egyetlen sort sem eredményez az SQL-lekérdezés, akkor nem lesz hibaüzenet, és semmilyen adat nem jelenik meg. Az én munkaállomásomon kapott eredmény a következőkben látható. A movies-adatbázist néhány Gyors megoldások részben használjuk példaként, a mysql-adatbázist a MySQL használja adminisztrációs célokra, és a test-adatbázist a MySQL hozza létre tesztelési célokra:

```
movies
mysql
page test
```

A PostgreSQL-ben az adott szerveren lévő adatbázisok a **pg-databases** nevű táblában találhatók, ami az adminisztrációs adatbázisban van elrejtve. Az adatbázisok listáját a **select-pa-**ranccsal jelenítheted meg, mint a következő kód első sora mutatja. A PostgreSQL adminisztratív táblái a táblák felsorolásában saját magukat is tartalmazzák, ezért akik adminisztratív jogokkal rendelkeznek, plusz táblákat látnak, amelyeket a példaprogram második select-utasításával lehet kizárni. A példaprogramban, hasonlóan az előzőhöz, a pg_exec() hajtja végre a lekérdezést, és a **pg_fetch_row()** soronként adja az eredményt. A PostgreSQL-ben egy hibaüzenet-utasítás van, a **pg_errormessage(),** ez helyettesíti a MySQL **mysql_error()** és **mysql_errno()** utasításait.

```
select datname from pg-databases
select datname from pg-databases where tablename not üke 'pg%'

$database["sql"] = "select datname from pg database";
if($result = @pg_exec($database["sql"] ) )
    {
    $rows = pg numrows($result) ;
    if($rows)
        {
        for($i = 0; $i < $rows; $i++)
            {
            $row = pg_fetch_row($result, $i) ;
            print("<br>"   .   $row[0]);


    else

        print{"<br>Zero results. Error: " . $error .
            "<br>from sql: " . $database["sql"] ) ;


else

    print("Pg_exec()    error:    "    .
        pg_errormessage .    "<br>using    sql:    "    .
        $database["sql"] ) ;
```


A PostgreSQL **pg_fetch_row()**-utasításának a sor indexét is meg kell adni. A sorok 0-val kezdődő indexszel vannak ellátva, ezért a **pg_fetch_row()-t while()** helyett for()-ciklusba kell ágyazni. A ciklust annyiszor hajtjuk végre, amennyi a sorok száma, ezt pedig a pg_numrows() segítségével kapjuk meg. Az *előző* kódot abban a tekintetben is kiterjesztettük, hogy elválasztjuk azokat az eseteket, mikor a pg_exec() hibásan fut le, azoktól, mikor működik, de nulla számú sort ad eredményül. Az **if($rows)** utasítás vizsgálja a nullasor feltételt és egy speciális hibaüzenetet eredményez, vagyis ellenőrizheted az SQL logikai hibáit is. Az alábbiakban látható az eredmény az általam készített test- és movies-adatbá-zisokkal, plusz a PostgreSQL telepítő által létrehozott két template-tel:

```
test
templatel
templateO
movies
```

A PHP MySQL-utasításai közt szerepel egy speciális utasítás, a **mysql_list_dbs(),** amely felsorolja az adatbázisokat a következő kódban bemutatott módon. A **mysql_list_dbs()** beolvassa a **$connect** opcionális paramétert, csatlakozik a MySQL-szerverhez, és az eredményt felsoroló listára mutató pointert ad eredményül; majd a **mysql_num_rows()** megadja az eredmény sorainak számát, és a for()-ciklus végiglépked a sorokon. Az egyes sorokat a **mysql_db_name()**-utasítással kapjuk meg, amelyet kifejezetten adatbázisok nevének visszanyerésére írtak.

```
$result    =    mysql    üst    dbs ($connect) ;
```

```
írows = mysql_num_rows($result) ;
for($i = 0; $i < $rows; $i++)
    {
    print("<br>"    .    mysql_db_name($result,
```

A sorokat egy $x nevű objektumban is megkaphatod a mysql_fetch_object()-utasítással és az adatbázis nevéhez a $x-> Database paranccsal férhetsz hozzá. Vedd észre, hogy az objektumok tulajdonságainak nevében számítanak a kis- és nagybetűk, vagyis a Database-t nagy ű-vel kell írni:

```
$result    =    mysql_list_dbs();
while($x    =    mysql_fetch_object($result) )
    {
    print("<br>" . $x->Database);
```

Most már képes vagy kapcsolódni egy MySQL- vagy PostgreSQL-szerverhez, és megkaphatod a hozzáférhető adatbázisok felsorolását. A következő Gyors megoldásból megtudhatod, hogyan találhatsz meg egy táblát és egy mezőt, valamint hogyan hozz létre új táblát és adatot.

Adatbázistáblák megjelenítése

Ha szeretnéd megtudni, milyen táblák vannak az adatbázisban, a MySQL támogatja az SQL show table-parancsát (lásd a példakódot), amely megmutatja az adatbázisban található táblákat. A kód feltételezi, hogy már kapcsolódtál az adatbázishoz (lásd a korábbi „Kapcsolódás az adatbázishoz" Gyors megoldásokat) és hogy az adatbázist movies-nak hívják:

```
$database["database"]    =    "movies";
$database["sql"]    =    "show    tables    from    "    .    $database["database" ];

if($result    =    @mysql_query($database["sql" ] ) )

    while($row    =    mysqlfetchrow($result)

        ) print("<br>"    .    $row[0]);

    }
else
    {
    print("Query error: " . mysql errno()  . ", " . mysql error() .
        " using sql " . $database["sql" ] ) ;
```

A kód a mysql_query()-utasításon keresztül továbbítja az SQL-parancsokat, ami hiba esetén hamisat ad eredményül, majd a mysql_fetch_row() soronként beolvassa az SQL eredményét és hamisat ad eredményül, miután beolvasta az utolsó sort. Egy adatbázis elérési hiba hibaüzenetet eredményez, és sem hibaüzenet, sem más adat nem kerül megjelenítésre, ha az adatbázis nem tartalmaz adatokat. A movies-adatbázis a következő eredmény adta; ezeket a táblákat fogjuk használni a további példákban:

```
cast
director
movie
person
producer
title
```

A következőkben az *előző* kód PostgreSQL megfelelőjét láthatjuk. A kiválasztást az adminisztrációs táblában **(pg_tables)** hajtottuk végre, amelyik valójában nem is tábla, hanem a pg_class-tábla nézete. Vedd észre, hogy nincs megadva adatbázis. A **pg_class** egy sor definíciót tartalmaz táblákra és indexekre vonatkozóan, de ezek nem kifejezetten úgy írják le az adatbázist, ahogyan a MySQL táblák az adatbázishoz viszonyulnak.

A kód további része hasonló ahhoz, ahogyan az adatbázisok neveit az *előző*, „Adatbázisok felsorolása" részben megjelenítettük, de eltér az ekvivalens MySQL-kódtól. Noha lehetséges közös kód írása a két adatbázishoz, néhány alapvető különbség a kódban eltemetve nehezen érthetővé válna. A PostgreSQL **pg_fieldname()**- és **pg_fetch_row()**-utasítása egyaránt indexet használnak a pg_exec()-ből származó eredményhez, ezért mindkettőhöz for()-ciklust kell használni **while()** helyett, és szükség van egy számlálóra a ciklus leállításához. A **pg_fieldname()** számlálója a táblák száma a **pg_numfields()-ből,** a **pg_fetch_row()** számlálója pedig a **pg_numrows()** eredményéül kapott sorok száma. A táblázat formázása megegyezik a korábbi Azonnali megoldással:

```
$database["sql"] = "select * from
pg_tables"; if($result =
@pg_exec($database["sql"] ) )

    $rows = pg_numrows($result) ;
    if($rows)

        print ("<table border=\"l\"xtr>" ) ;
        $fields = pg_numfields($result);
        for($f = 0; $f < $fields;
        $f++)

            print("<td><em>" . pg_fieldname($result, $f) .

        print("</tr>");
        for($i = 0; $i < $rows; $i++)

            $row = pg fetch row ($result, $i);
            print ("<tr>");

            for($f = 0; $f < $fields; $f++)

            print ("<td>" . $row[$f] . "</td>");

    print("</tr>") ; print{"</table>") ; else

        print("<br>Zero results. Error: " . $error .
            "<br>from sql: " . $database["sql"]) ;
```

```
else {
    print{"Pg_exec{)
        error: .    "<br>using      .   pg
        sql:    "                   errormessage()
                                    $database["sql"]);
```

A PHP a mysql_list_tables()-utasítást is támogatja az adatbázis tábláinak kinyerésére. A mysql_num_rows() a beolvasandó sorok számát adja, és a for()-ciklus végigmegy a sorokon; az egyes sorokat a mysql_tablename()-utasítással kapjuk meg, amelyet kifejezetten a táblák nevének kinyerésére írtak. Azért használunk for()-ciklust while() helyett, mert a mysql_tablename() számára egy indexérték, a $i szükséges. Minden más ugyanaz, mint az előző példában:

```
$database["database" ] = "movies";
$result = mysql_list_tables($database["database"]);
$rows = mysql num rows($result);
for($i = 0; $i < $rows; $i++)
    {
    print("<br>"    .    mysql__tablename($result,    $i) ) ;
```

Egy másik mód a táblák felsorolására, ha a mysql_fetch_row()-t egy while()-ciklusba ágyazva a list()-utasítással kapjuk meg a sorok tartalmát, mint a következő kódban látható. Ez a technika közelebb áll ahhoz, ahogyan az előző Gyors megoldásokban az adatbázisokat felsoroltuk, és látszólag feleslegessé teszi a mysql_tablename()-utasítást. De nem tudhatjuk, hogy a jövőbeli PHP- és MySQL-verziók mysql_list_tables()-utasítása hogyan fog működni, ezért a mysql_tablename() csak erősíti a verziófüggetlenséget.

```
$database["database"] = "movies";
$result = mysql_list_tables($database["database"]);
while (üst ($table) = mysql_fetch_row ($result) )
    {
    print ("<br>"    .    $table);
```

Táblák mezőinek megjelenítése

Ha meg szeretnéd találni egy adatbázistábJa összes mezőjét, az SQL show fields-parancsa a MySQL-ben megmutatja azokat, csakúgy, mint a show columns. A relációs adatbázisok nyelvén az oszlop *[column)* a megfelelő kifejezés, de a mező *(field)* és az oszlop szavak a legtöbb adatbázisban felcserélhetőek. Az első példa megmutatja a mezőinformációk kinyerésének két általános SQL-megoldását, majd ezeket egy MySQL- és egy PostgreSQL-ver-zió követi. Az adatbázis a movies-, a tábla a person-, és az eredmények egy új weboldal tesztelésére szánt MySQL-adatbázisból származnak:

```
show columns frotn person from movies show
fields from person from movies
```

A MySQL tábláinak felsorolásához kapcsolódj a MySQL-hez az első Gyors megoldásokban bemutatott módon, és futtasd a következő kódot. A **mysql_list_fields()** az adatbázis egy táblájának mezőit sorolja fel. A **mysql_num_fields()** a mezők számát biztosítja az eredmény olvasásához, a for()-ciklus pedig végigmegy az eredményen. A mezők egyes tulajdonságait a **mysql_field_name()-, mysql_filed_type()-, mysql_filed_len()-** és mysql_filed_flags()-utasítások adják:

```
$result = mysql üst fields{"movies",
"person"); $rows = mysql_num_fields($result) ;
for($i = 0; $i < Srows; $i++)

    print("<br>" . mysql_field_name($result,
        . " type: " . mysql_field__type(íresult, $i)
        . ", length: " . mysql_field_len($result, $i)
        . ", flags: " . mysql_field_flags($result, $i));
```


Az eredmény:

```
entry type: int, length: 10, flags: not_null primary_key unsigned
auto_increment
updated type: timestamp, length: 14, flags: not_null unsigned zerofill
timestamp
name type: string, length: 60, flags: not_null
realname type: blob, length: 255, flags: not_null blob
born type: date, length: 10, flags: not null
comment type: blob, length: 65535, flags: not_null blob
```

A következő példában áttekinthetőbb formában jelenítjük meg az információt azáltal, hogy egy **<table>** tag-be ágyazzuk. Az oszlopok szegélyeinek megjelenítéséhez a kód a border="3" paramétert használja a **<table>** HTML-tag-ben. A sorok megjelenítése előtt a kód beilleszti a **<table>** tag-et és egy fejlécsort, majd a ciklus végigmegy az adatsorokon, és végül beilleszti a záró **</table>** tag-et. A ciklusba ágyazott kód beilleszti a táblázat egyik sorát, annak összes oszlopával:

```
$result = mysql_list_fields{"movies" , "person");
$rows = mysql num fields($result);
print ("<table border=\"3\"xtr><td><em>Field</em></td>"
    . "<tdxem>Type</em></td>"
    . "<td><em>Length</em></td>"
    . "<td><em>Flags</em></tdx/tr>") ;
for($i = 0; Si < $rows; $i++)

    print("<tr><td>" . mysql^field_name(Sresult, .
        "<td>" . mysql_f ield__type ($result, $i) .
        "<td>" . mysql_field len($result, $i) .
        "<td>" . mysql_field_flags($result, $i)     . "</tdx/tr>")

print("</table>");
```

A táblázat az 5.2 ábrán látható.

A PHP PostgreSQL-hez tartozó megfelelő utasítása sokkal bonyolultabb, mert a PostgreSQL a mezőneveket egy általános osztálytáblában tárolja, a mezőtípusokat külön típustáblában, a mezők tulajdonságait pedig külön tulajdonságtáblában. Először vess egy pillantást mindhárom táblára, hogy lásd, milyen adatokat tartalmaznak: futtasd le háromszor az előző Azonnali megoldás PostgreSQL-példaprogramját, behelyettesítve a három különböző tábla nevét. A pg_class-tábla mintája az 5.3 ábrán látható, a **pg_type** az 5.4 ábrán, a **pg_attribute** pedig az 5.5-en.

\Field	*\Type*	*Length*	*\Flags*	
entry	int	10	jnot null	primary key unsigned autó incrementj
updaled	timestamp	14	jnot null	unsigned zerofill timestamp
jname	**jslring**	**60**	**|not null**	
realname	blob	255	notnull	blob
bom	daie	10	not_null	
comment	blob	*65535*	inot_null	blob

5.2 ábra A person MySQL-táblázat mezői

relname	*\reltype*	*\relowner*	*\relam*	*\relfilenode*	*\relpages*	*\reltnples*
		1000	|1247			126
pg attribute	**i|75**					**573**
|pg_shadow		1000	**1260**			
pgclass		fíőöö"		![1259		**80**
pg toast 1215 idx	**10**					10

5.3 ábra A pgclass PostgreSQL-táblázat mintája

typname	*typov/ner*	*typlen*	*typprtlen*	*\typbyval*	*\typtype*	*\typisdefined*	*\typdelim*
:bool	**1000**	**1**	**\l**	**/|t**			
bytea	**1000**	**-1**	~	**r**			
jchar	**1000**		**1**	**l£**	t		
inamé	**1000**	**32**	**32**	**'ff**	!t		
jiotS	**1000**	**8**	**20**	**;|f**	**it**		

5.4 ábra A pg_type PostgreSQL-táblázat mintája

lattrelid	*mtname*	*\ftttypid*	*\titdispersion*	*\ittlen*	*\pttnxon*	*fatnelems*	*\attcacheqff*
			132				
		19				**l-l**	
			[1247	(typname			
		23	**1. |1247**	jtypowner			
1247	ryplen		21		-1		
1247	jtypprtlen			[0Tl9229			
1247	**jtypbyval**	21			**i l j H:**		
			10.198765				
		16	[0.518141	!5 íjl			

5.5 ábra A pg_attribute PostgreSQL-táblázat mintája

Nyilván nem szeretnéd keresztülrágni magadat a nyers adatokon, ezért használd a PostgreSQL fejlett SQL-jét az adatok különböző táblákból való összeszedésére és tetszőleges megjelenítésére. A következő kód az előző kódot használja a táblázat megjelenítésére is, az SQL-t viszont felváltottuk egy másikkal, amely mindhárom táblából kiválasztja a mezőket. Egy SQL-szelektben a from-előírásban több táblanevet is megadhatsz, míg a where kulcsszó révén bizonyos feltételeknek eleget tevő rekordokat választhatsz ki a sorok közül. Ez esetben a pg_class-tábla oid-jének - a PostgreSQL belső azonosítójának -meg kell egyeznie a pg_attribute-tábla attrelid-mezőjével, és a pg_type oid-jének meg kell egyeznie a pg_attríbute atttypid-mezőjével. Relációs fogalmakkal a pg_attribute összekapcsolja a pg_type- és a pg_class-táblákat, e kapcsolat nélkül a részek semmilyen jelentést nem hordoznak:

```
$database["sql"] = "select pg_attribute.attnum,"
    ." pg_attribute.attname as field,"
    ." pg_type.typname as type,"
    ." pg_attribute.attlen as length,"
    ." pg_attribute.atttypmod as variablelength,"
    ." pg_attribute.attnotnull as notnull,"
    ." pg_type.typprtlen as printlength,"
    •" pg_class.relname"
    ." from pg__class, pg_type, pg_attribute"
    ." where pg_attribute.attrelid = pg_class.oid"
    ." and pg_attribute.atttypid = pg_type.oid"
    ." order by pg_attribute.attnum";
```

Az SQL-szelekt az 5.6 ábrán látható eredménye még mindig nem egészen jó. Vannak negatív attnum-értékek is — ezek olyan sorokat jelentenek, amelyek nem relevánsak a mi táblafiníciónk szempontjából. A PostgreSQL adminisztrációs tábláiban sok negatív érték található, amelyek különleges jelentéssel bírnak - az 1 gyakran hamisat vagy használaton kívülit jelent, ellentétben a **PHP** megközelítésével, ahol minden nem nulla szám igazat jelent, vagyis ezeket a mezőket kisebb mint nulla feltétellel lehet vizsgálni.

\attnu	i\field	tyengtk	variable	■>tnull	\relname
-7	**tableoid oid**	¡*		**10**	**ípg_ipl**
\-f'	tableoid **oid**		-1	**10**	**■|pg_toást_1216**
-7	tableoid **oid**	k	-1	**10**	jpg_group
-7	tableoid oid	|4	-1	**10**	|pg toast **17058**
|-7	**tableoid oid**	K		**10**	**|pg_attribute**

5.6 ábra A pg_class, pg_type és pg_attribute első egyesítésének mintája

Adj hozzá egy további elemet az SQL where-előírásához, hogy azokat a rekordokat kapd vissza, amelyek attnum-értéke pozitív. Adj hozzá egy további feltételt is annak érdekében, hogy a mezők a megfelelő táblából legyenek kiválasztva. Ezt a következő SQL-példában látható pg_class.relname használatával teheted meg. Mivel ez a kód működik, az attnum és a **relname** megjelenítése elhagyható. Először futtasd le, hogy megbizonyosodj arról, hogy

működik, aztán olvasd el a PostgreSQL dokumentációját, és kísérletezz az adminisztrációs táblák más értékeinek megjelenítésével is:

```
$database["table"] = "person";
$database["sql"] = "select pg_attribute.attname as
    field," " pg type.typname as type,"
•   " pg_attribute.attlen as length,"
.   " pg_attribute.atttypmod as maximumlength,"
.   " pg_attribute.attnotnull as notnull,"
.   " pg_type.typprtlen as printlength"
.   " from pg_class, pg_attribute, pg_type"
.   " where pg_class.relname = '" . $database["table"] .
"'"
.   " and pg attribute.attnum > 0"
.   " and pg_attribute.attrelid = pg_class.oid"
.   " and pg_attribute.atttypid = pg_type.oid"
.   " order by pg_attribute.attnum";
```

Az 5.7 ábrán a végső felsorolás látható, bár ez még mindig nem tartalmazza, melyik mezőé az elsődleges index, vagy más, PostgreSQL-ben elérhető tulajdonságokat. Ha elégedett vagy az SQL-szelekttel, lementheted egy nézetként, amelyhez aztán egyszerűbb SQL-pa-ranccsal férhetsz hozzá. A nézetek definiálása a PostgreSQL-ben túlmutat e könyv keretein, de a PostgreSQL fejlesztői minden verzióhoz újabb adminisztratív nézeteket dolgoznak ki. Lehet, hogy egy tábla mezőinek nézete lesz a következő projektjük.

\field	*type*	*lengi*	*maximumlength\iotnull*	
entry	**iní4**	**4**	**-1**	**10**
updated	**tiraestamp**		**: A _ ... _ . ***	**47**
name	varchar	**-i**	**64 ..... ‖t**	**-1**
^ealname	varchar	**-i**	**259 .... J L .**	**-1**
(bom	**date**	**♦**	**-i .**	**10**
comment	text	**-i**	- i	**-1**

5.7 ábra A mezők listája a person PostgreSQL-táblában

Most már birtokában vagy a kódnak, amellyel megjelenítheted a PostgreSQL-táblák mezőit, ismered a táblák összekapcsolására szolgáló SQL-utasítást, és van tapasztalatod az adminisztratív táblákkal - tehát fel vagy vértezve egy hosszú és boldog PostgreSQL-használói pályafutásra. Plusz mindezt MySQL-ben is meg tudod csinálni.

Táblák adatainak megjelenítése

Ha szeretnéd egy tábla összes adatát megtalálni, de nem ismered a tábla mezőit, akkor azt szeretnéd, hogy a tábla összes mezője automatikusan bekerüljön a kiválasztásba. Az SQL select "'-operátora egy tábla összes oszlopát kiválasztja. A legtöbb adatbázis abban a sor-rendben adja vissza a mezőket, ahogyan definiálták azokat. A következő példa (és az összes példa ebben a Gyors megoldások részben) azt feltételezi, hogy a movies-adatbázisban a

person-táblát használjuk, amit először MySQL-ben hoztam létre, azután PostgreSQL-ben, és megpróbáltam az előbbihez a lehető legközelebb álló típusokat választani (a PostgreSQL-ben saját típusokat is lehet definiálni, amibe inkább e könyv és a PostgreSQL telepítője által létrehozott összes dokumentáció elolvasása és néhány hónapos PostgreSQL gyakorlat után érdemes belefogni):

```
select * from movies.person
```

Vedd észre, hogy az SQL-ben két nevet egy pont köt össze; az első név az adatbázisé, a második a tábláé. Az adatbázis nevét az adatbázis-hozzáférést megnyitó SQL-ben vagy PHP-utasításban is megadhatod. A MySQL a mysql_select_db()-utasítást biztosítja az adatbázis kiválasztására, mielőtt a megszokott **mysql_query()**-parancsot használnád. A PHP-ban szerepelt egy **mysql_db_query(),** amelyben paraméterként megadhattad az adatbázist, de az elkülönített **mysql_select_db()** és **mysql_query()** a helyes megközelítés. Kérlek, cseréld le, ha valahol találkozol a régi típusú kóddal.

Ha már gondtalanul meg tudod jeleníteni az egész táblát, akkor továbbmehetsz, és a *-ot lecserélve tetszőleges oszlopokat jeleníthetsz meg. Nem számít, hogyan adod meg az oszlopokat, vagy milyen adatbázist használsz, az oszlopokat visszaadó utasítások néhány egyszerű szabályt követnek: az oszlopokat abban sorrendben kapod vissza, ahogyan megadtad a kiválasztásban, és ha *-ot használtál, akkor olyan sorrendben, ahogyan az adatbázis felsorolja őket (többnyire úgy, ahogyan definiálva vannak), vagyis a sorrend megjósolható, így a sorokat számokkal indexelt tömbként is kezelheted és a megjelenítheted a mezőket a megfelelő címmel:

```
select narae, born from movies.person
```

Az egyedüli probléma az oszlopok indexszámokkal való elérésénél akkor jelentkezik, hogyha az adatbázisban módosítod az oszlopokat, mert akkor a számok is változnak (mikor *-ot használsz), vagy amikor módosítod az szelekthez használt mezőlistát. Ebben a Gyors megoldások részben bemutatott kódok azt a célt szolgálják, hogy automatizáljuk az adatok visszanyerését. A következő példakódban a **mysql_query()** kiválasztja a **person-tábla** oszlopait, és megjeleníti az adatokat egy fejléc nélküli nyers táblázatban:

```
$database["database"] = "movies";
$database["table"] = "person";
$database["sql"] = "select * from ";
if (isset($database["database"]) and strlen($database["database"]))
    {
    $database["sql"] .= $database["database"] . ".";
    }
$database["sql"] .= $database["table"];
$result = mysql_query($database["sql"]);
print ("<table border=\"l\">"); while($row
= mysql_fetch_row($result))
    {
    $fields = mysql_num_fields($result) ;
    print ("<tr>") ;
    for($i = 0; $i < $fields; $i++)
```

```
        print("<td>" . $row[$i]  .

    print("</tr>") ;

print("</table>") ;
```

Az 5.8 ábrán látható eredmény nem kifejezetten tetszetős. Egy finomítási lehetőség fejléc hozzáadása a mezőkhöz, legyen ez az első javítás a kódban.

jjf~ 00000000000000 Tom Berenger Thomas Michael Moore 1949-05-31
jJ2~ 00000000000000 MimiRogers Miriam Spickler 1956-01-27
ipT^0000000000000 Lorraine Bracco 1955-10-22
>4~ 00000000000000 JeiryOrbach Jerome Bemard Orbach 1935-10-20
, *5* 00000000000000 John Rubinstein
!|e" 00000000000000 Ridley Seott 11937-11-30
:l7~20010422203132 Russell Crowe Russell Ira Crowe 1964-04-07 The world's greatest actor.
IJÍTÍ20010423 083 731 Christopher Walken ,Ronald Walken ,1943-03-31

5.8 ábra A person-tábla megjelenítése a formázás előtt

Ha szeretnéd, hogy megjelenjen a mezők neve az oszlopok fejléceként, akkor minden egyes megjelenítéshez írhatsz egyedi kódot, de én azt tanácsolom, hogy amit csak lehetséges, újrafelhasználható függvényekbe **írj** meg; a következő példa a táblázatok megjelenítéséhez nyújt segítséget. A mysql_result_fields()-függvény argumentumában az eredmény azonosítója szerepel, és minden információt visszaad, amit a MySQL biztosít az eredmény mezőiről. A **mysql_field_name()-nek** és a hasonló utasításoknak az eredmény azonosítóját és a mező indexszámát **kell** megadni, a **mysql_num_field()** a mezők számát adja vissza, a for()-ciklus pedig végigmegy a mezőkön és megszerzi az információkat. A mezők tulajdonságai az output tömbben vannak eltárolva. Abban az esetben, ha nem lennének mezők, egy sor hamisra állítja az output értékét, ha a tömböt nem hoztad létre. A kód egy sor indikátort kap egy sztringben a **mysql_field_flags()** -függvénytől, szétbontja a sztringet az explode()-paranccsal, végül az indikátorokat az indikátor nevével megegyező mezőként a kimeneti tömbbe helyezi:

```
function    mysql_result_fields($result)
    {
    $num_fields = mysql_num_fields($result);
    for($i = 0; $i < $num__fields; $i + + ) {
        $flags = mysql_field_flags(Sresult, $i);
        $flag_array = explodef" ", $flags); while (üst
        ($k, $v) = each <$f lag_array) ) {
            $array[$i][$v] = true; }
        $array[$i]["length"] = mysql_field len($result, $i);
        $array[$i]["name"] = mysql_field_name($result,
        $array[$i] ["type"] = mysql_field_type($result,
```

```
if(!isset($array))    (?array    =
return($array);       falsé;}
```

Most, hogy a mezők információi egy tömbben vannak, a ciklus elő'tt megadhatod a tömb értékét, a tömb segítségével fejlécet illeszthetsz be, és más funkciókat, pl. formázást is vezérelhetsz vele. A következő" kód az előző táblázatrajzoló kód **while()** parancsával kezdődik, és egy 9 soros részt szúr be, ami a **mysql_result_field()**-függvényt használja a fejléc elkészítéséhez. Az eredmény az 5.9 ábrán látható. A $field_array-tömb értékeit a mysql_result_fields()-függvény szolgáltatja, vagyis nem kell ugyanazt a kódot minden sorra megismételni; a kód az if(!isset()) feltételbe van ágyazva, hogy csak akkor fusson le, ha a $field_array változó még nem létezik. A fejléc beillesztésére ez az if () a legjobb hely, mert csak az első sorra lehet végrehajtani, de nem megy végbe, ha egyáltalán nincsenek sorok az eredményben:

\pn t^tame irealname •rn \comment
~\bö,

5.9 ábra A person-tábla fejléce a megjelenítéshez

```
while($row = mysql fetch row($result))

    if(!isset($field array))

        $field array = mysql result fields($result);
        print("<tr>");

        while (üst ($k,    $v)    =    each($field   array))

            print ("<tdxem>"    .    $v["name"]    .

        "</em></td>"); print("</tr>");
```

Egy másik hasznos dolog, amit megtehetünk ezen a ponton, hogy egy könnyen módosítható és újrahasznosítható kóddal az updated-del és a realname-mel kezdődően töröljük a nemkívánatos mezőket. Először készíts egy felsorolást a nemkívánatos mezőkről, mint alább látható:

```
$exclude = array("updated", "realname");
```

Most írd meg a my**sql_table_f**ields()-függvényt, ami egy select-parancshoz használható tömbbe tárolja a mezők nevét, de elhagyja azokat a mezőket, amelyek a harmadik paraméterként átadott tömbben szerepelnek. A függvény a mysql_list_fields()-utasítással kapja meg egy tábla mezőit, igen hasonlóan ahhoz, mint mikor a **select** "'-lekérdezést futtatjuk; ezt követően a kód a mysql_field_name()-utasítással végigmegy az eredményen, és létrehozza a nevekből álló output-tömböt. Az **in_array()**-utasítással vizsgáljuk, hogy az argumentumában elsőnek megadott név szerepel-e az utasítás argumentumában másodikként megadott $exclude-tömbben. Az in_array() igazat ad eredményül, ha az első paraméter

szerepel a másodikban, vagyis a **$exclude** elemei nem szerepelnek az output-tömbben:

```
function mysql_table_fields($database, $table, $exclude="")

    Sresult = mysql_list_fields($database,
    $table); $result fields = mysql num
    fields($result); for($i = 0; $i < $result
    fields; $i++)

        $name = mysql_field_name($result,
        if ( !is array($exclude) or !in^array($name, $exclude)

            $array[] = $name;


    if(!isset($array) ) {$array
    = return($array);
```

Most már képes vagy átalakítani a tábla-megjelenítő kódot az 5.10-es ábrán bemutatott elegáns formátumra, fejlécekkel és elhagyott mezőkkel. Az implodeQ-utasítás arra használható, hogy a **mysql_table_fields**()-függvény eredményeképp kapott tömb elemeit egy sztringgé fűzzük össze az SQL **select** számára. A ", "-paraméter miatt a tömb elemei közé egy vesszőt és egy szóközt illeszt. Még egy változás van: az ábrákat eddig Netscape Navigator 4.76-tal készítettük, mert ez a böngésző hozza előtérbe a legtöbb problémát, azok közül, amelyek az Interneten használatos több tucat böngésző közt elszórva megjelennek - biztosan észrevetted, hogy az üres celláknak nincs szegélye. Ahhoz, hogy elkerüld ezt és számos más problémát a böngészőkkel, hozzáadtam egy sort a kódhoz, ami megvizsgálja az outputot, és ha valamelyik mezője egy nulla hosszúságú sztnng, akkor ezt **lecseréli** egy **HTML** nem törhető szóközre, azaz -re. A változások az előzőhöz képest ki vannak emelve:

```
$database["database"] = "movies";
$database["table"] = "person";
$exclude = array("updated", "realname");
$select = mysql table fields($database["database"    $database["table"],
   Sexclude); $database["sql"] = "select " . implode(",
", $select)
    . " from " . $database["table"]; mysql
select db($database["database"]); $result =
mysql_query($database["sql" ] ) ;
print("<table border=\"l\">"); while($row =
mysql_fetch_row ($result))
    f
    if ( lisset($field array))

       $field array = mysql result fields($result);
       print ("<tr>") ;
       while (üst ($k, $v) = each($field array))

       print ("<tdxem>" . $v["name" }      "</emx/td>'
       print("</tr>");
```

```
    $fields = mysql num fields($result);
    print ("<tr>") ;
    for($i = 0; $i < $fields; $i++)
        {
        if(!strlen($row[$i])) {$row [$i] = " ";}
        print("<td>" . $row[$i] . "</td>");
        }
    print("</tr>");
    }
print("</table>");
```

\entry 'fiamé	}fiom	fcomment
1 Tom Berenger	1949-05-31	
[2 jMimi Rogers	1956-01-27	
(3 |**Lorraine**	1955-10-22	
Bracco |4 iJerry	1935-10-20	
Orbach [5 [John	1946-12-08	
6 iRidleyScott		
7 JRussell Crowe	**1964-04-07** JThe world's greatest **actor.;**	
11 |**Christopher**	**1943-03-3l[**	

5.10 ábra A person-tábla a kihagyott oszlopokkal

Adatsor beillesztése

Ha adatokat akarsz bevinni, egy adatforrásra van szükséged, ami rendszerint egy űrlap. Hogy takarékoskodjak a papírral, röviden vázolok egy űrlapot a 9. fejezetben kifejlesztett függvények segítségével, majd úgy használom az adatokat, ahogyan megjelennek a szkriptben, amely adatokat fogad az űrlapból.

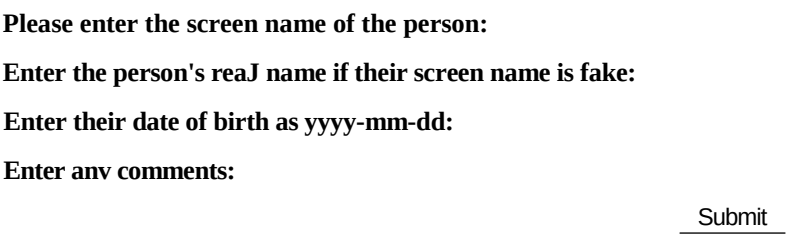
Hivatkozás:	oldal:
Űrlapok létrehozása függvényekkel	**307**

Tegyük fel, hogy a people.html-oldalon színészek, rendezők és producerek nevét lehet megadni. A 9. fejezet függvényeivel így hozható létre az űrlap:

```
$question[]    =array ("name"  =>  "person",  "type"  =>  "text",
    "question" =>  "Please  enter  the  screen  name  of  the
    person:");
$question[]    =array("name"  =>  "reál",  "type"  =>  "text",
    "question" =>  "Enter  the  person's  reál  name  if  their"
    .  "  screen  name  is  fake:");
$question[]    =  array("name"  =>  "born",  "type"  =>  "text",
    "question" =>  "Enter  their  date  of  birth  as  yyyy-mm-dd:")
    ;
$question[]    =  array("name"  =>  "comment",  "type"  =>  "text",
```

```
    "questíon" => "Enter any
comments:") ; form("people.html",
$question);
```

Az 5.11 ábrán látható, hogy néz ki az űrlap a képernyőn.



5.11 ábra Űrlap a színészek nevének beviteléhez

Mikor az űrlap kitöltője megnyomja a Elküld-gombot, az oldal megkapja a színész adatait. Ehhez a gyakorlathoz az alábbi adatokat fogjuk használni:

```
$person = "Russell Crowe";
$real = "Russell Ira Crowe";
$born - "1964-04-07";
$comment = "The world's greatest actor.";
```

A person-táblában két egyéb mező van, és a MySQL mindkettőt automatikusan frissíti. Az elsődleges kulcs mező egész, **auto_increment-tel,** azaz a MySQL a következő elérhető számot rendeli hozzá. Az **updated** nevű mező típusa **timestamp,** azaz a MySQL minden alkalommal frissíti ezt a mezőt, amikor a rekord frissül, vagyis ez alapján megállapítható, mikor frissült utoljára. Sajnos a MySQL nem állít be használható értéket a timestamp-mezőnek a rekord első beillesztésekor, amit kijavíthatsz, ha a beszúró SQL-utasításhoz hozzáadod az updated-mezőt **null** értékkel. Az alábbi SQL-kód illeszti be az új sort:

```
$sql = "insert intő person set updated=null"
    . ", name='" . Sperson . "'"
. ",realname='" . $real .
. ",born='" . $born . "'"
. ",comment='" . addslashes($comment) . "'";
```

Felhívom a figyelmed az addslashes()-utasításra, amely visszaper jelet (\) ad a sztringhez. A visszaper jel átváltási kód, vagyis megakadályozza, hogy az utána következő karakter megszakítsa az SQL-utasítást. Egy SQL-sztringben a mezőket sima idézőjelek határolják, tehát a sima idézőjeleket kell, hogy megelőzze egy visszaper jel. A visszaper jel nincs eltárolva az adatbázisban, így nem kell eltávolítani, mikor olvasod az adatokat.

És most lássuk a kódot, ami beír egy új sort. Ez a legegyszerűbb kód:

```
mysql_select_db("movies");
$result = mysql
query($sql);
```

Ez az egyszerű kód durva hibaüzeneteket eredményezhet, ezért illessz egy @-ot a mysql_query() elé, és ágyazd be az utasítást egy hibaellenőrző kódba, mint a következőkben látható. Aztán dolgozz ki valami értelmes hibaüzenetet, és cseréld le az én rejtélyes üzenetemet:

```
if($result = @mysql

    query($sql)) print("Success!"};

else

    print("Error: " . mysql_errno()  . ", " . mysql_error() .
        " in sql: " . $sql);
    };
```

A kód működik és ezt jeleníti meg:

```
Success!
```

Hogy láss egy hibát, a name-et lecseréltem nname-re, és újra lefutattam a kódot. A mysql_errno() egy hibaszámot ad eredményül, a mysql_error() pedig egy hibaüzenetet. Kedved szerint használhatod ezeket az utasításokat, mivel most már a kezedben van a hibaüzenetek megjelenítéséhez szükséges eljárás. A szándékos hibának ez lett az eredménye:

```
Error: 1054, Unknown column 'nname' in ' field list[1] in sql:
insert intő person set updated=null, nname='Russell Crowe[1],
realname='Russell Ira Crowe', born='1964-04-07', comment='The
world\'s greatest actor.'
```

Figyeld meg, hogy a komment értéke sima idézőjelet tartalmaz, amit visszaper jellel láttunk el.

Még egy dolgot tehetsz, mikor új rekordot írsz be az adatbázisba: visszajelzést kérhetsz a beillesztés azonosítójáról, ha a táblában automatikusan növelt érték van. Képzeld el, hogy létrehozol egy oldalt, ahol be lehet vinni egy filmcsillagot és leghíresebb filmjeit. Az oldal a csillag nevét a person-táblába írná be, majd a kapott azonosítót beillesztené az egyes filmekhez tartozó szereposztáslistába, vagyis összekapcsolhatod a filmeket és a színészeket. A person-táblában van automatikusan növelt mező, vagyis a mysql_insert_id()-utasítással megkaphatod az azonosítót:

```
$personentry = mysql insert_id();
```

Az egyedüli paraméter, amit a mysql_insert_id()-nak megadhatsz, a kapcsolat, ha több aktív kapcsolatot használsz. A visszatérési érték az auto_increment-mező legaktuálisabb értéke az éppen futó szálban, ezért az utasítást közvetlenül a sikeres insert után kell helyezni. Vagyis valahogy így:

```
if($result = mysql_db_query("movies", $sql))                    >*

    $personentry = mysql insert id() ;
```

Adatbázis létrehozása

Ha MySQL-ben egy új adatbázist akarsz létrehozni, használhatod a **PHP mysql_creat_db**()-utasítását, amit a következő példa mutat. A példa feltételezi, hogy már kapcsolódtál a MySQL-hez, amit a fejezet első Azonnali megoldásában bemutatott kóddal tehetsz meg. Hogy hibát provokáljon (hogy hibaüzenetet láthass), a kód egy **movies** nevű adatbázist próbál létrehozni, amely már létezik a tesztgépemen. A **mysql_create_db()** visszatérési értéke igaz, ha sikerrel jár, és hamis ha kudarcot vall. A **mysql_error()** a MySQL által legutoljára jelentett hibát adja vissza. A kód hiba esetén a hibaüzenet előtt egy saját megerősítő üzenetet is megjelenít, tehát nem lehet félreérteni, sikerült-e a művelet vagy sem:

```
$database["database"] = "movies";
if(mysql_create_db($database["database"]))

     print ("<p>Created database " . $database["database" ;

  else

     print("<p>Failed to create database " .
     $database["database"] . ", error: " . mysqlerror() .
     "</p>");
```

Hogy lásd, milyen a **mysql_error()** eredménye, a következő példa a hibaüzenetet mutatja, amit azért kaptam, mert már létezett az adatbázis:

```
Failed to create database movies, error: Can't create database 'movies'.
Database exists
```

Egyéb hibalehetőségek: nem sikerül kapcsolódni a szerverhez, vagy új szerverhez kapcsolódsz és nincs megfelelően konfigurálva; engedélyezési hiba, ha nincs jogosultságod adatbázis létrehozására; és az a kevésbé nyilvánvaló hiba, ami abból fakad, hogy olyan névvel próbáltál adatbázist létrehozni, amit nem fogadott el az operációs rendszer. Az érvénytelen adatbázisnév azért fordul elő, mert a MySQL megpróbál az adatbázis számára egy azonos nevű új könyvtárat létrehozni, amit az operációs rendszer visszautasít.

Mikor adatbázisokat és táblákat hozol létre SQL-en keresztül, könnyen megfeledkezhetsz a szerverről és az operációs rendszerről, amit használsz. Könnyű elkövetni egy apró hibát (mint pl. egy szóköz a tábla nevében), és órákat tölthetsz az adatbázis félrevezető hibaüzeneteinek megfejtésével. Egy dologról ne feledkezz el: az adatbázisból könyvtár, a táblából pedig fájl lesz, és mindkettőre érvényesek az operációs rendszer megszorításai.

Táblák létrehozása

A táblák létrehozása és egyéb SQL-művelet futtatása közt nincs sok programozási különbség. A MySQL-ben használd a **mysql_query()-t,** a PostgreSQL-ben a pg_exec()-t. Mindkét adatbázisban vannak különleges funkciók, ám a PostgreSQL-ben vannak táblaszerű konst-

rukciók is, mint a nézetek; bizonyos esetekben a nézet életképes alternatívája lehet az új tábla létrehozásának.

A következő példában a **mysql_select_db()** -utasítás az első, „Kapcsolódás az adatbázishoz" c. Gyors megoldások megnyitott kapcsolatot használja, és kiválasztja az adatbázist. A tábla létrehozását szolgáló SQL-utasítás a **mysql_query()**-parancs argumentumában szerepel, melynek visszatérési értéke igaz, ha az SQL-utasítás működik és hamis, ha nem. A kód többi része, a hibaüzenet rész és a **mysql_error()**-utasítás megegyezik az *előző,* „Adatbázis létrehozása" c. Gyors megoldásokban bemutatott kóddal:

```
$database["database"] = "movies";
$database["table"] = "director";
$database["sql"] = "create table " . $database["table"]
    . " (movieentry int (10) unsigned default '0' not null,"
    . " directorentry int (10) unsigned default '0' not null,"
    . " updated timestamp(14) , "
    . " personentry int (10) unsigned default '0' not null,"
    . " comment text not null,"
    . " primary key (movieentry, directorentry));";
mysql_select_db($database["database"]) ; if(mysql
query($database["sql"]))

    print("<p>Created table " . $database["table" ] . "</p>");

else {
    print("<p>Failed to create table .
        ", error: " . mysql_error()
                                        $database["table"]
```

Most már képes vagy táblákat létrehozni egy szkriptből, és ugyanez a kód más SQL-utasí-tásokkal is használható, mint pl. **drop table director-táblák** törlése, és képes vagy megtölteni a honlapodat különböző trükkös megoldásokkal, mint pl. dinamikusan létrehozott tábla egy bolt minden termékkategóriájához. Átmeneti szálláshelyként is létrehozhatsz táblákat köztes adatok számára, pl. mikor a lecseréled fájlrendszert vagy az adatbázist.

Adatbázisok használata session-ökhöz

A PHP session-kezelése jól működik fájlokkal is, de nagyszámú session esetén az adatbá-zis-bejegyzések hatékonyabbak. El kell végezned néhány dolgot, mielőtt session-öket használhatnál, és van még néhány további lépés, ha adatbázist akarsz használni a session-ökhöz. Ezeket a lépéseket abban a sorrendben követjük végig, amelyben a webszerver és a PHP telepítésekor elvégeznéd őket.

Mikor a PHP-t telepíted, a PHP fordításakor meg kell adni az --enable-trans-id-paramé-tert annak érdekében, hogy a PHP session-azonosítókat adhasson az URL-ekhez, ha a böngésző nem fogad cookie-kat. A **PHP** Windows-os, Windows 2000-es és NT-s előre lefordított verzióiban már eleve szerepel ez az opció.

A php.ini-ben be kell kapcsolnod két opciót: a **track_vars-t,** ami a 4.0.4-es verzió felett automatikusan bekapcsolódik és a **register_globals-t,** ami akkor is hasznos, ha nem használsz session-öket. Ha a PHP-t CGI-ként futtatod, akkor a PHP minden szkript futásakor beolvassa a php-ini-t, tehát a php.ini-ben eszközölt változtatások azonnal érvényesülnek, míg ha Apache-modulként lett telepítve, akkor le kell állítanod az Apache-ot, elvégezni a változatásokat a php.ini-ben, majd újraindítani az Apache-ot, hogy a PHP beolvassa a frissített php.ini-t.

Automatikusan indíthatsz session-öket, ha a php.ini-ben beállítod, hogy **session.auto_start** = 1, vagy manuálisan is indíthatsz szkriptből, a **session_register()-** vagy a session_start()-parancsokkal. Akkor alkalmaznád az automatikus session-kezdeményezést, ha a honlapod összes látogatóját követni szeretnéd, és akkor kezdeményeznéd manuálisan, ha csak egy bizonyos személy bejelentkezése esetén akarsz session-t használni. Ha a session-ök a bejelentkezéshez kapcsolódnak, akkor kijelentkezéskor a session_destroy()-utasítással szüntetheted meg őket.

Mikor egy session megkezdődik, egy új session-azonosítót kap, amit mindaddig megtart, amíg meg nem semmisül a **session_destroy()** által, vagy túl nem lépi az időkorlátját. A session-azonosító oldalról oldalra vándorol, vagy cookie-k, vagy URL-ek által. A PHP a szkript végén a merev lemezre menti a session-információkat, majd pedig újra betölti őket a memóriába, ha egy újabb szkript ugyanahhoz a session-höz kapcsolódik. Mivel a session-azonosítója egyedi a honlapodon, kulcsként használhatod azt a session-adatbázisban. A session-adatokban lévő információkat pedig alkalmazhatod tetszőleges adatbázisokra való hivatkozásra, mint például fogyasztói profil vagy az aktuális rendelés teljesítése.

Eldöntheted, hogy a php.ini **session.use_cookies** = **1** beállítása révén automatikusan használod a cookie-kat, vagy inkább manuálisan. A manuális használat lehetővé teszi, hogy leellenőrizd a cookie-k működését még mielőtt látogatóid elégedetlenkedni kezdenének. Ha már valami máshoz használsz cookie-kat, akkor merül) el az automatikus cookie-k tanulmányozásában, különös tekintettel a cookie-k használatának a session-kezeléssel történő összekötésére, hogy áthelyezhesd az adatokat a cookie-kból a session-rekordokba. A cookie-k neve **session.name** = **PHPSESSID.** Megváltoztathatod a nevet és az élettartamot, a **session.cookie_lifetime-ot,** melynek az alapértelmezése 0, ami azt jelenti, hogy a session **a** böngésző bezárásáig tart. Miközben módosítod a php.ini-t, győződj meg róla, hogy a **session.save_handler** = user-beállítás van érvényben, hogy biztosítani tudd a session-kezelő rutinokat.

Hozz létre egy sessions nevű adatbázist és egy **session** nevű táblát; használhatod a korábbi Azonnali megoldásokat vagy előre megírt adatbáziskezelő alkalmazást. MySQL esetén időt spórolhatsz meg ezzel az SQL-utasítással:

```
create table session (
   id varchar(32) binary not null,
   updated timestamp(14), data text
   not null, primary key (id)
```

A további példák MySQL-en alapulnak, de a korábbi Gyors megoldások kódjai alapján le-fordíthatóak PostgreSQL-hez vagy más, a 6. fejezetben említett adatbázishoz.

A session-változó elmentéséhez egyszerűen regisztráld a változót a session_register()-uta-sítással, és a szkript futásának végén a PHP elmenti a session-rekordba, majd a szkript kezdetén újra betölti. Ha szeretnéd lehetővé tenni a látogatóknak, hogy kiválasszanak egy kategóriát és szeretnéd ezt session-ök révén megőrizni, akkor nevezd át a változót $category-ra és illeszd be az alábbi kódot a szkriptbe:

```
session_register("category");
```

A $category-vákozót bármikor létrehozhatod, de ne feledd, hogy a szkript kezdetén törlődik, és minden formok által létrehozott változót felülír. Ha egy formon keresztül szeretnéd egy új kategória kiválasztását biztosítani a látogatónak, akkor valami más nevet kell adnod a mezőnek, pl. Snewcategory, majd kiegészítened a szkriptet, hogy a $newcategory átadja a kiválasztott kategóriát a Scategory-változónak, mint a következő példa mutatja:

```
if(isset($newcategory) and strlen ($newcategory))
    { $category = Snewcategory;
```

Ha már nincs szükséged egy változóra, akkor a session_unregister("category")-utasítással engedheted el, pl. ezt tennéd egy fogyasztói profil változóval, ha az illető kijelentkezik. Úgy is dönthetsz, hogy nem regisztrálod, hanem a session-rekord erre rendelt mezőibe írod a változókat. Akkor tennéd azt a $category-val, ha azt szeretnéd, hogy egy online oldal kategóriánként felsorolja az aktuális látogatók számát. Amikor egy változó el van temetve a PHP session-rekordjában, akkor nem használhatod úgy az SQL-t az adatok kiválasztására, rendezésére vagy összegzésére, mint ahogy megteheted, ha az adatok egy külön adatbázismezőben vannak eltárolva.

A session-rekordok készítéséhez, az adatbázisban való eltárolásához, visszanyeréséhez és az elavult rekordok törléséhez számos utasításra van szükséged. A session-ök feldolgozása a feldolgozó utasítások definiálásával kezdődik. Ezt követi az utasítások regisztrációja, majd a session indítása a session_start()-utasítással vagy a session_register()-parancs első használatával. Ha a php.ini-ben be van kapcsolva az automatikus feldolgozás, akkor a session feldolgozása a session_set_save_handler()-utasítással kezdődik, mint a következő kódban látható. A session_set_save_handler() az utasítások definiálását szolgálja, és megelőzi a session_start()-ot vagy az első session_register()-t, valamint a változók első használatát, amelyeket különböző oldalakon regisztráltál egy session-ben:

```
session_set_save_handler("mysql_session_open",
    "mysql_session close",
    "mysql__session_read", "mysql_session_write",
    "mysql_session_destroy", "mysql_session
    gc");
```

Mikor a session kezdetét veszi, az alábbihoz hasonló hibaüzeneteket kaphatsz. Ez azt jelzi, hogy a session túl későn próbálta meg elküldeni a cookie-kat, vagy a session-kód meggátolta a szkript hátralévő részét a fejléc elküldésében:

```
Warning: Cannot send session cookie - headers already sent
by_ Warning: Cannot send session cache limiter - headers
already sent Fatál error: Failed to initialize session modulé
```

A cookie-k HTTP-fejlécek és a HTTP-fejléceket elsőként kell elküldeni, minden más output előtt, beleértve az eltévedt karaktereket, amelyeket esetleg észre sem veszel; ezért győződj meg arról, hogy minden oldalad PHP-módban indul **(<?php),** és mindaddig abban marad, amíg a session elkezdődik. Ha a session-kódod figyelmeztető üzeneteket eredményez, akkor ezek az üzenetek meggátolják a szkript többi részét a fejlécek elküldésében, ezért gondoskodj arról, hogy a fejlécek elküldése az oldal tetejére kerüljön, közvetlenül a session kezdete után. Ha a szkript olyan fejléceket küld, amelyek megakadályozzák az oldal gyorsítótárba való betöltését, akkor ezeket elhagyhatjuk, mivel a PHP session-kód szintén küld ilyen fejléceket.

Mielőtt a **session_set_save_handler()-t** használnád, definiálnod kell az általa meghívott függvényeket. Ha ezt elmulasztod vagy elgépelsz valamit, akkor a következő hibaüzenetet kapod. A PHP4 az egész kód szintakszisát leellenőrzi futtatás előtt, vagyis a függvényeidnek hibátlannak kell lenniük, ha egy ilyen hibaüzenetet kapsz:

```
Fatál error: Failed to initialize session modulé
```

A PHP-session-ök kezeléséhez szükséged van egy utasításra, ami megnyit egy session-t. Az alábbi MySQL példa egyszerűen megnyit egy állandó kapcsolatot a MySQL-szerverrel. A session-öket olyan adatbázison is tárolhatod, ami nem a webszervereden van, és állandó session-t is megnyithatsz, mikor több szervered van, pl. több Apache-webszerver osztozik a session-ökhöz rendelt adatbázison, és az összes többi adatbázis-tevékenység a többi adatbázis közt oszlik meg. A **$session** változó egy session-ökkel kapcsolatos információk felsorolását tartalmazó tömb, amely a függvények számára egyetlen globális definícióval elérhető. Ez azt jelenti, hogy a kód rendezett vagy strukturált módszerrel egyaránt hozzáfér a **$session** részeihez. A PHP biztosít egy utasítást, a **mysql_session_open()-t,** amelynek a fájlelérési utat és egy session-nevet kell megadni, ugyanis általában ez szükséges a fájl alapú session-ökhöz. Megadhatod az adatbázis és a tábla nevét, de tisztább, ha mindent a standard **Ssession** mezőkben tartasz:

```
$session["database" ] = "sessions";
Ssession["table"] = "session";
$session["host"] = "localhost";
$session["log"] = "t:\mysqlsessionlog.txt";
$session["user"] = "";
$session["password"] = "";
function mysql session open($path, $name)
    {
    global $session;
    mysql_session_log("mysql_session_open");
    if<$session["connection"] = @mysql pconnect($session["server"],
        $session["user"]/ $session["password"]))
        {
        if(!isset($session["fields"]))
            {
            $result = mysql üst fields ($session ["database"] ,
```

```
            Ssession["table"], $session["connection"]);
        $result_fields = mysql_num_fields($result);
        for($i = 0; $i < $result_fields; $i++)
            {
            $session["fields"][mysql_field   name($result,    $i)]
               = mysql_field_type($result,


    return(true);
    } mysql_session_log("Connect failed to server: " .
$database["server"]
    . ", MySQL error: " . mysql_error());
return(falsé);
```

Ha megvizsgálod a **mysql_session_open()**-függvény kódját, észreveheted, hogy a korábbi gyors megoldásokból származik: pontosabban abból a kódból, amely a kapcsolat létrehozására szolgált az első megoldásban, plusz a mezők neveinek kinyerésére szolgáló kódból egy későbbi Gyors megoldásból. A kódot a mezőnevek és mezőtípusok megállapítására írtam, vagyis mindkettőt használhatod a session-rekordok adatainak kezelésére szolgáló kiíró és olvasó függvényekben, amikor elkezdesz különálló mezőket hozzáadni a session táblához.

Az előző kódban használt és a következőben bemutatott mysql_session_log()-függvény azért készült, mert a session write- (kiíró) és session close- (lezáró) utasításokkal nem jeleníthetsz meg diagnosztikai üzeneteket a képernyőn, mivel mindkét utasítás akkor hajtódik végre, mikor az oldal az utolsó adatot is elküldte a böngészőnek. A **mysql_session_log()** egy külön fájlba írja a naplóbejegyzéseket, amit bármikor böngészhetsz, és mivel ezek nincsenek eltemetve az Apache vagy a PHP naplóállományaiban, nem kell az Internet-szolgáltatóval küzdened, hogy hozzáférést kapj a fájlokhoz (ami néha előfordul, ha nem a saját szervereden van a honlapod). A napló tartalmaz egy dátumot és egy időt, hogy tudd, melyik teszt eredményezte az adott bejegyzést, és minden üzenet végére odatesz egy újsor **(newline)** karaktert, hogy el tudd különíteni a bejegyzéseket, mikor egy szerkesztőprogramban böngészed a szövegfájlt:

```
function    mysql_session_log($message)
    {
    if($file   =   fopen($session["log"],    "a"))
        {
        fwrite($file, date{"Y-m-d H:i:s  ") . $message . "\n");
        fclose ($file);
```

Amit megnyitsz, azt egyszer le is kell zárnod, és a **mysql_session_close()** pont ezt teszi a session-nel. Azonban ha adatbázist használsz, nincs szükség erre a függvényre. A MySQL-kapcsolatot nem kell lezárni, valójában szándékosan állandó kapcsolatként volt megnyitva, hogy csökkentsük a megnyitásokat és lezárásokat. Ez a függvény egyetlen naplóüzenetet küld el, amely a tesztelés segítségére szolgál és aztán kitörölhető (ha a teljesítményt szeretnéd tesztelni a honlapodon, akkor a napló idejét lecserélheted a PHP microtime()-jára$_3$ és a

mysql_session_close() naplóbejegyzésének idejét tekintheted a szkript befejezésének időpontjaként):

```
function mysql_session_close() {
    mysql_session_log("mysql_session_close");
    return (true); }
```

A következő kód a **mysql_session_read()**-függvény, mely a **mysql_query()**-utasítást használja a „Táblák adatainak megjelenítése" címen a Gyors megoldásokban ismertetett módon, és mindössze néhány új dolgot csinál. A **mysql_fetch_array()** második paramétereként megadott **MYSQL_ASSOC** révén az eredményként visszaadott tömb elemeit csak kulcsszavakkal lehet elérni, indexszámokkal nem (az alapértelmezés szerint mindkettő használható). Ez a trükk lehetővé teszi, hogy egy ciklus úgy menjen végig egy soron, mint egy tömbön, melynek elemeihez tartozó kulcsszavak a mezők nevei. A visszatérési érték a **data** nevű mező tartalma lesz, míg a többi mező a mezőnév szerinti indexszel a **$session** változóba kerül. A plusz kód azt jelenti, hogy tetszőleges számú mezőt hozzáadhatsz a session-táblához, és automatikusan visszanyerheted őket anélkül, hogy meg kellene változtatnod a **mysql_session_read()-et:**

```
function mysql_session_read($id)
    {
    global $session;
    mysql_session_log("mysql_session_read") ; $session["sql"]
    = "select * from " . $session["table"]
        . " where id='" . $id .
    mysql_select_db($session["database" ] ) ;
    if($result = @mysql_query($session["sql"] , $session["connection"])) (
       if($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
           while (üst ($k, $v) = each($row))
               {
               if($k != "data")
                   {
               $session[$k] = $row[$k]; } }
           return($row["data"] ) ; }
       else { return(""); // This point means there is not yet a record.


    else {
       mysql_session_log("Session read query error: "
           . " using sql " . $session["sql"]};          mysql error ()
       return ("") ;
```

A session-t kiíró függvény, a **mysql_session_write()** a session-azonosítót és a session-ada-tot fogadja bemenő paraméterként, és megpróbál frissíteni egy létező rekordot. Ha a frissítés nem sikerül, a **mysql_session_write()** feltételezi, hogy a session most próbált először az adatbázisba írni, és beilleszt egy új rekordot. Az SQL kezelése ugyanaz, mint korábban. Az **addslashes()** révén megbízhatóan beillesztheted a szöveges adatot az SQL-en keresztül. A kód az SQL-utasítás dinamikus generálásához a **mysql_session_open()** által létrehozott mező'felsorolást (Sfield_list) használja. Mielőtt a mezők bekerülnek az SQL-be, néhányat kitörlünk a tömbből: a updated-et, mert ezt az adatbáziskezelő szoftver frissíti, a **data-**mezőt, mert ezt külön illesztjük be az SQL-be, és az azonosítót, az id-t, mert frissítéskor nem kell azonosítót beilleszteni:

```
function mysql session write($id,
    $data) {
    global Ssession;
    mysql_session_log("mysql_session_write");
    $session["id"] = Sid;
    $fields_sql = " set data='" . addslashes($data)
    . "'"; $fields_list = $session["fields"];
    unset($fields_list["data"] ) ; unset($fields_list["id"]) ;
    unset ( $ f ields_list["updated"] ) ; while (üst ( $k, $v) =
    each ( $ f ields_list) )

       if(isset(Ssession[$k]))

          Sfields sql .= ", " . Sk . "='      addslashes($session[$k] )



   Ssession["sql"] = "update " . $session["table"]
       . $fields_sql . " where id='" . $id . "'";
   mysql_select_db(Ssession["database"]);
   if ($session["result"] = @mysql_query(Ssession["sql" ] , _
       $session["connection"] ) ) {
       if(mysql_affected_rows($session["connection" ] ) ) {
          mysql_session_log("mysql_session_write update using sql: "
              . $session["sql"]);
          return (true);

       else

          // Update failed so insert new record:
          Ssession["sql"] = "insert intő " . Ssession["table"]
              . $fields_sql . ", id='" . $id . "'"; if
          (ísession["result"] = @mysql db
          query(Ssession["database"],
              Ssession["sql"], Ssession["connection"]))
              {
             mysql_session_log("mysql_session_write insert using sql: " .
                 Ssession["sql"]);
```

```
            return (true);
            }
        else
            {
            mysql_session_log("mysql_session_write insert error: " ,
                mysql_error()
                , " using sql " . Ssession["sql"]);
            return(falsé); 1


else
    {
    mysql_session_log("mysql_session_write update error: "
        . mysql^error()
        . " using sql " . $session["sql"]);
    return(falsé);

}
```

A mysql_session_destroy()-függvény a mysq_query()-t használja a megadott session-azo-nosítóval rendelkező session-rekord törlésére. Ha ez a kód véletlenül meghagy egy rekordot, akkor a következőkben bemutatott szeméteítakarító kód fogja kitörölni:

```
function    mysql_session_destroy($id)
    {
    global Ssession;
    mysql_session_log("mysql_session_destroy") ;
    $session["id"] = $id;
    Ssession["sql"] = "delete from " . Ssession["table" ]
        . " where id = '" . $id . "'";
    mysql_select_db(Ssession["database"]);
    if(Ssession["result"] = @mysql_query(Ssession["sql"] , _
        Ssession["connection"]))
        {
        return(true);

    else

        mysql_session__log ("Session destroy error:      mysql   error
            . " using sql " . Ssession["sql"]);          ()
        return(falsé);
```

A PHP session-kezelése magában foglalja egy session után fennmaradó szemét eltakarítását is. A szemét session rekordokat jelent, amelyeket olyan látogatók hagytak, akik nem jelentkeztek ki az oldalról, vagy olyanok, akik 3 hétre elutaztak, és a bekapcsolva hagyott böngésző az állandó Internet-kapcsolaton keresztül ugyanazt az oldalt böngészi. Minden rekordnak van egy updated nevű mezője, amit az adatbáziskezelő frissít, vagyis kitörölheted azokat a sorokat, amelyekben az updated-mező túl öreg. Az öregség definícióját úgy kapod, hogy a mező értékét az aktuális idő és a php.ini-ben beállított session **lifetime** (élet-

tartam) különbségével veted össze. A time()- és date()-függvények, valamint a lifetime mind másodperc alapúak, tehát a time() eredménye egyszerűen kivonható a lifetime-ból, amiből a date()-utasítással kapsz MySQL formátumú dátum/időt. A MySQL függvényeit is használhatnád az idő meghatározására és számítására, de azáltal, hogy PHP-ban végzed a számításokat, lehetőséged nyílik arra, hogy lásd az eredményt. Az egyedüli hibalehetőség, ha a szerver helyi időhöz van állítva, míg a MySQL a GMT-hez vagy fordítva. A PHP-ban vannak utasítások, amelyek a helyi időt adják meg, és vannak, amelyek a GMT-t, vagyis bármilyen szerverhez vagy beállításhoz alkalmazkodhatsz:

```
function mysql_session__gc ($lifetime)
    {
    global Ssession;
    mysql_session_log ("mysql_session__gc") ;
    $session["gcdate"] = date("YmdHis", time() -
    $lifetime); $session[Msql"] = "delete from " .
    Ssession["table"]
        . " where updated < '" . $session["gcdate"] .
    "'"; mysql_select_db <$session["database"]);
    if(Ssession["result"] = @mysql_query($session["sql"] , _
        ?session["connection"]))
        {
        return (true);
        }
    else

        mysql_session_log("Session gc error:
        " . " using sql " . mysql_error()
        Ssession["sql"]); return (falsé);
```

Mos már birtokában vagy a mágikus session-kódnak, amely MySQL-lel működik de módosíthatod más adatbázisokhoz. Könnyedén hozzáadhatsz egyes mezőket a session-rekord-hoz, vagyis módosíthatod a session-jeidet és elemezheted a rekordokat. Megállapíthatod, kik látogatják a honlapodat, honnan böngésznek és mit keresnek. Ha van keresési lehetőséged, akkor a keresési sztringet kezeld külön mezőként a session-rekordban, ahelyett, hogy a session-adatokból próbálnád előásni. így írhatsz egy lekérdezést, amivel online megtudhatod, melyek a legfrissebb vagy legkeresettebb fogalmak.

A kód megtisztítása

A legegyszerűbb módon úgy foghatsz neki a MySQL vagy a PostgreSQL adminisztrálásához, ha telepítesz egy előre megírt alkalmazást, mint pl. a phpMyAdmint vagy a phpPgAdmmt. Ezek segítséget nyújtanak az első adatbázisok, táblák és mezők létrehozásához, és lehetővé teszik, hogy kísérletezz, példakódokat és példa SQL-utasításokat biztosítanak, és megkönnyítik az adatok más forrásokból való importálását. Ha szeretnél ilyen alkalmazásokat találni, megnézheted az ajánlásaimat a petermoulding.com-on, vagy kereshetsz a fershmeet.net-en vagy a sourceforge.net-en.

Ha már van kódod, szeretnéd megtisztítani a hibáktól; ez a rész ezzel a kérdéssel foglalkozik, és példának a phpPgAdmin 2.3-at használja. A változtatásokat bármire alkalmazhatod, akár más fejlesztők kódját is finomíthatod, így a Ford szintű kódok Mercedes színvonalra emelkednek.

Mikor kódot tisztítasz, gyakran van szükséged a függvények megkeresésére, de a legtöbb kódban keverednek az első példában bemutatott megoldások, vagyis néhol egy szóköz szerepel az utasítás neve és a zárójel között. Ezért egy globális keresést kell futtatnod a kódon a " ("-sztringre, hogy kiszűrd a függvénynevek után előforduló szóközöket és el kell távoli-tanod ezeket, hogy a kód úgy nézzen ki, mint a második sor. Ha már alkalmaztad ezt a javítást a kódra, akkor megbízhatóan rákereshetsz a függvényekre, és alkalmazhatod azokat az általános változtatásokat, amelyeket a következőkben ismertetek:

```
include      ( "footer.inc.php") ;
include( " f ooter.inc.php");
```

Az előre megírt kódok feltételezik, hogy megbízható hozzáféréssel rendelkezel a PHP include- (beágyazás) könyvtárához, de sok Internet-szolgáltató nem biztosítja a szükséges hozzáférést, tehát ahhoz, hogy olyan kódot használhass, mint pl. a phpPgAdmin, a helyi könyvtárból kell beágyaznod a fájlokat. Ha a beágyazott fájlok létrehozása megfelelő volt, nem fognak biztonsági vagy megbízhatósági problémákat okozni a böngészőben, és néhány Internet-szolgáltatónál könnyebb létrehozni egy kisméretű, jelszavas hozzáférésű adminisztratív alkönyvtárt, mint egy saját PHP include-könyvtárat kapni.

Ahhoz, hogy a helyi könyvtárból ágyazd be az include-fájlokat, a következő formájú include- (és require-) parancsokat:

```
include("footer.inc.php");
```

le kell cserélned erre a formára:

```
include("./footer.inc.php");
```

Mivel egy ilyen kézi módosítást minden alkalommal el kell végezni, ha módosítod a website konfigurációs beállításait, menj tovább egy lépéssel, és a következő formátumra cseréld a kódot. Az include_prefix-hez hozzáfűzöd a fájlnevet, így az előtagot csak egy helyen adod meg, és egy mozdulattal módosíthatod az összes előfordulását:

```
include{include_prefix      .      "footer.inc.php");
```

Még tovább fejlesztheted a kódot, és felkészülhetsz arra a napra, mikor a .php utótagot lecseréled valami másra, mondjuk áttérsz a .php5-re (ami már tervbe van véve). Egyszerűen cseréld ki a .php-t az include_suffix-ra, mint itt látható:

```
include(include_prefix      .      "footer.inc"      .      include_suffix);
```

Ha lecseréled az include()-ot, ne felejtsd el lecserélni az include_once(), a requireQ és a require_once() összes előfordulását is. A require()-utasítás megegyezik az includeQ-dal, de a PHP sokkal hamarabb feldolgozza, még a feltételes állítások, pl. if()-ek előtt. Az include()-ot lehet If()-fel vezérelni, de a require()-utasítás mindenképpen beilleszt egy fájlt, függetlenül attól, hogy milyen kódba van beágyazva.

Az includeprefix-et és include_suffix-et valahol meg kell adni, ezért hozz létre egy kis include.html nevű fájlt (vagy include.php-t, ha a webszerver nem a PHP-vel dolgozza fel a .html-fájlokat), majd add meg az értékeket az include.html-ben, ahogy a következő kód mutatja:

```
<?php
define("include_prefix", " . /") ;
define("include_suffix", ".php");
```

Bármilyen helyi konfigurációs információt megadhatsz az ínclude.html-ben. Az összes alkalmazásodban használhatod a fájlt, és azt tanácsolom, ezt a megközelítést használd, ha azzal szembesülsz, hogy le kell cserélned a fájlnevek végződéseit egy alkalmazásban. Egyszer akkor fogtam bele egy ilyen vállalkozásba, amikor a kódjaim egy része .php3 kiterjesztésű oldalakon volt, egy másik részükre mindenképpen szükség volt, a harmadik részük pedig .phtml kiterjesztésű oldalakban szerepelt. Ahelyett, hogy az egyik részt manuálisan kijavítottam volna, hogy illeszkedjen a másikhoz, hogy aztán újra el kelljen végezni ezt a műveletet, annak érdekében, hogy más honlapokhoz is passzoljanak az oldalak, inkább kicseréltem az include-parancsokat, hogy tartalmazzák az include_suffix-et.

A standard definíciók beágyazásához minden PHP-oldal tetején el kell végezned egy egyszeri beillesztést, amit a következő kód mutat. Én minden weboldalamat PHP-módban kezdem, de vannak, akik HTML-módban kezdik és aztán ide-oda ugrálnak. A három soros példa bármilyen oldal kezdetéhez passzol, függetlenül attól, mi következik utána:

```
<?php
require^once("./include.html");
```

Ha a webszerver a .html-t nem PHP-ként értelmezi, akkor .php-t kell használnod, és ha rosszul van beállítva, akkor lehet, hogy a <?php-t ki kell cserélned <?-re. A require_once() csak a PHP4-ben szerepel, de megvan a módja, hogy lekódold a megfelelőjét PHP3-ban; a következőkben ezt mutatjuk be.

Ahhoz, hogy az include-html-fájl kompatíbilis legyen a require_once()-t és az include_once()-t nem tartalmazó PHP-verziókkal, adj egy változót a kódhoz - mint a következő példa mutatja -, amely követi a standard elnevezési megállapodást, és azt jelzi, hogy az include-fájl beágyazása megtörtént. Én a fájl nevét plusz az _included utótagot használom, azaz az include.html-hez egy $include_included nevű változót használok. Az include.html-t beágyazó include()-utasítást feltételes include-ra cserélem és requireQ helyett include()-ot használok, mert a require() nem működik az if()-utasítással. Az include included-ot változóként használom, mert a konstansok problémát okoznak, ha egy feltételben vizsgáljuk őket. Az isset() nem működik konstansokkal, és a konstansokhoz alkalmazható megfelelője, a defined() egy hibaüzenetet eredményez, ha a konstans nincs definiálva:

```
<?php
$xnclude_included = "yes";
define{"include prefix", "./");
define("include_suffix", ".php");                                    ,
```

```
<?php
if(!isset($include_included))
    {
    include("./include.html" ) ;
```

A phpMyAdmin-t, a phpPgAdmin-t és sok hasonló kódot úgy teszteltek, hogy a php.ini-ben kikapcsolták a figyelmeztető üzeneteket, vagyis a kódok azt feltételezik, hogy a hiányzó változók nem szükségesek, és mindenféle gépelési és logikai hibát megengednek. Mivel te bekapcsolt figyelmeztetésekkel fogod futtatni őket, egy rakás üzenetet fogsz kapni a nem definiált változók miatt. Ha egy figyelmeztető üzenetet kapsz, annyit változtathatsz, hogy az if()-ekhez hozzáadod az isset()-utasítást. Az ilyen sorokat:

```
if ( Iprintview)
```

cseréld le erre a kiterjesztett változatra:

```
if (!isset(printview)     or     íprintview)
```

Néhány szabadon hozzáférhető kód azt feltételezi, hogy könnyedén felfedezed a sztrin-gekben eltemetett változókat, de nem minden editor képes (más színnel) kiemelni az eltemetett változókat, ezért megkönnyítheted az életedet és másokét is, ha kiásod a változókat. Ha módosítasz egy kódot és eltemetett változót találsz, mint a **$HTTP_HOST** a következő sorban,

```
$short_realm .= "$HTTP_HOST:local";
```

akkor így nyerheted ki a változót:

```
$short_realm    .=    $HTTP_HOST    .    ":local";
```

Ez a kód a phpPgAdmin 2.3 **lib.inc.php** könyvtárából kimásolva, és egyszerűen azt feltételezi, hogy a $short_realm-változó létezik, ezért egy figyelmeztető üzenetet eredményez, amikor a nem létező változóhoz próbál meg adatokat hozzáfűzni. A hiba kiküszöböléséhez beillesztettem egy sort, amely egy üres változót hoz létre, ha a változó nem létezik:

```
if (! isset <$short__realm) )     { $short_realm    =    "";}
```

Más figyelmeztető üzenetekre is számíthatsz, és sok programozási trükköt kell kijavítanod, hogy megbízható kódot kapj. Ha nyílt forráskódú alkalmazást használsz, mint amilyen a phpMyAdmin, és huzamosabb ideig használod, akkor szentelj egy keveset az időből arra, hogy a fejlesztőknek visszajelzésként elküldöd a változtatásaidat, így a következő verzióból talán kevesebb hibát kell kitisztítani. Ha keresztülrágod magad ezen a könyvön, minden PHP-alkalmazás megtisztítására képes leszel, és akár a nyílt forráskód hősévé is válhatsz.

6. *fejezet*

Adatbázisok

Áttekintés

Számos adatbázist használhatsz a honlapsod kiszolgálására, és számos, adatbázisokhoz hasonló szerkezetű fájltípus szolgál ki egyedi alkalmazásokat. Ha a MySQL nem *igazán* megfelelő' és a PostgreSQL túl bonyolult, akkor egy széles skáláról választhatsz, kezdve a MySQL-nél sokkal kisebb adatbázistól az egészen komolyakig, amelyekhez képest a PostgreSQL a kezdők játékszerének tűnik. Ez a fejezet az 5. fejezetben nem tárgyak adatbázisokról szól.

Az Oracle, mint az első, nagy rendszerekre szabott adatbázis tűnik ki, és számos nagygépes rendszeren elérhető. Az IBM DB2-je az IBM legnagyobb partnereinek kedvenc adatbázisa, akik korábban a legnagyobb számítógéprendszerek használói voltak. A Microsoft SQL szervere pedig azon rendszergazdák millióiból húzott hasznot, akik már az oviban MCSE-tréningre jártak.

Rengeteg különleges igényt kielégítő választási lehetőség van, egy floppylemezre felmásolható adatbázisoktól kezdve az MP3-at és filmeket tároló és lejátszó adatbázisokig. E fejezet révén jobban megértheted, hogyan működnek az adatbázisok, és segít a megfelelő, a PHP által támogatott adatbázis kiválasztásában.

SQL

Az SQL olyan az adatbázisok számára, mint a HTML a weboldalaknak. Az SQL92 a legrégebbi SQL-szabvány; a világon mindenhol használják, és a legtöbb adatbázis ugyanúgy támogatja, mint ahogyan a korai böngészők támogatták a HTML-szabványt, ami rendkívül nagy erőssége az SQL92-nek, ám az SQL92 nem nyújt teljes körű megoldást. Fontos dolgok hiányoznak ebből a szabványból, ezért ahhoz, hogy egy átlagos alkalmazást készíts, nem szabványos SQL-kiterjesztéseket kell használnod. Mindamellett az SQL92 nem követeli meg az adatbázis-szállítóktól, hogy pontosan megmondják, mi a szabvány és mi nem. Ez a Ford egyik takarékos modelljének egy ausztrál hirdetését juttatja eszembe: noha az ausztrál törvények szerint az ablaktörlő kötelező, a Ford az ablaktörlőt az „ingyenes extrák" közé sorolta.

Az SQL99-szabvány az SQL92-t váltotta fel, és a fejlesztők egy *kis* csoportja, pl. a PostgreSQL mögött álló csapat jelenleg is dolgozik az SQL99 továbbfejlesztésén. Azt tanácsolom, hogy SQL99-kompatíbilis adatbázist próbálj választani, jelezve, hogy a fejlesztők a jövőre gondolnak.

Indexeljünk vagy ne indexeljünk

A táblák indexelése növeli az olvasási teljesítményt és lelassítja a frissítést. A PHP-ban semmi nem befolyásolja közvetlenül az indexek használatát, és az egész kérdés lényegében azon múlik, hogyan értelmezi az adatbázis-kezelő szoftver az SQL-t. A PHP-ban két dolgot tehetsz: telepíthetsz egy felhasználóbarát-adatbázis karbantartó interfészt, mint pl. a

phpMyAdmin vagy a phpPgAdmin, amivel könnyen létrehozhatsz és törölhetsz indexeket, másrészt az SQL-lekérdezéseket microtime()-keretbe illesztve összehasonlíthatod a válaszadási időket indexekkel és azok nélkül. A „Visszanyerési idők: a teljesítmény fokozása" című Gyors megoldás a visszatérési idők hatékony naplózását mutatja be.

Kapcsolatok - Relációk

A kapcsolatok legalább annyi problémát szülnek, mint amennyit megoldanak - kérdezd meg bármelyik házas ismerősödet. Ha szeretnéd jobban kézben tartani a dolgokat, és hajlandó vagy lemondani a rugalmasságról, akkor felejtsd el a MySQL-t, ismerkedj meg a PostgreSQL-lel és alapozd relációkra az adatbázisod. Az alkalmazásod megbízhatóbban fog működni, még akkor is, ha a relációk meggátolnak abban, hogy ügyes trükköket alkalmazz az adatokon. Boldogabb és nyugodtabb leszel; nem fogsz egy reggel arra ébredni, hogy azon tűnődsz, mit tettek ezek az idegenek az adataiddal.

Gondolj azokra a mindennapos szabályokra, mint a kézmosás evés előtt, a biztonsági öv bekapcsolása inulás előtt stb., és képzelj el hasonló szabályokat egy adatbázisban. A relációk lehetővé teszik, hogy összekapcsolj sorokat, és vagy együtt dolgozod fel őket, vagy sehogy. Olyan szabályokat biztosítanak, amelyek lehetővé teszik, hogy egy számla kitörlése maga után vonja a számla összes tételének a kitörlését, vagy fordítva, nem engedi, hogy valaki töröljön egy számlát, ha léteznek a számlához kapcsolódó tételek. Ez a fajta szoftveres biztonsági öv elengedhetetlen olyan alkalmazásokhoz, mint a könyvelés, ahol mind a szándékos csalást, mind a felelőtlen hibákat egyaránt nagy összegű pénzbüntetéssel vagy börtönnel büntetik. Miért kockáztassunk ilyesmit egy szoftver összeomlása miatt? Fejlesszük az alkalmazásainkat a kezdetektől megbízható relációkkal.

Állandó kapcsolatok

Még nem láttam elegendő részletes kimutatást a különböző típusú adatbázisokra az állandó kapcsolatok erőforrás-megtakarítására vonatkozóan, vagyis az eredményeid eltérhetnek a tapasztalataimtól. Az állandó kapcsolatok lehetővé teszik a PHP számára, hogy újra felhasználja a már létező adatbázis-kapcsolatot, ha a szerver, a felhasználó, a jelszó és minden egyéb, a kapcsolódáshoz szükséges paraméter változatlan. A PHP csak akkor használhatja az állandó kapcsolatokat, ha Apache-modulként fut, és még néhány egyéb tényező is közrejátszik, mint pl. a tranzakciók közti idő és az adatbázis-kezelő szoftver. Az állandó kapcsolatok erőforrásokat kötnek le és sok memóriát igényelnek (ugyanakkor az újrakapcso-lódás szintén erőforrás- és memóriaigényes).

Ha állandó kapcsolatot állítasz be, és valami meggátolja az állandó kapcsolatot, az eredmény hagyományos kapcsolat lesz. Előfordulhat, hogy soha nem fedezed fel, mikor ez bekövetkezik, és nem leszel tudatában, hogy nem állandó kapcsolatokat használsz. Vannak jelentések, melyek technikai érveket hoznak fel az állandó kapcsolatok használata ellen és vannak emberek, akik minden probléma nélkül állítanak be állandó kapcsolatokat. Ha kétségeid vannak, kérdezd meg a webszervered adminisztrátorát, van-e valamilyen különösebb ok, ami az állandó kapcsolatok használata ellen szól; ha nincs, állíts be állandó kapcsolato-

kat, és hagyd, hogy a PHP megállapítsa, fel tudja-e használni többször ugyanazt a kapcsolatot. Az állandó kapcsolat a felhasználói azonosítón alapszik, ezért egy tízezer látogatót vonzó weboldalon, - ahol mindenki a saját azonosítójával lép be és kap hozzáférést az adatbázishoz - ez tízezer állandó kapcsolatot jelent, ami egyáltalán nem hatékony és megbénítja a webszervert. Ha ez a tízezer látogató a saját azonosítóját csak bejelentkezésre használja, és minden további hozzáféréshez egy közös adatbázis-azonosítón osztozik, akkor közösen használhatnak egy olyan kapcsolatot, amely hatékony, és az állandó kapcsolatot pontosan erre találták ki. A valóságban a tízezer látogató oldallekéréseit 50 *gyermekfolyamat* dolgozná fel, mindegyik külön kapcsolattal. Ötven állandó kapcsolat még mindig sokkal jobb, mint tízezer egyedi kapcsolódás. (A gyermekfolyamatok leírása bármelyik jó Apache-kézikönyv-ben megtalálható.)

Ha egy állandó kapcsolatot indítasz és elfelejted lezárni, vagy a szkript futása megszakad, mielőtt lezárná, akkor a művelethez rendelt erőforrások mindaddig foglaltak maradhatnak, amíg a szervert újraindítják. Tekintve, hogy a modern merevlemezek élettartama 300 ezer óra és minden más hardver hosszabb életű, ezek az erőforrások igen sokáig le lesznek foglalva. Ha állandó kapcsolatokat használsz, iktass be egy PHP shotdown-utasítást, hogy visszavond a nyitva maradt műveleteket.

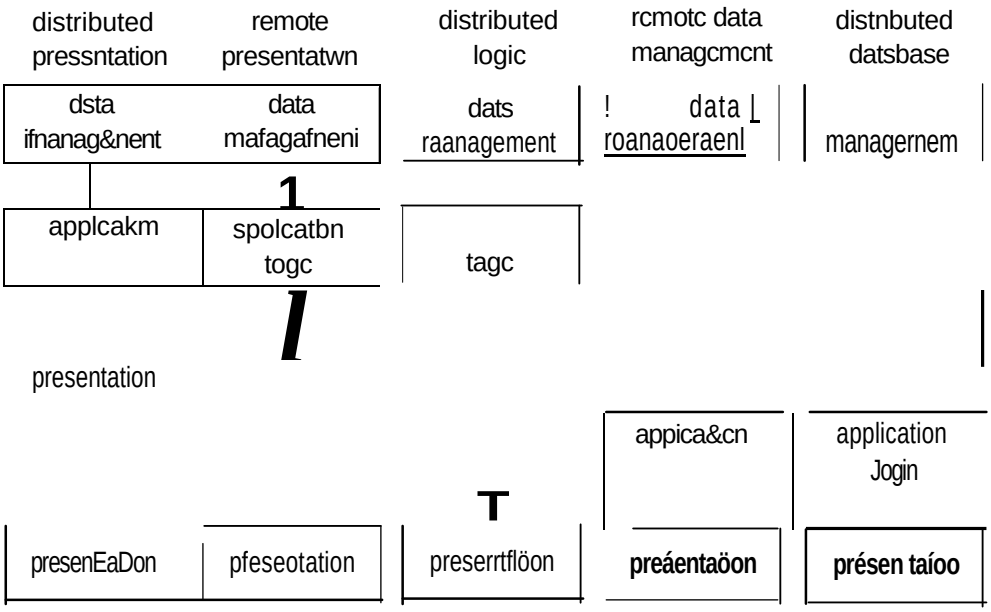
ODBC

Az Object Database Connectivity (ODBC) számos adatbázishoz leegyszerűsíti a kapcsolódást azáltal, hogy adatbázis-független interfészt biztosít, és néhány lehetősége kifejezetten hasznos a hálózaton keresztüli kapcsolódáshoz. Ha a honlapodat több különböző adatbázis szolgálja ki, akkor érdemes megfontolni az ODBC használatát. Az ODBC-nek az a *hátránya., hogy* bizonyos szituációkban növeli a hálózati forgalmat.

A 6.1 ábra az adatfeldolgozás Shuíman/Gartner féle kliens/szerver modelljét mutatja a saját értelmezésemben; a középen látható kritikus vonal a hálózati forgalmat képviseli. A probléma nem sokat változott, mióta sok évvel ezelőtt először ismertettem elképzeléseimet olyan nagyvállalatok vezetőivel, amelyek több tízmillió dollárt költöttek nagygépes rendszerekre. Ha minden adatfeldolgozást a szerveren helyezünk el, és a kliensre alig bízunk többet a megjelenítésnél (pontosan úgy, ahogyan az egy szervert használó weboldalak működnek), akkor kis hálózati forgalmat bonyolítunk. Ha minden lehetséges adatfeldolgozást a kliensre helyezzünk (ahogyan egy honlap működik, ahol a webszerver egy képszerverró'l veszi a képeket és a saját gyorsítótárában raktározza őket), akkor a kezdeti magas forgalom mellett a további forgalom alacsony szintjére számíthatunk. Általában az a legrosszabb eset, ha a munka egyenlően van elosztva két szerver között, mert a két feldolgozó egység közti kapcsolat jelentős forgalmat bonyolít - ez a osztott logikájú rendszer az ábra közepén.

Mikor a PHP-szkripted lefut egy szerveren és ODBC-kliensként kapcsolódik egy adatbázisszerverhez, egyetlen szkript egyetlen oldallekérésre válaszolva hét SQL-lekérdezést is el-küldhet, és némely SQL 100 vagy annál is több sort nyer vissza az adatbázisból. Azaz egyes oldalak 500 sor visszanyerését igényelhetik. Néhány adatbázis API-ja még tovább rontja a helyzetet, mivel minden egyes adatelemhez külön lekérést igényel, ezzel az 500 rekord

5000 mezőlékéréssé gyarapszik. Az ODBC a hálózati forgalom visszaszorítása érdekében annyit nyújt, hogy az **odbc_fetch_row()-utasítás** egy egész sort ad vissza, és az odbc_result()-paranccsal mezőnként lépkedhetsz végig a helyi gyorsítótárban lévő rekordon. Az ODBC-interfész felépítése révén az 5000 mezőlekérés 500 sor lekérésére korlátozódik.



6.1 ábra A kliens/szerver feldolgozás Schulman/Gartner-modellje

500 sor egyenként történő átküldése a hálózaton *mé?* mindig problémát jelent. Ha egy olyan SQL-t küldesz, amely 500 sort választ ki, akkoi mind az 500-ra szükséged van, vagyis az ODBC-nek egy hálózati átvitellel kellene visszanyernie az 500 sort. Ha csak néhány sorra van szükséged, mondjuk az első 10-re, akkor az olyan SQL-funkciókat kell használnod, mint például a **LIMIT.** A hálózati forgalom azért növekszik, mert az ODBC megpróbálja kitalálni, mire van szüksége az alkalmazásodnak, és az adatátvitel egységét másra változtatja, mint amit kértél. Az ODBC akkor lenne tökéletes a hálózaton keresztül történő tranzakciókhoz, ha az ODBC-interfész képes volna egyszerre több sort is visszanyerni, hogy illeszkedjen az SQL **LIMIT** és **START** paramétereihez. Minden SQL-utasításhoz egyetlen hálózati adatátvitel tartozna. Az ODBC jobban hasonlítana az SQL-hez, és a rendszer közelebb állna a 6.1 ábra „távadatkezelés" modelljéhez.

Ha az ODBC a legjobb módja sok különböző adatbázis elérésének, akkor az adatbázisszállítók számára logikus célkitűzés, hogy minimalizálják az adatbázisuk ODBC-n keresztüli használatával járó nehézségeket, és a saját mterfészépítő munkájukat is minimalizálják azáltal, hogy csak egy interfészt fejlesztenek: egy ODBC-interfészt. Logikus, hogy az ODBC szoftvereket a fejlesztők számára is minimalizálják az egyes interfészekre fordított munkát, vagyis az egyes adatbázisokhoz csak egy plusz kapcsolódó modult készítenek. A végeredmény, amire Egységes ODBC-ként szoktak hivatkozni, hogy a PHP ODBC-uta-sításai közvetlenül kapcsolódnak az IBM DB2-höz vagy más adatbázisokhoz, mivel ezek az adatbázisok szabványos ODBC-formátumú API-vel rendelkeznek. Ahol nincs ODBC-interfész, ott a PHP iODBC-t használ, amely egy olyan ODBC-interfész, amely kis plug-

ineket fogad különböző adatbázisokhoz. Hosszú távon több adatbázis fog vagy ODBC-interfésszel vagy ÍODBC plug-innel rendelkezni, ami lehetővé teszi, hogy ODBC-t használj minden adatbázishoz minden platformon. Az egyetlen dolog, amit nem egységesítettek, a dátum és hasonló mezők formátuma, és az a mód, ahogyan az adatbázisok a haladóbb funkciókat tartalmazó kérésekre válaszolnak. Néhány szoftver, ha olyan kéréssel szembesül amellyel nem tud mit kezdeni, egy udvarias nemmel, hamissal, vagy nullával válaszol. Más szoftverek egy hibaüzenetet generálnak. Néhány szoftver azonban a lehető legrosszabb dolgot teszi: egy érvényesnek tetsző választ ad, ami teljesen félrevezető.

DB2

A DB2 az IBM korábbi nagygépes adatbázisát váltotta fel, majd az IBM más platformjain is elterjedt, pl. AIX alapú webszervereken. Miután a Linux kezdett elterjedni az IBM-platfor-mokon egészen a nagyteljesítményű gépekig, az IBM átültette a DB2-t Linuxra is. Az IBM DB2-t mindenhová irányelve eredményeképp az IBM az Oracle-lel fej-fej mellett vezeti a üzleti adatbázisok piacát, annak harmadát birtokolva. A DB2 sok technológiai újítást elsőként alkalmazott az adatbázisok területén, és az IBM nagygépes rendszereit használó nagyvállalatoknál, pl. bankoknál többnyire ezt használják.

Az IBM nemrégen megvásárolta az Informix-adatbázist, vagyis a DB2-ben feltűnhetnek az Informix technológiai megoldásai, és az Informix számos módon fog a DB2-höz kapcsolódni. Ha Informix alapú rendszeren dolgozol, akkor választhatod a DB2-re való áttérést, mivel remek többlépcsős kliens/szerver-funkciókkal rendelkezik, vagy tehetsz egy lépést a nyílt forráskódú PostgreSQL, vagy talán a népszerűbb és kevésbé fejlett MySQL felé.

A DB2 jelenleg AIX-, AS400-, HP-UX-, Linux-, OS/2-, OS/390-, Sun Solaris-, Windows-, Windows NT-, VSE- és VM-platformokon elérhető, ami valamennyi nagy mennyiségű adat Interneten keresztüli továbbítására alkalmas operációs rendszert és szervert lefed. A PHP az *Egységes* ODBC-n, vagyis olyan rendszeren keresztül éri el a DB2-t, amelyben az adat-\ bázis saját alkalmazásfejlesztő interfésze (API) olyan, mint az ODBC, vagyis az ODBC hátrányai nélkül férhetsz hozzá az adatbázishoz. PHP-programozási szempontból a DB2-kód bármilyen ODBC- vagy Egységes ODBC-adatbázissal használható.

SAP DB

Az SAP egy jól ismert német alkalmazásfejlesztő, aki a nagygépes rendszereket használó ügyfelek körében népszerű, és a logisztikai, számviteli és humán erőforrás osztályokat célozta meg. Az SAP DB az SAP-hoz használt adatbázis, amely SQL alapú, és támogatja az olyan objektumokat, mint az XML vagy strukturálatlan adatok. Elérhető az ODBC-n keresztül, és nemrég jelent meg a nyílt forráskódú GPL-hcence. A forráskódot a www.sap.com/solutions/technology/sapdb-oldalról töltheted le.

Noha az SAP DB jól hangzik, a forráskód látszólag C+ + Pascal és Python keveréke és az SAP fejlesztőeszközét kell használnod, ha új komponenseket akarsz kifejleszteni és Össze akarod kapcsolni az eredményt. Senkit nem ismerek, aki szakértője lenne a C + +-nak, a

Pascalnak és a Pythonnak, vagyis az SAP DB hosszú távú fejlesztéséhez külön fejlesztő-csapatok szükségesek, vagy újra kell írni a Pascal és Python részeket valamilyen más nyelven. Véleményem szerint az SAP fejlesztőinek az összes felhasználói felületet azonnal le kellene cserélniük PHP alapú adminisztrációra, mint pl. a phpMyAdmin, az összes Pascalt át kellene konvertálniuk C + +-ra vagy Pythonra, hosszabb távon pedig a C Gnome-hoz használt verziója felé kellene elmozdulniuk, és az összes Pythont C-re vagy PHP-re kellene cserélniük. Valójában azt hiszem az SAP DB-nek meg kellene próbálnia összeolvadni a PostgreSQL-lel, így a PostgreSQL is értékesebbé válna az SAP DB funkcionalitása révén, és az SAP megspórolná az SAP DB fenntartását.

Ha már SAP-ot használsz, akkor az SAP DB-t nem SAP alkalmazásokhoz is használhatod adatbázisnak, de ez korlátozhatja a jövőbeli rugalmasságodat. Ha ODBC-n keresztül használod az SAP DB-t, az nagyobb rugalmasságot biztosít, de kódod még mindig az SAP DB lehetőségeitől fog függeni. Kísérletezz más adatbázisokkal, pl. PostgreSQL-lel, így meg tudod ítélni, melyik alkalmazás igényli az SAP DB-t és melyiket lehet átírni, hogy más adatbázisokkal is kompatíbilis legyen.

Más adatbázisok

A következő adatbázisok különleges funkciókat tartalmaznak és különleges igényeket elégítenek ki, de legjobb, ha ODBC-n keresztül éred el őket vagy lecseréled őket PostgreSQL-re vagy MySQL-re. Előfordulhat hogy dolgoznod kell valamelyikükkel, és ha képes vagy minden kapcsolatot ODBC-n keresztül megvalósítani, akkor jelentős kódújraírást takaríthatsz meg egy új adatbázisra való áttérésnél.

Adabas

A Software AG **(http://softwareag.com)** Adabas-a az Internet korában nagyapának számít az adatbázisok családjában. Az Adabas az IBM IMS-ével versenyzett, amíg az IBM le nem cserélte az új gyermekével, a DB2-vel. Az Adabas semmilyen előnyét nem tudnám megemlíteni az Oracle-lel vagy a DB2-vel szemben, de látogasd meg a cég honlapját, és nézd meg az új adatbázisukat, a Taminot.

Úgy tűnik a Tamino, a relációs adatbázisokat megelőző hierarchikus adatbázisok újbóli megjelenése, ezúttal XML-dokumentumok tárházaként szolgál. Ahelyett, hogy egy relációs adatbázis szöveges mezőiben tartanád az XML-dokumentumaidat, a Tamino lehetővé teszi, hogy egy hatalmas XML alapú struktúrába rendezd őket. Az XML-lel a 20. fejezetben foglalkozunk.

A filePro olvasása

A filePro az fP Technologies, Inc.-től **(www.fptech.com)** származik, és az első megfigyelésem, hogy az fP weblapján PHP helyett JSP-t használnak. A filePro első ránézésre az adatbázis-fejlesztés egy régi, kihalófélben lévő oldalágát képviseli, ahol az adatbázis készítői egy jelentéskészítőt is biztosítanak, és megpróbálják elérni, hogy ezzel a világon mindent meg lehessen oldani. Felsoroltam a PHP filePro-utasításait, de nem javaslom a használatukat,

hacsak nem arra, hogy átkonvertáld az adataidat valami modern adatbázisba, pl. MySQL-be vagy PostgreSQL-be:

- **filepro()** - Beolvassa és ellenőrzi az adattérképet tartalmazó fájlt.
- **filepro_fieldname()** - A mező nevét **adja.**
- **filepro_fieldtype()** - A mező típusát adja.
- **filepro_fieldwidth()** - A mező hosszát adja.
- **filepro_retrieve()** - Visszanyeri az adatot.
- **filepro_fieldcount()** - Az adatbázisban található mezők számát adja.
- **filepro_rowcount()** - Az adatbázisban található sorok számát adja.

Az utasítások olvasási joggal rendelkeznek, és mivel a filePro-ban nem lehet fájlokat zárolni, egy weboldalon keresztüli online-frissítés veszélyes lehet. Ugyanezen okból egyéb eszközöket se használj a filePro-fájlok frissítésére, pl. parancssorból futtatható programokat, mert ezek elérhetőek a weboldaladról.

A filepro()-utasítás argumentumában az adatbázist tartalmazó könyvtár nevét tartalmazó sztring szerepelhet. Megvizsgálja, működik-e az adatbázis, és információkat nyer vissza, pl. a mezők számát. Úgy tűnik, ezen túl még megnyit egy belső kapcsolatot további filePro-utasítá-sok számára. A **iÍlepro_retrive()** egy mezőt ad vissza a sor száma és a mező száma alapján. Ezen a ponton talán felismered, hogy a filePro-adatbázis úgy hangzik, mint egy táblázatkezelő táblázata, nem pedig mint egy relációs adatbázis táblája, és a MySQL sokkal jobban hangzik. A **filepro rowcount()** és **a filepro_fieldcount()** biztosítják azokat a számokat, amelyekkel a ciklusod végig tud lépkedni az adatbázis sorain és mezőin. Miközben a ciklusod végigmegy az első sor mezőin, a **filepro_fieldname()-, a filepro_fieldtype()-** és **a filepro_fíeldwidth()-uta-**sításokkal kaphatsz az output formázásához szükséges információkat.

A FrontBase olvasása

A www.frontbase.com-ról letölthető FrontBase még új a PHP 4.0.6. verziója számára -annyira új, hogy még nem volt alkalmam tesztelni, ezért csak az utasításokat tudom felsorolni. Az utasítások nagyon hasonlítanak az ODBC-hez; a FrontBase sok időt megspórolhatott volna a felhasználók számára, ha az API-jét ODBC-kompatíbilissá teszi:

- fbsql_affected_rows() - Az *előző* lekérdezésben szereplő sorok számát adja vissza.
- **fbsql_autocommit()** - Ki és bekapcsolja az autocommitot.
- **fbsql_change__user()** - Megváltozatja a felhasználói azonosítót a kapcsolatban.
- **fbsql_close()** - Bezárja a kapcsolatot.
- fbsql_connect() - Megnyitja a kapcsolatot egy szerverrel.
- **fbsql_create_db()** - Létrehoz egy adatbázist.
- fbsql_data_seek() - Az eredmény pointerét a következő sorra mozgatja.
- **fbsql_db_query()** -Végrehajt egy lekérdezést.
- **fbsql_drop_db()** - Kitörli az adatbázist.
- **fbsql_errno()** - A legutolsó hibaüzenet számát adja.
- **fbsql_error()** - A legutolsó hibaüzenet szövegét adja.
- **fbsql_fetch_array()** - Az eredmény egy sorát egy tömbként hívja le.

fbsql_fetch_assoc() - Az eredmény egy sorát egy asszociatív tömbként adja vissza.
fbsql_fetch_field() - Egy oszlop információit egy objektumként adja.
fbsql_fetch_lengths() - Az eredmény mezőinek hosszát adja.
fbsql_fetch_object() - Az eredmény egy sorát egy objektumként hívja le.
fbsql_fetch_row() - Az eredmény egy sorát egy tömbként adja.
fbsql_field_flags() - A mező indikátorait adja vissza.
fbsql_field_name() - Az eredmény egy mezőjének nevét adja.
fbsql_field_len() - Egy mező hosszát adja.
fbsql_field_seek() - Az eredménymutatót egy mezőofszetre állítja.
fbsql_field_table() - Egy mezőt tartalmazó tábla nevét adja.
fbsql_field_type() - Az eredmény egy mezőjének típusát adja.
fbsql_free_result() - Felszabadítja az eredmény által lefoglalt memóriát.
fbsql_insert_id() - Az előző beillesztés azonosítóját adja.
fbsql_list_dbs() - Felsorolja az adatbázison elérhető adatbázisokat.
fbsql_list_fields() - Felsorolja az eredmény mezőit.
fbsql_list_tables() - Felsorolja egy adatbázis tábláit. [1]
fbsql_num_fields() - Az eredményben szereplő mezők számát adja.
fbsql_num_rows() - Az eredményben szereplő sorok számát adja. [1]
fbsql_pconnect() - Egy állandó kapcsolatot nyit egy szerverrel.
fbsql_query() - Elküld egy lekérdezést. [1] fbsql_result() - Az eredményadatokat adja. [1] fbsql_select_db() - Kiválaszt egy adatbázist. [1] **fbsql_tablename()** - Egy mezőhöz tartozó tábla nevét adja. [1]
fbsql_warnings() - Be- és kikapcsolja a figyelmezető üzeneteket.

Hyperwave

A Hyperwave-et eredetileg az Institute for Information Processing and Computer Supported New Media (www.iicm.edu) fejlesztette ki, és a Hyperwave (www.hiperwave.com) csinált belőle üzleti terméket. A PHP-ben vannak utasítások a Hyperwave-hez, de a Hyperwave pénzbe kerül, és nem sok előnyét látom új alkalmazásokhoz. A Hyperwave lényegében a weblapokban tárolt meta-információkhoz hasonló azonosító információkkal tárolja a dokumentumokat. Képzelj el egy átlagos weblapot, ahol a szöveget hosszú sztringekben tárolod, a tag-eket pedig kapcsolódó táblákban. Az információt felhasználva újra összeállíthatod a weblapot, plusz az SQL-t használva kiválaszthatod az összes olyan oldalt, amelynek a címében szerepel a *hal* szó. Bármilyen azonosító információt hozzáadhatsz, és az információt tárolhatod a szöveghez kapcsolva, vagy a szöveg egy meghatározott területén.

Manapság automatikusan XML-t használnál, talán egy XML-kompatyíbihs adatbázist, mint a Tamino, vagy egyszerűen a szövegben hagynád az XML tag-eket egy hosszú szöveges mezőben, és az azonosítókat hagyományos relációs adatként tárolnád (és PostgreSQL-re váltanál, hogy kihasználd a relációkat), majd beolvasnád a dokumentumot PHP-ba a PHP vagy a DOM XML-utasításait használva. Akkoriban, mikor az XML elődjének tekinthető

Hyperwave-et tervezték, a PHP XML-támogatása és a nyílt forráskódú adatbázisok messze voltak attól, hogy dokumentumtárolásra alkalmas megoldást nyújtsanak, vagyis a Hyperwave elő'relépés volt. Azonban ma már visszalépést jelent a saját tulajdonú szoftverek felé, amelyek nem illeszkednek egy kis fe)lesztőkörnyezethez, mivel legalább két adatbázis ismeretét és fenntartását igénylik.

A PHP fejlesztői jó munkát végeztek, mikor dokumentálták, hogyan illeszthető a Hyperwave-szerver az Apache-hoz. A Hyperwave átlátszóvá válik, és a weboldalad látogatói csak a PHP által generált oldalakat látják. Szükséged lehet a Hyperwave használatára, ha eleve egy Hyperwave-adatbázisban kapod a dokumentumokat és a Hyperwave hosszú távon is jó választás lenne, ha a fejlesztői nyílt szabványú formátumra helyeznék és időszerű technológiával látnák el.

Informix

Az IBM megvásárolta az Informix-adatbázist, vagyis az Informix-adatbázis jövője bizonytalan. Ezen túl jelen könyv írásakor a PHP Informix 7-hez tartozó függvényeit kifejlesztő programozók éppen a 9-es verzióhoz való utasításokon dolgoztak. Ha felsorolnám az Informix-utasításokat és paramétereiket, az elavult lenne, mire a tinta megszárad. Ha Informix utasításokat kell használnod, akkor küldj egy e-mailt az informix@petermoulding.com-címre, írd meg, milyen verziójú Informix-ot használsz és egy friss Gyors megoldást írok az általad használt Informix-verzióhoz a PHP legfrissebb Informix-utasításait használva.

Ingres II

Az Ingres egy másik adatbázis, amely a Berkeley-ből származik. A neve az Interactive Graphics Retrival System rövidítése. Van egy régi, mindenki által hozzáférhető nyílt forráskódú változat, a modern Ingres II pedig a Computer Associates-től **(www.cai.com/products/ingres.htm)** elérhető. Ha szeretnél példákat látni az Ingres II használatára, lapozd fel az „Ingres II elérése" Gyors megoldást.

InterBase

Az InterBase a Borland **(www.borland.com)** adatbázisa, és van egy külön weboldala a **www.interbase.com** címen. Az InterBase ODBC-interfésszel rendelkezik, vagyis nem kell különleges utasításokat használnod a hozzáféréshez, az alábbiakban pedig áttekintjük a PHP-utasításokat.

Az InterBase Sybase-stílusú aposztrófot használ karakterek kihagyására az SQL-ben (a Sybase-t a fejezet későbbi részében tárgyaljuk). A legtöbb adatbázisban az SQL a "Ted's great adventure"- (Ted nagyszerű kalandja) sztringet így értelmezi "Ted\'s great adventure", de a Sybase és az InterBase a "Ted's great adventure"-sztringet fogja keresni. Az InterBase-nek más furcsaságai is vannak: a **create table stock (brand varchar(60))-uta-**sítás egy **BRAND** nevű mezőt hoz létre, és **BRAND-et** kell begépelned, ha el akarod érni ezt a mezőt; viszont a **create table stock ("brand"** varchar(60))-utasítás egy **brand** nevű mezőt hoz létre.

Az ibase_connect() és az ibase_pconnect() más adatbázisok azonos utasításaihoz hasonlóan sima és állandó kapcsolatot hoz létre egy adatbázissal. A paramétereik az adatbázis elérési útvonala, és az opcionális felhasználói név, jelszó és egyéb opcionális paraméterek, amelyekre nincs szükség egy teszt környezetben. Az elérési út hálózati előtagokat is tartalmazhat más gépeken található adatbázisok elérésére, és ezek közül a TCP/IP protokollhoz használt hostname: forma a legfontosabb. Az ibase_close() bezárja a kapcsolatot, ha elvégezted a feladatot, és állandó kapcsolatokhoz nem használatos.

Az ibase_query() argumentumában egy kapcsolatazonosító és egy SQL-parancs szerepel, és azonnal végrehajtja a lekérdezést. Az ibase_prepare() és az ibase_execute() hasonlóan működnek ODBC-megfelelőikhez, és akkor használhatóak, ha ugyanazt a lekérdezést változó értékekkel többször akarod lefuttatni. Az ibase_free_query() felszabadítja az ibase_prepare() által a lekérdezés számára allokált memóriát.

Az **ibase_fetch_row()** paramétere a lekérdezés azonosítója, és a mezők egy hagyományos tömbjét adja vissza, ugyanúgy, mint a **mysql_fetch_row(),** az ibase_fetch_object() pedig objektumként adja vissza a tömböt. Ha fejdolgoztad az eredményeket, akkor az ibase_free_result()-utasítással tudod felszabadítani az eredményhalmazhoz rendelt memóriát. Ha a lekérdezésedhez tranzakciókra van szükség, akkor az Íbase_trans()-szel indíthatsz tranzakciót, az ibase_commit()-tal hajthatsz végre frissítést a tranzakció végén, és az ibase_rollback()-paranccsal vonhatod azt vissza. Ha e lekérdezés egy hibát eredményez, akkor az ibase_errmsg()-utasítás adja vissza az InterBase legutolsó hibaüzenetét.

Az **ibase_field_info()** paraméterei az eredmény azonosítója és a mező száma, és a mezőkről szolgál információkkal, tehát a mező nevétől vagy típusától függően dolgozhatod fel az adatokat. A MySQL-hez hasonlóan az InterBase is 0-val kezdó'dően számozza meg a mezőket. Olyan sok egyéb közös vonása van a MySQL-lel, hogy az 5. fejezetben található MySQL-példákba beilleszthetted az InterBase függvényeket. Az **ibase_num_fields()** egy eredményhalmaz mezőinek számát adja vissza, amelyeken egy for()-ciklussal mehetsz végig. Az **ibase_timefmt()** beállítja a *mező* formátumát a dátum, az idő és az időbélyegző alapján.

Microsoft Access

A MS Access az egy felhasználós, sima munkaállomások számára valószínűleg a világ legjobb adatbázisa. Az adatbázis képén néhány egérkattintással létrehozhatod a relációkat. Ismerek néhány embert, akik hatalmas adatbázisokat modelleznek Access-ben, aztán az „életnagyságú" adatbázissal kínlódnak, hogy elérjék ugyanazt a szintű funkcionalitást.

Annak érdekében, hogy a MS Access ne szorítsa ki a MS SQL-szervert a kis intranetek piacáról, az Access-ben csak egy felhasználó frissítheti és csak kis számú felhasználó olvashatja az adatokat. Az Access könnyedén elérhető a PHP-val ODBC-n keresztül, tehát demonstrációs célra használhatod az Access-t, és ugyanazt a kódot, ODBC beállítást és SQL-t alkalmazhatod egy másik adatbázisra. Ha az Access minden funkcióját kihasználod, és át szeretnél térni egy nagyobb számú frissítési joggal rendelkező felhasználót támogató adatbázisra, akkor a MySQL helyett PostgreSQL-t (vagy valamely megfelelőjét) kell választanod. A PostgreSQL esetében erőteljesebb fejlesztésekre számíthatsz a jövőben, hí-

szén a PostgreSQL kezelőfelülete igen visszamaradott. (A phpPgAdmin megkönnyíti ugyan a PostgreSQL használatát, de még mindig nagyon kidolgozatlan a MS Access-hez képest.

Mikor egy nagyméretű, alapvető üzleti alkalmazásokat futtató Microsoft SQL Serverrel támogatott weblap számára készítettem adminisztrációs alkalmazásokat, MS Access-t használtam, mert napokat megtakarítottam ahhoz képest, ha az SQL Serveren hoztam volna létre az adatbázist. Egy felesleges, lassú, olcsó PC-n mindössze egy óra alatt futott le a napi naplózást és bizonyos elemzéseket elvégző' program, de ugyanez négy órát igényelt az összes többi alkalmazást futtató és a tranzakciók zárolásával hátráltatott SQL Serveren. A MS Access 2 GB fölött meghal (a Microsoft programozási módszere miatt), de ez alatt a határ alatt tökéletes.

Ha eléred a 2GB-os határt az Access-ben, a gyorsaság és a megbízhatóság tekintetében az a legjobb, ha Apache-csal és PHP4-gyel használt MySQL-re váltasz, és a phpMyAdmin-nal végzed a kezdeti adminisztrációt. Ha relációkat akarsz használni és Windows-on szeretnél maradni, akkor a MySQL kiesett, és az összes többi lehetőség sokkal nehézkesebben telepíthető és adminisztrálható.

Microsoft SQL Server

Ha még nem hallottál volna a Microsoftról, ők azok, akik a Windows-t írták, vagyis az ő SQL-szerverüknek nagyszerűen kellene működnie Windows alatt. Nos, a MS SQL Servernek vannak előnyei, de hasonlóan komoly hátrányosságaí is vannak. A MS SQL Server a Sybase-re alapszik (www.sybase.com), majd megkapta Bili Gates speciális kezelését, amelyet a Windows-ból és Windows NT-ből ismerhetünk. Noha a Sybase-t minden fontos webszerver-platformra kifejlesztették, a MS SQL Server egyedül a Windows NT-re korlátozódik.

A két adatbázis bizonyos mértékben még mmdig kompatíbilis, vagyis használhatod a Sybase Unixos komponenseit, ha Unix alól éred el a MS SQL Servert, de sokkal egyszerűbb és megbízhatóbb ODBC-t használni. Valójában a PHP MS SQL Server utasításainál is egyszerűbb és megbízhatóbb a PHP ODBC utasításait használni, még akkor is, ha a webszervered Windows NT-n fut. Éppen ezért átugrottam az MS SQL Server-utasításokat és az MS SQL Server használóknak azt tanácsolom, lapozzanak az „Adatbázisok elérése ODBC-vel" című Gyors megoldáshoz.

mSQL

Az mSQL-t vagy miniSQL-t a Hughes Technologies Pty Ltd (www.hughes.com.au) fejleszti, és a legjobban úgy lehetne jellemezni, mint a MySQL egy kisebb, könnyedebb verzióját. A MySQL fejlesztői tulajdonképpen az mSQL-bó'l indultak ki, és azért döntöttek egy új adatbázis írása mellett, mert úgy találták,,hogy az mSQL nem növelhető a kívánt mértékben. Eddig még nem használtam mSQL-t, mert a MySQL minden általam használt platformon elérhető. A MySQL a nyílt forráskódú GPL-licenc alatt használható, és ha adatbázist szeretnék cserélni, akkor valószínűleg a magasabb szintű funkcionalitást biztosító PostgreSQL irányába mozdulnék.

Ha mSQL-ed van és PHP-t szeretnél használni hozzá, akkor az 5. fejezetben olvashatsz a MySQL-ről, és használhatod az ott szereplő kódot, csak be kell helyettesítened a MySQL-utasítások helyére az mSQL-utasítások nevét. A Hughes Technologies, amely az mSQL-t értékesíti, egy könyvet is árul az mSQL-ró'l, de az a néhány különbség könnyen kideríthető a PHP mSQL-utasításokat leíró dokumentációjából. A probléma annyiban összetettebb, hogy a Hughes Technologies éppen most készül piacra dobni egy új verziót, és ha ez valódi eló'relépést jelent, akkor a PHP mSQL parancsait is meg kell változtatni.

Oracle

Az Oracle-adatbázist egy sor üzleti alkalmazás, pl. az Oracle-adatbázison alapuló ügyviteli szoftverek mellett az Oracle vállalat készíti. Az Oracle az üzleti adatbázisok piacának harmadát birtokolja, fej fej mellett az IBM DB2-jével. Az Oracle volt az első több platformon széles körben elérhető üzleti adatbázis, és a számítástechnikai osztályokat nagygépes rendszerekkel kiszolgáló Unix-szállítók többnyire ezt választották.

A PHP **ora_** előtagú utasításokat biztosít az Oracle Corporation **(http://oracle.com)** régebbi Oracle-verzióihoz, és újabb, oci8_ kezdetű utasításokat az Oracle 7-es és 8-as verzióihoz. Azon nehézségek alapján, amelyekről az Oracle saját utasításaival kapcsolatban hallottam, úgy találom, hogy ODBC-interfészt érdemes használni. Az ODBC-interfész működésre bírása már nem jelent különös nehézséget Linuxon, Windows **NT-n** még egyszerűbb, és egyéb platformokon valahol a kettő között van.

Az Oracle nagyszerű választás, ha különböző platformokat használsz, különösen, ha eltérő Unix-változatokat. Az Oracle többször tudta magáénak a világ legnagyobb, leggyorsabb vagy legnagyobb adatforgalmat bonyolító aktív üzleti adatbázisának címét. Az IBM DB2-je és az NCR Teradatája vezet a téradatok területén, és az IBM saját platformjain többnyire a DB2 nyújtja a legjobb teljesítményt, de Unix-platformon még viszonylag újnak számít. Előfordulhat, hogy Oracle-projekten kell dolgoznod, de nem fogod az Oracle saját API-jét használni. Noha az SAP-nek megvan a saját adatbázisa, az SAP mögött is találkozhatsz Oracle-lel, mert a nagyvállalatok minden alkalmazásuk mögé Oracle-t tesznek, az Oracle-lel osztják meg az adatokat az alkalmazások között.

Ha szokatlan dolgokat csinálsz az adataiddal, a mód, ahogyan ezt teszed, befolyásolja a jövőbeli lehetőségeidet. Egy ügyfél az adatbázisában két számjegyben tárolta az évszámokat, és ahelyett, hogy megvásárolta volna a négy számjegyet támogató frissítést a legújabb verzióhoz, egy olyan eljárás írásával próbálkozott, amely két számjegyben tárolja a négy jegyű évszámokat. Ha sikerült volna megírniuk a dátumkonvertáló tárolt eljárást, akkor az adatbázishoz láncolták volna magukat, és megnehezítették volna saját maguk számára, hogy a jövőben egy nyílt forráskódú adatbázisra váltsanak. Végül is a tárolt eljárásuk megbízhatatlan volt, és csak úgy tudták kijavítani a kódot, hogy megvásárolták a frissítést a legújabb verzióhoz.

Ha az adatfeldolgozást tárolt eljárásokba zárod az adatbázisodban, akkor tárolt eljárásokkal rendelkező adatbázisokra korlátozod magad. Az Oracle legszélesebb körben hozzáférhető, tárolt eljárásokat támogató alternatívája a nyílt forráskódú PostgreSQL. A legfrissebb Oracle nagyszerű funkciókkal rendelkezik, amelyek lehetővé teszik nagy adatbázisok több

szerver közötti elosztását oly módon, hogy az adatbázis túléli az egyes szerverek leállását. Az Oracle erejét arra használd, hogy a nagy weblapok hasonló kihívásainak megfelelj, de ne írj olyan kódot, amely megakadályozza, hogy a kisebb honlapok olcsóbb adatbázissal működjenek.

1

Ha az összes nem megszokott adatfeldolgozást PHP-ban oldod meg és egyszerű adatábrázolást használsz, akkor bármilyen adatforrás, adatbázisfájlt, vagy HTML-űrlapot egyszerű ábrázolási formában adhatsz hozzá az adatbázishoz és adhatsz át másik adatforrásnak. Az ora-utasítások ugyanúgy működnek, mint az ODBC-parancsok, és az összes különleges adatfeldolgozást PHP-ban (egy jó dolog), SQL-ben (nem olyan jó, de OK) vagy tárolt eljárásokban (általában rossz) képesek megoldani. Az újabb oci8-utasítások lehetővé teszik, hogy az interfészt használd a különleges feldolgozáshoz, ami igen erős korlátozó körülmény (az Oracle használatára korlátozza a lehetőségeidet), és kétségtelenül ez a legrosszabb választás.

Ovrimos SQL Server

4

1

Az Ovrimos S.A. **(www.ovrimos.gr)** Ovrimos SQL Sérvére az egyetlen görög termék ebben a fejezetben. A Ovrimos kicsi és gyors, és inkább a MySQL-lel versenyez, mint a PostgreSQL-lel. Az Ovrimos támogatja a tranzakciókat, ami korábban előnyhöz juttatta a MySQL-lel szemben.

A teljes Ovrimos Web Server alkalmazás egy webszerver és egy adatbázis kombinációja, vagyis lehet, hogy kisebb és kevésbé erőforrás-igényes, mint a Microsoft IIS és SQL Sérvére. Azt nem tudom, hogy felér-e egy Apache MySQL kombinációval, de sokkal többen értenek az Apache-hoz és a MySQL-hez, ezért jobban jársz, ha a saját webszerveredet Apache MySQL-kombinációval valósítod meg. Az Ovrimost néhány webes alkalmazásban előre konfigurált módon használják, és az adatbázist a szerveren úgy kell elérned, mintha távoli adatbázis lenne - egy ilyen szituációban többnyire az ODBC a legjobb választás.

Az Ovrimos-hoz van egy ODBC-imerfész, azaz semmilyen különleges PHP-utasítást vagy telepítési beállítást nem kell használnod; egyszerűen ugorj a fejezet ODBC részéhez. A PHP-ban vannak külön utasítások is az Ovrimos-hoz, **ovrimos_** előtaggal, és szinte megegyeznek a MySQL-utasításokkal, vagyis használhatod az 5. fejezet MySQL-példakód-jait, az Ovrimos-utasításokat behelyettesítve. Az Ovrimos-ban a legújabb MySQL-hez hasonlóan vannak tranzakciók, és az Ovrimos commit-utasítása ugyanúgy néz ki, mint az ODBC commit-ja - még egy ok arra, hogy ODBC-t használ).

SESAM/SQL-Server

A Fujitsu Siemens Computers **(www.fujitsu-siemens.com)** SESAM/SQL-Servere BS2000 operációs rendszerére írt SQL alapú adatbázisa. A BS2000 ma már számos különböző Fujitsu-komputeren elérhető, a kisebb rendszerektől kezdve a nagygépekig. Ha elakadtál a BS2000-rel, akkor a SESAM miatt akadtál el, és a SESAM-ot a PHP SESAM-utasításaival is elérheted, vagy a PHP ODBC-utasításaival a SESAM-hoz használt ODBC-kiterjesztésen keresztül. A SESAM-hoz való ODBC-kiterjesztésért külön kell fizetni, és egy másik vállalat szállítja, a Fujitsu Siemens egyik üzleti partnere.

Mikor a SESAM saját interfészét installálod, a fő részt az Apache-ba telepíted, a PHP-ba pedig egy kis linket teszel - szokatlan megközelítés, amit elég nehéz működésre bírni, és az első kapcsolódáskor lassú, de a további kapcsolatok során már gyors lesz. A SESAM-inter-fész támogatja a *szekvenciális* és *a görgethető* kurzorokat; az első gyors, a második rugalmas. A szekvenciális kurzorok lehetővé teszik, hogy egyszer lépkedj végig a kurzoron, és ha végigmentél az adatokon, akkor valószínűleg eldobhatod őket. Ez alighanem tovább csökkenti az erőforrások igénybevételét azáltal, hogy addig nem nyeri vissza a sorokat, míg nem kérted le őket. Ha hosszú listákhoz szekvenciális kurzort használsz, akkor erőforrásokat takarítasz meg, és ha vissza kell lépned, még mindig le tudod menteni az adatokat egy PHP-tömbbe. A görgethető kurzorok lehetővé teszik, hogy a kurzor körül ugrálj, kereséseket végezz és mindenféle egyéb dolgot, melyeket semmi értelme a kurzorral végezni. Ha egy SQL-utasítással sok adatot választasz ki és a kurzoron belül keresel, akkor nem sikerült hatékonyan használnod az SQL-t. A görgethető kurzorok legtöbb értelmes felhasználása a weblapokat megelőző online rendszerekhez kapcsolódott, de nem olyan jellegű hozzáféréssel, mint amit a mai weblapokhoz használnak. Alaposan vizsgáld meg a görgethető kurzorokat, és váltsd ki őket jobb SQL-utasításokkal vagy PHP-tömbökbe betöltött adatokkal.

Mivel a SESAM-hoz van ODBC-interfész, az ODBC-t javaslom a SESAM saját, a PHP sesam_-eló'taggal ellátott utasításaival használható interfészével szemben. De ha nem teheted meg, hogy megvásárolod a kiegészítő szoftvert a SESAM ODBC-interfészéhez, és találsz valakit, aki konfigurálja a SESAM-ot Apache alatt, akkor használd a PHP SESAM-uta-sításait. A SESAM-utasítások majdnem megegyeznek az ODBC-parancsokkal, egy kivétellel: a sesam_diagnostic() egy állapottömböt ad eredményül, amely a kurzorban található sorok számát tartalmazza, valamint az esetleges hibaüzeneteket és egyéb információkat.

Solid

A Solid Information Technology **(www.solidtech.com)** a beágyazott rendszerekhez alkalmazott szoftverekre koncentrál. A honlapján olyan ügyfeleket sorol fel, mint a Nokia vagy a Nortel, vagyis a termékei illeszkednek a telefonokhoz és a routerekhez. Egy napon lehet, hogy olyan weblapot kell írnod, amelyik a hordozható eszközökhöz illeszkedik, és lehet, hogy PHP-t, Apache 2-t és Solid adatbázist fogsz használni, de ha ODBC-vel éred el az adatbázist, akkor semmilyen különbséget nem fogsz tapasztalni a Solid- és bármely más adatbázis közt.

Sybase

A Sybase, Inc. **(http://sybase.com)** Sybase adatbázisa hasonló az MS SQL Sérvedhez (mivel az MS SQL Server a Sybase-en alapszik), és előnyeiként említhető, hogy sok platformon fut, egy ODBC-kapcsolattal rendelkezik, és sok év fejlesztő munkája áll a hátterében. A Sybase saját PHP-utasításai számos olyan, főleg hibakezeléssel kapcsolatos dolgot megengedne, amit az ODBC-kapcsolat nem, de egyiket sem nevezném fontosnak, és az ODBC-utasítások választéka és képességei folyamatosan gyarapodnak.

Adatbázis-absztrakciós rétegek

Az absztrakciós szintek elválasztják az adatbázisodat a kódodtól és lehetővé teszik, hogy egy külső szoftver technikai igényei helyett az alkalmazásod logikájára és az adataidra koncentrálj. A gyakorlatban az adatbázis-absztrakciós rétegek abba a nehézségbe ütköznek, hogy az adatbázisoknak megvan a saját mezőformátumuk, a saját SQL-változatuk, és különböző módokon kezelik az olyan funkciókat, mint az autoincrement-mezők. A PHP több adatbázis absztrakciós réteget tartalmaz, mivel a programozók több csoportja próbálja megoldani ugyanazt a problémát.

DBA-utasítások

A PHP adatbázis absztrakciós rétege (DBA) utasításai révén az adatbázisok egy csoportjához egy utasításcsoporttal férhetsz hozzá:

- **dba_close()** - Bezárja az adatbázist.
- dba_delete() - Kitöröl egy kulccsal meghatározott **adatelemet.**
  dba_exists() - Ellenőrzi, hogy egy kulcs létezik-e az adatbázisban.
  dba_fetch{) - Egy kulcs által meghatározott rekordot ad vissza.
  dba_firstkey() - Az első kulcsot adja vissza az adatbázisból.
  dba_insert() - Beilleszt egy új rekordot.
  dba_nextkey() - A következő kulcsot adja vissza az adatbázisból.
  dba_open() - Megnyitja az adatbázist.
  dba_optimize() - A nem használt terület eltávolításával optimalizálja az adatbázist.
  dba_popen() - Megnyitja az adatbázist állandó kapcsolattal.
  dba_replace() - Kicserél egy rekordot.
  dba_sync() - Szinkronizálja az adatbázist.

Az adatbázisok csoportja magában foglalja a Sleepycat Software **(www.sleepycat.com)** Berkeley DB-adatbázisát (A Berkeley DB-t korábban DB2-nek hívták, de nem kapcsolódik az IBM DB2-jéhez), valamint a **http://cr.yp.to/cdb.html-** (egy szerver a University of Illinois at Chicago Matematika, Statisztika és Számítástudomány tanszékéről) oldalról letölthető cdb-adatbázist. A következő részben ismertetett DBM-utasítások hasonlóak és a Sleepycat Berkeley DB termékeihez használhatók (A DBM a Berkeley DB elődjének neve).

A PHP-utasítások mindent megengednek, amit az alapjukat képező adatbázis megenged és olyan kérések továbbítását is lehetővé teszik, amelyeket esetleg az nem támogat az adatbázis. Rajtad múlik, hogy elolvasd a dokumentációt és elvégezz néhány tesztet, hogy pontosan meghatározd, mit csinál az adatbázis szoftver a kéréseddel.

A **dba_open()** félúton van az adatbázishoz kapcsolódó és a fájlkezelő utasítások között. Az első paramétere egy elérési útvonal, a második egy hozzáférési mód: az **r** olvasási hozzáférést jelent, vagy w írási hozzáférést, a c az adatbázis létrehozásához és írási/olvasási jog biztosításához használatos, az **n** pedig adatbázis létrehozásához, csonkításához és írási/olvasási jog biztosításához. A harmadik paraméter az aktuális fájlkezelő modul neve, és további paramétereket is hozzáadhatsz, amelyek átadásra kerülnek a fájlkezelőnek. Mivel meg kell ne-

vezned a fájlkezelőt, a dba-utasítások kezdenek eltávolodni az igazi adatbázis-absztrakciótól. A **dba_open()** az adatbázis kezelőt adja vissza, amely hasonló a fájlkezelőhöz vagy az adatbázis-kapcsolathoz, és más dba-utasításokban használható, például a **dba_close()-ban.**

A dba_popen() a **dba_open()** állandó változatát biztosítja. Ha többet szeretnél olvasni az állandó kapcsolatokról, lapozz vissza az **odbc_pconnect()-hez.** A **dba_close()** argumentumában a **dba_open()** vagy a **dba_popen()** által visszaadott kezelő szerepel, és bezárja az adatbázist.

A **dba_fetch()** paramétere egy sztring, amely egy rekordra mutató kulcsot és egy adatbázis-kezelőt tartalmaz. Visszanyeri a kulcshoz kapcsolt rekordot, és visszatérési értéke a siker esetén a rekord, hiba esetén hamis. Ha a rekord visszanyerése előtt kulcs szerint szeretnél keresni az adatbázisban, a **dba_firstkey()** és a **dba_nextkey()** az adatbázis-kezelő alapján az első és a rákövetkező kulcsot adják vissza.

A **dba_insert()** egy rekordra mutató kulcsot tartalmazó sztringet, egy a rekordot tartalmazó sztringet és egy adatbázis-kezelőt fogad el paraméterként. Beilleszti a rekordot és siker esetén igazat, ha pedig a kulcs már szerepel az adatbázisban, hamisat ad eredményül. A **dba_delete()** egy rekordra mutató kulcsot tartalmazó sztringet és az adatbázis-kezelőt fogadja. Kitörli a kulcshoz kapcsolt rekordot és siker esetén igazat, abban az esetben pedig, ha nem találja a kulcsot, hamisat ad eredményül. A **dba replace()** egy rekordra mutató kulcsot és egy, a rekordot tartalmazó sztringet valamint az adatbázis-kezelőt fogadja. Kicseréli a kulcs által hivatkozott rekordot és siker esetén igazat, abban az esetben pedig, ha nem sikerül a csere, hamisat ad eredményül. A **dba_exists()** egy rekordra mutató kulcsot tartalmazó sztringet, és az adatbázis-kezelőt fogadja. Megvizsgálja, hogy létezik-e a kulcs, és ha létezik igazat, ha nem találja, hamisat ad eredményül.

A **dba_sync()** paramétere az adatbázis-kezelő, és az összes puffereit rekord merevlemezre való kiírását kényszeríti, amit el kell végezned, ha egy olyan adatbázisba írsz, mint a Sleepycat DB2-je. Tesztelheted az adatbázisodat, ha egy olyan szkriptet futtatsz, amely a **dba_sync()** nélkül ír az adatbázisba, majd egy másik szkriptet futtatsz, amely kiolvassa a beírt adatot. A **dba_optimize()** egy adatbázis-kezelőt fogad, és bizonyos jellegű optimalizálást hajt végre az adatbázison. Feltételezem, hogy a minimum eltávolítja a törlések és eltérő méretű frissítések következtében keletkezett üres helyet, tehát naponta egyszer vagy minden ezredik törlés után érdemes ütemezni.

DBM-utasítások

A DBM utasításokkal az adatbázisok egy csoportjához egy utasításcsoporttal férhetsz hozzá:

- **dblist()** - A használatban lévő DBM-könyvtárat írja le.
- **dbmcloseQ** - Bezárja az adatbázist.
- **dbmdelete()** - Kitöröl egy rekordot.
- dbmexists() - Ellenőrzi, hogy a kulcs létezik-e az adatbázisban.
- **dbmfetch()** - Visszanyer egy sort az adatbázisból.
- **dbmfirstkey()** - Az adatbázis első kulcsát nyeri vissza.
- **dbmínsert()** - Beilleszt egy rekordot.
- dbmnextkeyO - A következő kulcsot nyeri vissza az adatbázisból. . ■■

- **dbmopen()** - Megnyitja az adatbázist.
- **dbmreplaceQ** - Kicserél egy rekordot.

Az adatbázisok között szerepel a Sleepycat Software **(www.sleepycat.com)** Berkeley DB-adatbázisa és a GNU **(www.gnu.org/directory/gdbm.html)** gdbm-je is. A „DBA utasítások" részben leírt DBA-utasítások ezekhez hasonlóak, és a leírás szerint „a Sleepycat DB2 termékéhez használhatók", ami a Berkeley DB egy régebbi neve (semmi köze az IBM DB2-jéhez). Gyanítom, hogy a Sleepycat azért cserélte le a DB2 nevet, mert az IBM DB2-je már Unixon is elérhető és a PHP-dokumentáció egy kissé lassan követi a névváltozásokat.

A PHP-utasítások olyan műveleteket is lehetővé tesznek, amelyeket a háttérben álló adatbázis nem támogat, ezért olvasd el a dokumentációt, *végezz* teszteket, hogy megbizonyosodj, az utasítás valóban úgy működik az adatbázisoddal, ahogy feltételezed. Ahol az adatbázis mind a DBA-, mind a DBM-utasításokat elfogadja, mint pl. a gdbm, teszteld mindkét interfészt, és pontosan határozd meg, melyikre van szükséged.

A **dbmopenQ** első paramétere egy elérési útvonal, a második pedig egy hozzáférési mód: az **r** olvasási hozzáférést jelent vagy **w** írási hozzáférést, a **c** az adatbázis létrehozásához és írási/olvasási jog biztosításához használatos, az **n** pedig adatbázis létrehozásához, csonkolásához és írási/olvasási jog biztosításához. A **dbmcíose()** a **dbmopenQ** által visszaadott kezelőt fogadja el paraméterként, és bezárja az adatbázist. A **dbmfetch()** paramétere egy adatbázis kezelő és egy rekordra mutató kulcsot tartalmazó sztring. Visszanyeri a kulcshoz kapcsolt rekordot, és visszatérési értéke a siker esetén a rekord, hiba esetén hamis. Ha kulcs szerint szeretnél keresni az adatbázisban, a **dbmfirstkey()** az adatbáziskezelő alapján az első kulcsot adja vissza, a **dbmnextkey()** pedig a rákövetkező kulcsot.

A **dbminsertQ** paraméterei egy adatbáziskezelő, egy rekordra mutató kulcsot tartalmazó sztring, és egy rekordot tartalmazó sztring. Beilleszti a rekordot, és siker esetén igazat, ha pedig a kulcs már szerepel az adatbázisban, akkor hamisat ad eredményül. A **dbmdelete()-é** egy adatbáziskezelő és egy rekordra mutató kulcsot tartalmazó sztring. Kitörli a kulcshoz kapcsolt rekordot, és siker esetén igazat, abban az esetben pedig, ha nem találja a kulcsot, hamisat ad vissza. A **dbmreplace()-é** egy adatbáziskezelő, valamint egy rekordra mutató kulcsot és egy rekordot tartalmazó sztring. Kicseréli a kulcs által hivatkozott rekordot, és siker esetén igazat, abban az esetben pedig, ha nem sikerül a csere, hamisat ad eredményül. A dbmexists() paraméterei egy adatbáziskezelő és egy rekordra mutató kulcsot tartalmazó sztring. Megvizsgálja, hogy létezik-e a kulcs, és ha létezik, igazat, ha nem találja, hamisat ad eredményül. A **dblist()** kilistázza a DBM-utasításokhoz használt DBM-könyvtár tartalmát.

DBX-utasítások

A DBX a **www.guidence.nl/php/dbx/doc/-oldalon** dokumentált, és az utasításai egyszerű összeköttetést tesznek lehetővé az adatbázisok egy csoportjával. Az adatbázisok közt szerepel a MySQL és a PostgreSQL, valamint más ODBC alapú adatbázisok, és a dokumentáció útmutatást tartalmaz arra vonatkozóan, hogyan adhatsz hozzá további adatbázisokat (ha tudsz C-ben programozni):

- **dbx_close{)** - Lezárja a kapcsolatot az adatbázissal.
- **dbx_connect()** - Megnyitja a kapcsolatot az adatbázissal.
- dbx_error() - A legutolsó hibaüzenetet adja.
- **dbx_query()** - Végrehajt egy lekérdezést.
- **dbx_sort()** - Rendezi a lekérdezés eredményét.
- **dbx_cmp_asc()** - Összehasonlítja a sorokat növekvő sorba rendezéshez.
- **dbx_cmp_desc()** - Összehasonlítja a sorokat csökkenő sorba rendezéshez.

Ezek a PHP-utasítások primitívnek tűnnek az 5. fejezetben tárgyalt MySQL-, PostgreSQL-és ODBC-utasításokhoz képest, vagyis mielőtt üzleti alkalmazásban használnád ezeket az utasításokat, ellenőrizd, hogy van-e olyan frissítés a **php.net-en,** amely kiegészíti a fenti utasításcsoportot. A MySQL-utasításokat könnyen és egyszerűen használhatónak találtam, az ODBC-vel néhány finom fogást is alkalmazni lehetett a távoli adatbázisokra, ezért én várnék a dbx-utasítások használatával, amíg megközelítik vagy a MySQL-, vagy az ODBC-utasításokat.

A **dbx_connect()** paraméterei egy modulnév, szervernév, adatbázis, felhasználói név, jelszó és egy változatlanságjelző, visszatérési értéke pedig egy kezelő az adatbázis-kapcsolathoz; egy kicsit olyan, mint a **mysql_connect()** és a **mysql_select_db()** összegyúrva. A modulnév mssql, mysql, odbc, vagy **pgsql** lehet, attól függően, hogy mi van telepítve. A **dbx_close()** a **dbx_connect()** által visszaadott adatbázis-kezelőt fogadja paraméterként, és lezárja a kapcsolatot az adatbázissal.

A **dbx_query()** paraméterei egy adatbázis-kezelő, SQL-állítás és opcionális jelzők, amelyeket túl bonyolult volna itt bemutatni. Végrehajtja a lekérdezést, kinyeri abból a sorokat, és egy olyan objektumot ad vissza eredményül, amely tartalmazza a sorokat, valamint a sorokkal és az oszlopokkal (mezőkkel) kapcsolatos információkat, az adatbázis-kezelőt azonosító információkat és további opcionális, a jelzőktől függő információkat is. A **dbx_error()** argumentumában az adatbáziskezelő szerepel, és a legutolsó hibaüzenetet adja vissza. A dbx_sort()-, dbx_cmp_asc()- és dbx_cmp_desc()-parancsok lehetővé teszik, hogy visszanyerés előtt rendezd az eredményt, de ennek semmi előnyét nem látom, mikor az SQL is tartalmazhat sorba rendezést. Valójában ha az SQL **limit-, start-,** vagy **group by-**klauzuláját alkalmazod, akkor szükségszerűen az eredményhalmaz létrehozása előtt rendezel, vagyis a DBX rendező utasításait nem használhatod olyan sokféleképpen, ahogyan az SQL rendezését.

Gyors megoldások

Adatbázis elérése ODBC-vel

A PHP ODBC-parancsai számos adatbázishoz biztosítanak kapcsolatot a hagyományos ODBC-mechanizmuson keresztül, valamint olyan adatbázisok is elérhetők általuk, mint az Adabas-, IBM DB2-, Solid- és Sybase-, amelyek saját API-je úgy van megírva, hogy emulálja az ODBC-t. A **www.iodbc.org** oldalról letölthető iODBC és a PHP ODBC-utasításai összekapcsolhatóak, és az iODBC-plug-in-ek lehetővé teszik, hogy további adatbázisokat is elérj.

A PHP 4.0.5 php.ini-fájlja a következő példában bemutatott ODBC-opciókat tartalmazza. Néhány beállítást felülbírálhatnak a PHP ODBC-utasításai, ami rendkívül hasznos, ha az Internet-szolgáltató nem biztosít hozzáférést a php.ini-hez, vagy azokban az esetekben, mikor különböző típusú adatbázisokat érsz el, és mindegyik különböző beállításokat igényel:


```
[ODBC]
;odbc.default_db = Not yet implemented (Jelenleg még nincs megvalósítva)
;odbc.default_user = Not yet implemented
; odbc. def ault_jpw = Not yet implemented
; Allow or prevent persistent links. (Engedélyezi vagy gátolja az állandó
  linkeket.)
odbc . allow_jpersistent = On
; Check that a connection is still valid before reuse. (Ellenőrzi, hogy
  ;a kapcsolat még érvényben van-e az újbóli felhasználás előtt.)
odbc.check_persistent = On
; Maximum number of persistent links.  -1 means no limit. (Az állandó linkek
  maximális száma.) (A -1 azt jelenti, hogy nincs korlátozás.)
odbc.max_persistent = -1
; Maximum number of links (persistent + non-persistent). -1 means no limit.
  (A linkek maximális száma (állandó + nem állandó).) (A -1 azt jelenti,
  hogy nincs korlátozás.)
odbc.max_links = -1
  Handling of LONG fields.  Returns number of bytes to variables.
   (A LONG mezők kezelése. A bájtok számát adja vissza a változóknak.)
  0 means passthru. (A 0 passthru-t jelent.)
odbc.defaultlrl = 4096
  Handling of binary data. 0 means passthru, 1 return as is, 2
  convert to char. See the documentation on odbc_binmode and
  odbc_longreadlen for an explanation of uodbc.defaultlrl and
  uodbc.defaultbinmode
  (Bináris adatok kezelése. A 0 passthru-t jelent, az 1 változtatás
  nélkült, a 2 pedig char formátumra alakítást. Az uodbc.defaultlrl és az
  odbc.defaultbinmode magyarázatához nézd meg az odbc_binmode-hoz és az
  odbc_longreadlen-hez tartozó dokumentációt.)
odbc.defaultbinmode = 1
```

Az SQL az ODBC szabványos nyelve, de nem sok adatbázis beszél tiszta SQL-t. A britek a közepes méretű *sört pintnek* hívják, még akkor is, ha a mennyiség nem egy pint, és függetlenül attól, hogy van „csomagolva". Az ausztrálok tetszés szerint keverik a *beér* (sör), *glass* (üveg), *scbooner* (korsó) és a *pint* kifejezéseket, amelyek mind egy üveg sörre vonatkoznak, a kisüvegesre a *stuhbie* kifejezést használják, míg a dobozos sört *canneW* (doboz) hívják (kivéve Sydneyt, ahol sokan *tinnie-nek* - bádogdoboz - hívják a dobozos sört). Ha sörivók nem tudnak megegyezni egy általánosan elfogadott terminológiában, akkor hogyan várhatjuk, hogy az adatbázis-készítők megegyezzenek? A Microsoft néhány szoftverben aposztrófot (') használ a dátumok körül, de az MS Access-ben kettőskeresztet (#) (amit néhány ember *pound-nak* - font - hív), amit a következő példakód mutat. Tovább fokozza az SQL gyötrelmeit, hogy néhány adatbázisnak a megjelenítés formátumában kell megadni a mezőket, míg másoknak belső formátumban. Ha egy dátum mezőt ÉEÉÉ-HH formátumra állítasz be és az SQL-ben is EÉÉÉ-HH-formátumot használsz, és az SQL különös hibaüzeneteket küld a dátummal kapcsolatban, akkor próbálkozz a teljes dátumformátummal, mint ahogy itt látható - lehet hogy az adatbázis a saját belső formátumával megegyező teljes formátumot szeretné megkapni az ODBC SQL-jétől. A második sor a MS Access date()-formátumát mutatja:

```
$sq± = "select * from stock where updated >= #2002-04-20
00:00:00#"; $formatted_date = date("Y-m-d H:i:s");
```

Az **odbc_connect()** kapcsolódik az adatbázisszerverhez, igen hasonlóan a mysql_connect()-hez és a pg_connect()-hez, és a lekérdezésekhez, a **commit-** és a rollback-utasításokhoz használatos kapcsolat azonosítót adja vissza. Az **odbc_close()** hamisat (nullát) ad vissza, ha a kapcsolat megszakadt. Az **odbc_pconnect()** állandó kapcsolatot hoz létre, amely takarékosabban bánik az erőforrásokkal. Ugyanazokat a paramétereket igényli, mint az **odbc_connect(),** és csak akkor működik, ha a PHP Apache-modulként fut, ha CGI-ként, akkor nem. A következő' példák különböző kapcsolatokat mutatnak. Az első a minimális paramétereket használja: **dsn,** felhasználó és jelszó. A **dsn** tetszőleges azonosító lehet, amit az ODBC-szoftvered megenged, hát olvasd el a szoftver dokumentációját.

Kapcsolódás az adatbázishoz 142

Ötlet: Ha Windows-os ODBC-t használsz, akkor ne usert vagy file-t, hanem SYSTEM típust használj, mert a webszerver systemként (rendszer) fut, és nem fogja látni az user vagy file ODBC definíciókat.

Az odbc_connect() lehetővé teszi, hogy meghatározd az ODBC-hez használt kurzor típusát. A négy típust a következő, harmadiktól hatodikig terjedő példákban mutatjuk be. Ha az első lekérdezésedre valamilyen furcsa hibát kapsz, akkor próbáld meg az SQL_CUR_USE_ODBC-kurzortípust, és az ODBC-szoftverhez biztosított dokumentáció alapján használd a többi kurzort. A kidolgozott példák tipikus esetei annak, mikor egy mindenféle funkcióval, például friss hírekkel vagy aukciókkal ellátott portál alkalmazásai más szállítóktól származnak, és mindegyik más adatbázist használ. Az utolsó példa azt mutatja, milyen kapcsolódási sztringet kell megadni néhány ODBC-szoftvernek:

```
Sconnection["vitamins"] = odbc_connect{"vitamins" , "", "");
$connection["stolen goods"] = odbc_connect("auctionsite", "bili",
    "zz15zz");Sconnection["nuclear weapons"] = odbc_connect("NSAserver",
"georgew",
    "w", SQL_CUR_USE_ODBC); $connection["erotica"] =
odbc_connect("everywhere", "peter",
    "Over21", SQL_CUR_USE_IP_NEEDED); $connection["stolen goods"] =
odbc_connect("auctionsite", "peter",
    11 z z 1 5 z 2 " , SQL_CUR_DEFAULT ) ; Sconnection["music"] =
odbc_connect("napster", "peter",
    "blb6bl2", SQL_CUR_USE_DRIVER);
Sconnection["vitamins"] = odbc_connect(
    "DSN=chemicalfactory;UIS=peter;PWD=blb6bl2");
```

Az odbc_exec() és a szinonimája, az **odbc_do()** paraméterei a kapcsolatazonosító és a lekérdezés, visszatérési értéke pedig az eredmény azonosítója, vagy hamis, ha a lekérdezést nem sikerült végrehajtani. A következő' példakódban az **odbc_exec()** futtatja a megadott SQL-lekérdezést, és az eredmény azonosítóját adja vissza. Az **odbc_prepare()** és az **odbc_execute()** együttműködnek, és egy árnyalatnyi különbséget, valamint egy rejtett csapdát nem számítva ugyanazt végzik el, mint az **odbc_exec(). Az odbc_prepare()** által átadott SQL-kérdőjeleket (?) tartalmazhat az értékek helyén, és az **odbc_execute()** a kérdőjeleknek megfelelő paramétereket tartalmazó tömböt ad át az ODBC-nek. Néhány adatbázisszoftver a hatékonyabb erőforrás-kihasználás érdekében előre lefordítja az SQL-t, így az egyes **odbc_execute()**-utasítások egy kicsit gyorsabban futnak le, ami időt takarít meg, ha a szkript ugyanazt az SQL-t futtatja változtatott értékekkel:

```
$sql = "select brand, item, price from prices where item =
'bl2'"; $result = odbc_exec($connection["vitamins"], $sql);

$sql = "select * from prices where item = ? and price <
?";
$sql_result = odbc_prepare(Sconnection["vitamins"], $sql);
$selection = array("bl2", "5.00");
$result = odbc_execute($sql_result, Sselection);
$selection = arrayC'C", "3.50");
$result = odbc_execute($sql_result, Sselection);
```

Valóban hasznos-e tehát az SQL-t előre lefordítani? Ha a szknpted kötegelt adatfeldolgozást végez és minden bemenő rekordhoz lefuttatja a lekérdezést, akkor 300 ezer inputrekordot és műveletenként 1 tizedmásodpercet feltételezve a teljes megtakarítás 8 óra 20 percet tesz ki. Ha ugyanezt a lekérdezést online futtatod oldalanként egy lekérdezéssel, akkor az **odbc_prepare()** és az **odbc_execute()** kombinációja valószínűleg lassabb lesz, mint a **odbc_exec().** Az is előfordulat, hogy az adatbázisod nem fogadja el az **odbc_prepare()-t,** vagy a dokumentáció szerint az adatbázis nem alkalmaz előre lefordítást. Néhány adatbázisban a gyakran használt SQL-műveletek gyorsabbá tehetők, ha egy nézetként beilleszted őket az adatbázisba, vagy ha az adatbázishoz biztosított speciális előfordítót használod. -! H;

Az **odbc_commit()** végrehajtja a kapcsolatban az összes tranzakciót, míg a **odbc_rollbac()** az összesét törli, és visszaállítja a változtatásokat (feltéve, hogy az adatbázis képes rollback elvégzésére). Az **odbc_rollbac()** eredménye igaz, ha minden műveietet visszavont, és hamis, ha valami nem sikerült:

```
Sstatus = odbc_commit{Sconnection["vitamins"]);
Sstatus = odbc_rollback(Sconnection["vitamins"]);
```

Az odbc_close() lezárja az adatbázis-kapcsolatot a szerverrel, és ehhez az odbc_connect() eredményéül kapott kapcsolatazonosítóra van szüksége. Nem zárja le a kapcsolatot, ha még aktív műveletek vannak. Itt a vitamins-szerverrel zárjuk a kapcsolatot:

```
odbc_ciose{Sconnection["vitamins"]);
```

Az odbc_close_all() az összes ODBC-kapcsolatot egy mozdulattal bezárja, kivéve azokat, amelyeknek még nyitott műveletük van.

```
odbc_close_all();
```

Eredmények

Ezen a ponton rendelkezésedre állnak egy kapcsolat megnyitásához és bezárásához, valamint egy lekérdezés futtatásához szükséges kódok, de még nem nyerted vissza az eredményt, ezért ez a rész teljes egészében az eredményről szól.

Az odbc_num_rows() paramétere az eredmény azonosítója, és egy select-utasítás eredményéhez kapcsolódó kurzor sorainak számát, vagy az insert-, delete-, és update-utasítások valamelyike által módosított sorok számát adja. Hiba esetén -1 a visszatérési értéke (mivel 0 értelmes eredményt is jelenthet). Beszámoltak olyan adatbázisokról, amelyek mindig -1-et adnak, valószínűleg az adatbázisszoftver hibájából kifolyólag. A következő kód bemutatja, hogyan lehet végiglépkedni a kurzoron az odbc_num_rows() (feltéve, hogy működik az adatbázishoz) és az odbc_fetch_row() segítségével. A sorok számozása 1-gyel kezdődik, ezért a for()-ciklust 1-től indítjuk; továbbá, mivel az odbc_fetch_row() hamisat ad hiba esetén, egy hibakezelő részt is adtam a programhoz:

```
$sql = "select * from prices";
$result = odbc_exec($connection["vitamins" ], $sql);
$rows = odbc_num_rows($result);

for($r = 1; $r <= $rows; $r++)

    if(odbc_fetch_row($result, $r))

        // Process result with odbc_result()

    else

        // Insert error message here
```

Mikor az odbc_fetch_row() visszanyer egy sort, az adatbázisszerver a hálózaton keresztül átküldi a sort az ODBC helyi memóriájába, ahol az odbc_result()-utasítással férhetünk hozzá. Az odbc_fetch_row()-hoz whileQ-ciklust is használhatsz, hogy gazdaságosabban használd az erőforrásokat. A while()-ciklust a következő példa mutatja. Figyeld meg, hogy a második paraméter, a sor indexe kimaradt. Az odbc_fetch_row() alapértelmezés szerint az aktuális sort olvassa, és a sor pointerét a következő sorra viszi a következő adatkinyerés-

hez, igen hasonlóan ahhoz, ahogyan az each()-utasítás kezeli a tömböket (a magyarázatot lásd a 2. fejezetben):

```
$sql = "select * from prices";
$result = odbc_exec($connection["vitamins"], $sql);
while(odbc_fetch_row($result))
    {
    // Process result with odbc_result()
    }
else
    {
    // Insert error message here
```

Az **odbc_fetch_row()-t** arra is használhatnád, hogy többször végiglépkedj az eredmény kurzoron, de ezzel nagyon leterhelnéd a hálózatot. Jobban járnál, ha beolvasnád a kurzort egy PHP-tömbbe, és ezen mennél végig többször.

Az **odbc_result()** paramétere az odbc_fetch_row() által megadott eredményazonosító és j egy mezőazonosító, és egy mezőt ad vissza az eredménykurzorból. A mezőazonosító lehet a mező száma, 1-gyel kezdődően, mint az 5. sor mutatja, vagy lehet a mező neve, ami a 6. sorban látható:

```
$sql - "select brand, item, price from prices where
type='D'"; $result = odbc_exec($connection["vitamins"], $sql);
while(odbc_fetch_row($result))
    {
    print("<br>Brand: " . odbc_result($result, 1)
        . " , price: " . odbc_result($result, "price"));
```

Ha egyszerre több táblából nyersz ki mezőket és több különböző táblában ugyanolyan nevű mezőid vannak, akkor rendelj egy alternatív nevet a kérdéses név második és további előfordulásaihoz, így név szerint is azonosíthatod őket az SQL-ben, mint a következőkben látható:

```
$sql = "select brand, price, prices.updated,
    quantity," . " stock.updated as datestockchecked" . "
    from prices, stock where stock.item = prices.item";
```

Végtelen sok beszámoló foglalkozik azokkal az ODBC-problémákkal, ahol egy mező túl hosszú, vagy szokatlan típusú, vagy az eredmény túl sok mezőt tartalmaz; azaz próbáld csak a fontos mezőket kiválasztani, így a hálózatot sem terheled feleslegesen, és olvasd el az adatbázis dokumentációjából, hogyan adja át az adatbázis a nagyon hosszú mezőket az ODBC-nek. *Ha kétségeid vannak egy mezővel kapcsolatban, akkor hagyd ki.* Ha problémáid vannak egy SQL-lekérdezéssel, akkor hagyd ki a gyanús mezőket, és addig próbálkozz, amíg nem működik. Esetleg kezdd nulláról a minimális SQL-lel, és lépésről lépésre add hozzá a mezőket.

Az **odbc_result_all()** egy lekérdezés eredményazonosítóját fogadja paraméterként, valamint egy opcionális formázó sztringet, és kinyomtatja a kurzor tartalmát. A kurzor tartal-

mát egy HTML-táblázatba illeszti. A formázó sztring bármi lehet, amit szeretnél hozzáadni a <table> tag-hez. A megjelenítés azonnali, az outputot nem helyezheted el egy sztringben. A táblázat celláit vagy a cellák tartalmát nem lehet formázni, vagyis az odbc_result_all() tesztelési és hibakeresési feladatokra korlátozott. A következő példák a stock- és dump-teszttáblákat jelenítik meg. Az első kód eredménye a 6.2 ábrán látható, a prices-táblából származó eredmény a 6.3 ábrán.

```
$sql = "seiect * from stock";
$result =                                    $sql)
odbc_exec($connection["vitamins'
$result_all = odbc_result„all($result) ;

$sql = "select * from prices";
$result = odbc_exec($connection["vitamins"], $sql);
$result_all = odbc_result_all($result, "border=\"5\"
```

ED	braiid	item	updated	quantity
1	generic	A	2 2002-04-19	00:00:0
2	generic	C	5 2002-04-19	00:00:0
3	**Naíure's** pili	A	1 2002-04-21	00:00:0
4	Naíure's pili	B	4 2002-04-21	00:00:0
5	Naíure's pili	C	3 2002-04-21	00:00:0
6	Naíure's pili	bl2	7 2002-04-21	00:00:0

6.2 ábra Stock-táblázat megjelenítése az odbc_result_all()-lal

TD **brand**	**item**	**price**	**| npdated**
1 igeneric		2.9500	12002-04-19
jgeneric		00:00:00	____

2.9500 29.9500 2002-04-21
Í2002-04-19 00:00:00 00:00:00 38.9500
13 Nature's pilis'A ***s***
fi Nature's pilis B
J5~JNature's pilis jc (35.9500
pJÖÖZ^Ö4-2TÖÖ:00:00
[6 ^NatureVpílís bl2~782^5000 2ÖÖ2-Ö4-2**1** 00:**00**:**00**

6.3 ábra Price-táblázat megjelenítése az odbc_result_all()-lal és egy formázó sztringgel

Vannak helyzetek, mikor szeretnéd tudni a mezők (vagyis oszlopok) számát az odbc_exec()-ből származó eredményben: ekkor az odbc_num_fileds() adja meg a választ. Mivel a legtöbb szkript a sorokat tömbként kapja meg, az eredmény a mezők számának ismerete nélkül is feldolgozható, ezért felteszem, hogy sokkal inkább azokban az esetekben fogod használni ezt a számot, mikor csak a mezők számára vagy kíváncsi, és nem az eredményt akarod feldolgozni. A következő példában az odbc_num_fileds() használata látható, közvetlenül egy korábbi példából származó odbc_exec()-parancs után:

```
$sql - "select * from stock";
$result = odbc_exec($connection["vitamins"], $sql);
$number_of_fields = odbc_num_fields($result) ;
```

Az odbc_fetch_into() paraméterei az eredmény azonosítója, egy sor index, és egy tömb, mint a következő' kódban látható, és egy sort ad vissza az eredményhalmazból. Noha az utasítás olyannak tűnik, mint néhány másik sorvisszanyerő parancs, egy pár árnyalatnyi eltérés magyarázatra szorul. A tömb hivatkozás szerint kerül átadásra, vagyis az odbc_fetch_into() egyszerűen beírja a tartalmat a tömbbe, minden korábbi tartalmat törölve, és a változót tömbbé konvertálja, ha nem ilyen formátumú volt. Ahelyett, hogy más sorvisszanyerő utasításokhoz hasonlóan egy tömböt adna vissza, az odbc_fetch_into() a tömbben elhelyezett mezők számát adja vissza, vagy hamisat, ha nem nyert vissza semmit. Az utasításnak meg kell adnod a sor indexét, vagyis bármelyik sort visszanyerheted, ha pedig nullát adsz meg, akkor a sor pointerét növelve végiglépked a sorok egy halmazán, mint a következő példában látható:

```
$sql = "select * from
stock"; $result =                    "vitamins"],
odbc_exec($connection|               $sql);
if(Sresult)

    print ("<table border=\ "3 \ ">" ) ,-
    $array = "" ;
    while($result_all = odbc_fetch_into($result, 0, $array))
        {
        reset($array);
        print("<tr>" );
        while(list($k,    $v)    =    each($array))

            print("<td>" . $v . }
    print("</tr>" ); }
    print("</table>" );

else

    print("<br>No    result"
```

A példa eredménye a 6.4. ábrán látható.

```
[Tjgeneric        JA~~[2 [2002-04-19 00:00:00
                          2002-04-19 00:00:00
            pilis    A    T    2Ö02-Ő4^2Í
00:00:00 4 ÍNaíure's pilis B 4 2002-04-21
00:00:00    [5    JNature's    pilis    jcT~    (3
[2ŐŐ2ÍI4-2100:ÖÖÍÖO |?|Nature's pilis [bÍ2!7
J
```

6.4 ábra Stock-táblázaí megjelenítése az odbc_fetch_into()-val

Mikor az odbc_fetch_into()-parancsot a PHP 4.0.5 Win32 verziójában használtam, azt tapasztaltam, hogy felülírja a tömböt, de nem állítja vissza a tömb belső pointerét, és a

tesztkód csak egy sort jelenített meg. Beillesztettem a tesztkódba a reset($array)-utasítást, és mindent tökéletesen kinyomtatott. A legtöbb tömbfüggvény vagy a tömb végéhez adja a sort, vagy létrehoz egy ú) tömböt, és a tömb elejére állítja a pointert, ezért küldtem egy beszámolót a rendellenességről a **php.net-re.**

Megjegyzés: Ha gyanúsnak találod egy PHP-utasítás működését, nézd meg a hibákról szóló beszámolókat a http://PHP.net/bugs.php-oldalon. Mielőtt elküldenél egy hibáról egy beszámolót, először olvasd el a PHP-dokumentációt, a php.net FAQ-jait, alaposan tanulmányozd át a hibajelentéseket, és készíts olyan átfogó tesztkódot, amely minden lehetséges variációt megmutat még a hiba jelentése előtt. Ezután szedd össze a bátorságodat, és jelentsd a hibát. A segítséged javíthat a PHP kódján vagy dokumentációján.

Hibák

Mi történik, ha egy hiba van az ODBC-ben vagy az adatbázisban? Az **odbc_error{)** egy hatjegyű hibakódot tartalmazó sztringet ad eredményül, az **odbc_errormsg()** pedig egy hibaüzenetet. Mindkettő egy üres sztringet ad, ha nem volt hiba. Mindkét utasításhoz megadhatsz egy kapcsolatazonosítót, és a megadott kapcsolat utolsó hibájáról fognak információkat adni, de kapcsolatazonosító nélkül is használhatod őket, és ekkor az összes kapcsolatban legutoljára felmerült hibát adják vissza. A következő példakód azt mutatja, hogyan jelenítheted meg a hibainformációkat a lekérdezés után:

```
$sql = "select brand, item, price from prices where type=
'bl2'",-if(!$result = odbc_exec($connection["vitamins"],
$sql))
   {
   print("<br>ODBC error number " . odbc_error(Sconnection["vitamins"]) .
       ", message: " . odbc_errormsg($connection["vitamins" ]));
   }
```

Az MS Access olyan viselkedést mutat, amely tipikus bizonyos adatbázisokra, ezért itt MS Access-t használtam egyrészt a saját problémáinak illusztrálására, másrészt hogy megmutassak néhány dolgot, amire más adatbázisoknál is érdemes figyelni. Az MS Access több embernek lehetővé teszi, hogy megnézze az adatokat, de egy adott pillanatban mindössze egy személy frissítheti őket. Ha akkor próbálod meg elérni az adatbázist, mikor az meg van nyitva frissítésre, akkor a következőhöz hasonló hibaüzenetet fogsz kapni:

```
Warning: SQL error: [Microsoft][ODBC Microsoft Access Driver]
Could not use '(unknown)'; file already in use., SQL state S1000
in SQLConnect (Figyelem: SQL-hiba: [Microsoft][ODBC Microsoft
Access Driver]-t nem lehet megnyitni (ismeretlen), a fájl már
használatban van.)
```

Mezőkkel kapcsolatos információk

Mikor visszanyered az eredményt, és szeretnéd úgy módosítani a feldolgozást, hogy illeszkedjen hozzá, akkor jól jön, ha van egy listád a mezőnevekről, a hosszukról és egyéb információkról. A következő program az összes információt visszanyeri az eredmény valamennyi mezőjéről, azaz minden mezőt pontosan a saját igényeidnek megfelelően dolgozhatsz fel. Az egyetlen különbség az ODBC-verziók és más adatbázisok hasonló kódja közt, hogy az index nem 0-val, hanem l-gyel kezdődik. A kód valamennyi, a mezők infor-

mációinak begyűjtésére használatos ODBC-utasítást bemutatja:

```
function odbc_fields($result)

    $number_of_fields = odbc_num_fields(Sresult};
    for($i = 1; $i <= $num_of_fields; $i++)

        $array[$i]["name"J = odbc_field_name($result, $i);
        Sarray[$i] ["column"] = odbc_field_num(Sresult,
            Sarray[$i]["name"]);
Sarray[$i] [ "type"] = odbc_field_type(Sresult, $i); Sarray[$i]
["length"] = odbc_field_len($result, $i); //    Sarray[$i]
["length"] = odbc_field_precision($result,
Sarray[$i]["scale"] = odbc_field_scale($result, $i)

    if(!isset($array)    {Sarray    =
    return($array);      falsé;


$sql - "select * from stock";
Sresult = odbc_exec(Sconnection["vitamins"], $sql);
if <$result)

    $fields    =                                    5
    odbc_fields($result);
```

Az **odbc_field_name()** az oszlop indexhez tartozó mezőnevet, az **odbc_field_num() pedig** a megadott nevű mező indexét adja vissza, vagyis az *előző* kód csak szemléletesen állítja be a **Sarray[$Í] ["column"]** értékét az **odbc_field_num()** utasítással, mivel a szám már ismert **volt.** Az **odbc_field_precision()** az **odbc_filed_len()** szinonimája, ezért csak megjegyzésként szerepel az előző kódban. Az **odbc_field_scale()** a lebegőpontos számok tartományát adja meg.

További utasítások

Az **odbc_autocommit()** az alapbeállítás szerint bekapcsolt **autocommit** beállítást kapcsolja ki, vagy újra bekapcsolja, ha szükséges. Az **autocommit** kikapcsolása egy tranzakciót indít, amelyet az **odbc_commit()-,** vagy az odbc_rollback()-utasításokkal lehet lezárni. A következő mintakód első sora a jelenlegi állapotot adja vissza, a második sor be-, a harmadik pedig kikapcsolja az **autocommit-ot:**

```
$status    =    odbc_autocommit($connection["vitamins"]); Sstatus
=    odbc_autocommit($connection["vitamins"],     true); $status
=    odbc_autocommit($connection["vitamins"],     falsé);
```

Az odbc_binmode()-utasítással beállíthatod, hogy az ODBC-n keresztülhaladó bináris adatokat karakteres ábrázolásban kapd meg. Az első paramétere az eredményhalmaz azonosítója, a második pedig a következő példakódban bemutatott három választási lehetőség. Az **ODBC_BINMODE_PASSTHRU** változtatás nélkül adja át a bináris adatokat, az **ODBC_BINMODE_RETURN** hatására abban a formában kapod meg az adatokat, ahogyan az adatbázisban szerepeltek, **ODBC_BINMODE_CONVERT** pedig a hexadecimális

sztnng ábrázolási módba konvertálja a bináns adatokat. Ha az eredményhalmaz létrehozása előtt szeretnéd visszaállítani a bináris mód alapértékét, akkor az eredmény azonosítóját állítsd nullára az odbc_binmode()-utasításban. Ne feledd, hogy a beállításaidat az odbc_longreadlen()-utasítás is befolyásolja (amit a következő' részben tárgyalunk):

```
$status = odbc_binmode($result["vitamins"]   ODBC_BINMODE_PAS STHRU)
$status - odbc_binmode($result["vitamins"]   ODBC_BINMODE_RETURN);
$status = odbc_binmode($result["vitamins"]   ODBC_BINMODE_CONVERT);
```

Az odbc_cursor() paramétere az eredményhalmaz azonosítója, és a kurzor nevét adja vissza, lásd a következő kódot:

```
print("<br>Current cursor:      odbc_cursor($result["vitamins"];
```

Az odbc_free_results() felszabadítja az erőforrásokat, és olyan szkriptekben lehet hasznos, melyek sok kurzort hoznak létre, vagy nagyon hosszú kurzorokat használnak. A következő példában az eredmény feldolgozása után felszabadítjuk az eredményhalmazt (a feldolgozást a tömörség érdekében kihagytam). Ha kikapcsolod az autocommit-ot, majd egy tranzakciót kezdeményezel és kiadod az odbc_free_result()-parancsot, akkor a tranzakció törlődik, vagyis légy határozott, és először használd az odbc_commit()-, vagy az odbc_rollback()-utasítások valamelyikét. A felszabadított erőforrások mennyisége változó: minél hosszabb a kurzor vagy a szknpt, annál nagyobb mértékű a megtakarítás:

```
$sqi = "select brand, item, price from prices where
type='D'"; $result - odbc_exec($connection["vitamins"],
$sql); while (odbc_f etch_rov; ( $result) )

    // process result in

here odbc_free_result($result);
```

Egy online vásárlásra szolgáló weblapon gyakran egy sok alkalmazást és felhasználót kiszolgáló központi adatbázisszerver a szűk keresztmetszet - tehát minden felszabadított kurzor egy további vásárlót jelent, aki vásárolhat a boltban. Minden nyitva hagyott kurzor, különösen a végre nem hajtott tranzakciókhoz kapcsolódó, olyan vásárlót jelent, aki otthagyta lassú honlapodat és inkább a versenytársad oldalán vásárol (hacsak a konkurencia nem kezeli még hanyagabbul a tranzakciók végrehajtását és az erőforrások felszabadítását). Néhány adatbázisnál egy nyitva hagyott frissítési tranzakció gátolhatja vagy súlyosan lelassíthatja az olvasási tranzakciókat. Az olyan szkriptekben, amelyek hosszú kurzorokat olvasnak, érdemes fontolóra venni azt a lehetőséget, hogy a kurzort közvetlenül beolvasod egy tömbbe, ezáltal felszabadítod a kurzort, és az adatfeldolgozást a tömbből végezd.

Az odbc_gettypeinfo() egy kurzort ad vissza, amely a forrásadatbázis összes elérhető adattípusát és az egyes adattípusok tulajdonságait tartalmazza. A következő mintakóddal a 6.5 ábrán látható táblázatot kaptuk a MS Access-ből, amelynek 18 tulajdonságoszlopa volt, ezek közül néhány csak egy adattípusra vonatkozott. A 6.6 ábra a PostgreSQL-adattípuso-kat mutatja, amelyeknek 14 tulajdonságoszlopa volt. Az eredmény jelentős eltéréseket mutat az MS Access-hez képest, és az adatbázisok közötti különbséget hivatott szemléltetni, különösen az adatformátumok közötti, az ODBC által nem kezelt különbségeket:

TYPE NAME	DAT	TYPE COHJMN_SEE	jLTTERALPREFIX^ITERALSUrnX
GUID	-11	|36	
BTT	!|-7		
LONGBINARY	;-4		
"BYTE	-6	|3	
		11073741823	jOx
		jJ255~	
BINARY	1-2	1255	Ox
,LONGCHAR			
CHAR			
CURRENCY	2		
INTEGER	4	10	
COUNTER	4	10	
SMALLJNT	5	5	
REÁL	7	7	
DOUBLE	j|8	15	
		1255	
DATETIME	U	19.....	
VARCHAR			

6.5 ábra Típusinformáció MS Accessből ODBC-vel és odbc_gettypeinfo()-val

\|TYPE JVAMÉ IDATA TYPE PRECISION LITERALPREÍTX LITERAL SUFFTX

J.1? .....

■1-5 iintS

char	-7	1	1	9
char	1	254		9
dalé	9	10		
numeric	3 ...	1000		
float8	8	15		
float8	«	15		
iní4	4	10		
,lo	-4	■4		
text	-í	8190		
numeric	2	1000		
floaí4	7	7		
iní2	5	5		
time	10	8		
datetime	11	19		
int2	-6	5		
	12			
		254		
bytea	-3	254		
varchar				

6.6 ábra Típusinformáció PostgreSQL-ből ODBC-vel és odbc_gettypeinfo()-val

```
$result = odbc_gettypeinfo($connection["vitamins"]);
```

```
if($result)
    {
    $result_all = odbc_result_all($result, "border=\"3\"":
```

Az odbc_longreadlen() az odbc_binmode()-utasítással együttműködve a hosszú mezők kezelésének beállítására szolgál, mint ahogyan néhány adatbázisban a text, a MS Access-ben a memo és bármilyen long, long binary vagy blob nevű mező. Az első paraméter az eredmény azonosítója, a második pedig a maximális hossz bármely mező számára, amit átad az adatbázis. A nulla érték jelentése, hogy a hosszú mezők csonkítás nélkül kerülnek átadásra. A következő példában egy korábbi kód eredményt kiolvasó részébe, az odbc_exec()-parancs után beillesztettük az odbc_longreadlen()-t, amely 200 karakterre korlátozza a hosszú mezők méretét:

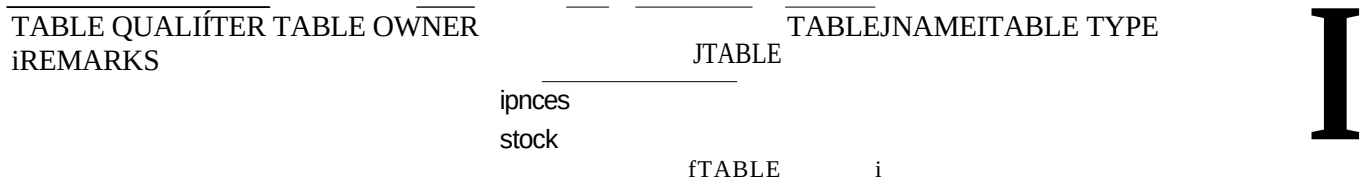
```
$sql = "select * prices";
$result = odbc_exec($connection["vitamins" ] , $sql);
odbc_longreadlen($result, 200))
while(odbc_fetch_row(Sresult))
```

Az odbc_setoption()-parancs valami olyasmi, mintha kinyitnád egy szalagtisztító készletedet, és egy hatalmas acélkalapácsot helyeznél a vattapamacsok mellé: vagyis ne használd, amíg nem tudod pontosan, hogy mit is csinál. Az odbc_setoption() lehetővé teszi, hogy opciókat állíts be a kapcsolathoz és az SQL-utasítások végrehajtáshoz, de az eredménye megbízhatatlan, megjósolhatatlan és alkalmanként kifejezetten antiszociális, mivel a lehetséges beállítások adatbázisonként és az adatbázisszoftverek verzióiként változnak. Ami számodra működik, végzetes lehet valaki más számára, aki a te kódodat próbálja használni. Az első paramétere a kapcsolat- vagy az eredményazonosító, a második értéke 1 a kapcsolatopciókhoz és 2 az eredményekhez, a harmadik az opció száma, amit az ODBC-szoftvered dokumentációjából tudhatsz meg, a negyedik pedig az opció értéke. A következő példa néhány ODBC-szoftvernél és adatbázisnál 60 másodpercre állítja az időtúllépési korlátot. Más szoftverekkel lehet hogy nem működik, illetve az is lehet, hogy egy végzetes értéket állít be egy másik paraméternek:

```
odbc_setoption($result, 2, 0, 60);
```

Az odbc_tables() egy kurzort ad eredményül, amely az adatbázis összes tábláját tartalmazza. Legalább a kapcsolatot meg kell adni, valamint számos opcionális paraméter megadható, név szerint: qualifier, owner, name és types. Az owner és a name SQL-stílusú keresési minták használatát teszi lehetővé. A % nulla vagy több karaktert helyettesíthet, a _ pedig egyet. A types-paramétertípusok vesszővel elválasztott felsorolását tartalmazhatja, mint pl. "TABLE_VIEW ", de az adatbázis bármelyik opciót figyelmen kívül hagyhatja, ha nem tetszik neki. (Például a PostgreSQL támogatja a nézeteket, de amikor nézeteket akartam, csak üres sorokat adott vissza.) A kódot, amit a táblák felsorolására használtam, a következő példa mutatja, az eredmény pedig a 6.7 ábrán látható:

```
$result = odbc_tables($connection["vitamins"]);
if($result)
    {
    $result_all    =    odbc_result_all($result,     "border = \" 3\ " ") ;
```



6.7 ábra A PostgreSQL-táblázatok listája odbc_tables()-el megjelenítve

Az **odbc_tableprivileges()** felsorol minden, a táblákhoz kapcsolódó előjogot. Ugyanazokat a paramétereket kell megadni, mint az odbc_tables()-utasításhoz, és ugyanolyan outputot állít elő. Az odbc_tableprivileges() csak a kitüntetett táblákat sorolja fel, vagyis ha az összes tábláról szeretnél egy felsorolást, amely külön megjelöli a kitüntetett táblákat, akkor először az odbc_tables()-t kell futtatnod, majd az eredményt összevonni az **odbc_tableprivilages()** eredményével.

Új utasítások

Ebben a részben új, hasznos ODBC-utasításokat sorolok fel, amelyek az általam ODBC-vel tesztelt adatbázisokkal nem működtek, nézd meg a **php.net-en,** hogy van-e frissített dokumentáció ezekhez a parancsokhoz. Az adatbázis dokumentációját is nézd meg, mert lehet, hogy az adatbázis nem biztosítja a megfelelő adatokat ODBC-n keresztül.

Az odbc_primarykeys() meghatározza az elsődleges kulcsot a táblában, az odbc_foreignkeys() pedig a táblában lévő idegen kulcsokat vagy a tábla elsődleges kulcsára mutató idegen kulcsokat más táblákban. Mindkettő fontos, ha olyan szkripteket készítesz, amelyek dinamikusan frissítik a táblákat, és ha hibásan kezeled őket, tönkreteheted a táblák közötti relációkat.

Ha az adatbázisod támogatja a tárolt eljárásokat, a PHP-ban van két új függvény: az **odbc_procedures()** felsorolja a tárolt eljárásokat, az **odbc_procedurecolumns()** pedig segít azonosítani a tárolt eljárásokban használt oszlopokat (mezőket). Akkor lehet szükséged ezekre az utasításokra, ha dinamikusan akarod kiválasztani, hogy melyik tárolt eljárást igényli a feladat. Kevés adatbázis támogatja a tárolt eljárásokat. Ha mégis ilyenre bukkansz, számos tárolt eljárás ugyanazokat a mezőket éri el némileg különböző stratégiával, hiszen úgy készítették el őket, hogy bizonyos lekérdezéseket optimalizáljanak.

Az **odbc_specialcolumns()** egy másik új utasítás, amely azon mezők (oszlopok) azonosítását szolgálja, melyek egyértelműen meghatároznak egy táblát, vagy az értékük automatikusan növelt. Akkor használhatod, ha dinamikusan akarsz hozzáférést adni egy táblához. Ha egy adatbeviteli képernyőt a táblainformációk alapján készítesz el, tudnod kell, hogy mely mezők szerepelnek az elsődleges kulcsban, hogy kötelezővé tehesd ezen mezők megadását. Ha egy frissítési oldalt hozol létre, akkor ki kell hagynod az automatikusan növelt értékű mezőket a felhasználó által frissített mezők közül és a frissítő SQL-parancsból.

Az odbc_statistics() statisztikát készít egy tábláról, de nem minden adatbázis gyűjt statisztikákat. Néhány csak akkor készít statisztikákat, ha speciális segédprogramokat futtatsz, és azok, amelyek statisztikákat készítenek, lehet hogy használják azokat a lekérdezések optimalizálásához, lehet hogy nem. A statisztikák alapján meghatározhatod, hogy hozzáadj-e egy indexet egy mezőhöz, vagy eltávolíts-e egy meglévő indexet, és hogy milyen gyakran tömörítsd az adatbázist.

"ĩ

r

¥

Adatbázis elérése DBA-utasításokkal

A következő' kód egy egyszerű frissítést végez gdbm-adatbázisban. Egy **$search** keresési kulcsot, és egy $new_value-változót használ, amely a megtalált és lecserélendő rekord új értékét tartalmazza. Az adatbázis megnyitásához a c-paramétert használja, amely lehetővé teszi az adatbázis létrehozását, ha az nem létezik, és van benne egy hibaüzenet, ha a csere nem sikerül, de más hiba esetén nem ad hibaüzenetet. A fájlnév perjeleket (/) tartalmaz, mert a kód Windows NT-n íródott. Az NT-n a / szimbólum vagy a visszaperjel (\) egyaránt használható, Unix vagy Linux alatt perjelet kell használni, míg a Windowsvisszaper jelet igényel. Figyeld meg a while-ciklust, amelynek semmi nincs az argumentumában. A ciklus a **dba_firstkey()-t** használva megy végig a kulcsokon. Az egyetlen követelmény, hogy fejezze be a ciklust, ha egyezést talál, vagy ha vége a fájlnak:

```
$search    =    "frog";
$new_value = "Frogs are small, wet and somé have fat juicy legs"
    . " that are a great source of protein for birds and somé humans." ; $db
= dba_open("t:/test/db.dbm", "c", "gdbm"); $key = dba_firstkey($db);
while($key != $search and $key = dba_firstkey($db)


If{$key == $search)

   if(dba_replace($key, Svalue, $db))

      print("<br>Replace worked.");

   else

       print ("<;br>Replace failed.");
       }

dba_close($db);
```

Adatbázis elérése DBM-utasításokkal

A következő kód egy mintafnssítés gdbm használatával. Annyit csinál, hogy rákeres egy kulcsra, a Ssearch-re, és a kapcsolódó rekordot egy új értékre cseréli, amit a **$new_value** tartalmaz. Az adatbázis megnyitásához a c-paramétert használja, amely engedélyezi az adatbázis létrehozását abban az esetben, ha az nem létezne. Van benne egy hibaüzenet, ha a csere nem sikerülne, de semmilyen más hiba esetén nem ad hibaüzenetet. Ha megnézted az előző Gyors megoldást, amely a DBM-utasításokat használja, azt láthatod, hogy a DBM-utasítások szinte teljesen megegyeznek a DBA-parancsokkal, tehát a kód nagyon hasonló:

```
$search    =    "dog";
$new_value = "Dogs are small, have wet noses and fást legs,"
. " eat a deal of protein and are eaten by somé humans.";
$db = dbbopen("t:/test/db.dbm", "c", "gdbm");
$key = dbmf irstkey ($db) ;                                          ...
```

```
while($key != $search and $key = dbmfirstkey($db)

If($key == Ssearch)
    if(dbmreplace($db, $key, $value))
       print("<br>Replace worked.");
    else
       print("<br>Replace failed.");

dba_close($db);
```

Adatbázis elérése DBX-utasításokkal

A következő kód egy egyszerű frissítés MySQL-adatbázis használatával. Egy egyszerű lekérdezést tartalmaz, és megjeleníti az eredmény első mezőjét:

```
Ssql = "select * from fruit";
$connection = dbx_connect("mysql", "", "food", "peter",
"xxOOyyy");
$result = dbx_query(Sconnection, $sql);
print("<table border=\"3\">");
for($r = 0; $r < $result->rows; $r++)

    print("<tr>") ;
    for($c = 0; Se < $result->cols; $c++)

         print("<td>" . $result->data[$r][Se]

      print("</tr>");

  print("</table>");
  dbx_close($connection);
```

Ingres II elérése

Az Ingres Il-ről a www.cai.com/products/ingres.htm oldalon olvashatsz. Korábban voltak érvek, amelyek az Ingres-adatbázis vásárlása mellett szóltak, de ma az Ingres egy olyan vállalat (a Computer Associates tulajdonában van), amely számos adatbázist megvásárolt, és néhányat nem fejlesztett tovább. Az Ingres továbbá kapcsolatonként egy lekérdezésre vagy egy tranzakcióra korlátozott, vagyis bizonyos alkalmazásokhoz nem használhatod, ami még egy ok arra, hogy inkább PostgreSQL-t válassz. Ha mégis szeretnéd elérni az Ingres-t, talán éppen egy másik adatbázisba való konverzió érdekében, akkor ellenőrizd az Ingres-tá-mogatást a php.ini-ben, és használd a következő utasításokat.

Az **ingres_connect()** paraméterei az adatbázis neve, a felhasználói azonosító és a jelszó, és egy kapcsolatazonosítót vagy kudarc esetén hamisat ad vissza. Ha nincsenek megadva, mindhárom paraméter alapértékét a php.ini beállításai határozzák meg. A kapcsolatazonosító alapértéke minden további Ingres-utasításban a legutoljára használt kapcsolat, vagyis nem kell megadnod a kapcsolatazonosítót, hacsak nem használsz több kapcsolatot. Az ingres_pconnect() állandó kapcsolatot biztosít, és ugyanazokat a paramétereket igényli, mint az **ingres_connect().** Az ingres_close() bezárja az aktuális kapcsolatot, és a MySQL-hez hasonlóan nem kell meghatároznod a kapcsolat azonosítóját.

Az **ingres_query()** paramétere egy SQL-lekérdezést tartalmazó sztnng és egy opcionális kapcsolatazonosító, és siker esetén igaz, kudarc esetén hamis a visszatérési értéke. Az **ingres_fetch_row()** siker esetén az eredmény sorait nyeri vissza, de a **mysql_error()**-utasí-tásnak, amely kudarc esetén egy hibaüzenetet ad, nincs megfeleló'je. Az **ingres_query()** a lekérdezést automatikusan egy létező tranzakcióhoz adja, vagy egy új tranzakciót indít. A tranzakciót az ingres_commit()- vagy az ingres_rollback()-panmcsokkal fejezheted be, illetve az ingres_autocommit()-utasítással elérheted, hogy minden tranzakció automatikusan végrehajtódjék. Az **ingres_autocommit()** argumentumában egy opcionális kapcsolatazonosító szerepel, amelynek az aktuális kapcsolat az alapértéke, és bekapcsolja az autocommit-ot (vagy kikapcsolja, ha az **autocommit** már be volt kapcsolva). Mind az **ingres_commit(),** mind az **ingres_rollback()** elfogad egy opcionális kapcsolatazonosító paramétert, és elvégzik, vagy visszavonják a függőben lévő frissítéseket.

A sorok és oszlopok számát az eredményben rendre az **ingres_num_j*ows()** és az **ingres_num_fields()**-parancsok adják. Azonban a híresztelések alapján az **ingres_num_rows()** bezárja a lekérdezés kurzorát, és eredmények nélkül hagy téged, vagyis vagy alaposan teszteld ezeket az utasításokat, vagy kerüld a használatukat. A sorok eredményben történő megmutatásához egyszerűen illeszd be valamelyik kinyerő utasítást egy while()-ciklusba, mint ahogy azt a következő kód is mutatja. Az **ingres_fetch_row()** hagyományos tömbként nyer vissza egy sort, az **ingres_fetch_array()** egy asszociatív tömbként, az ingres_fetch_object() pedig objektumként. A következő példában **a ingres_fetch_row()-t** használjuk:

```
$sql = "select * from vegetables";

if(ingres_connect("database", "username", "password")) {
    if( ingres_query($sql)) {
        print{"<table border=\"3\">");
        while($row =
        ingres_fetch_row()) {
            print("<tr>"); while(list($k, $v)
            = each($row))

                print("<td> $v
                " }                    "</td>");
        print("</tr>"); }
        print("</table>");
```

```
ingres_close();
}
```

Egy lekérdezés eredményében szereplő mezőkről a következő utasítások valamelyikével kaphatsz információkat. Csak a mező indexét kell megadnod nekik, amely 1-től kezdődik, valamint egy opcionális kapcsolatazonosítót is megadhatsz, ha több kapcsolatot használsz, és megjelenítik az eredményt. Az első ránézésre nagyjából megegyeznek a megfelelő ODBC-parancsokkal - annyira hasonlítanak, hogy nem értem, a Computer Associates miért nem tette ODBC-kompatibihssé az interfészt, mint ahogy az IBM tette a DB2-vel:

```
ingres_field_name()
ingres_field_type()
ingres_field_nullable()
ingres_field_length()
ingres_field_precísion()
ingres_field_scale()
```

Időzítések kinyerése: út a teljesítményhez

Ez a megoldás különböző parancsok, SQL-utasítások és adatbázison végzett változtatások - mint pl. indexek hozzáadása közötti apró eltérések - időzítéséről szól. Ezek sok ezer oldalletöltés esetén jelentős megtakarításokká adódhatnak össze. Ez a megközelítés azt feltételezi, hogy az SQL-utasítások futását nem adatbázis-segédprogramokkal időzíted, mivel ezek csupán a teljes idő egy részét képviselik, vagy pedig hogy az adatbázis-finomhangoló eszközöket arra használod, hogy módosításokat javasoljanak, amelyeket aztán ellenőrizhetsz a szknptedben. Mérheted egy oldal válaszidejét egy böngészőben, ami nagyszerűen használható ugyan a szkriptek finomhangolásának más területein, de semmit sem fog elárulni az egyes adatbázis-hozzáférések részleteiről.

Mindenekelőtt olvass utána a 2. fejezetben leírt microtimeQ-utasításnak; de jelen pillanatban mindössze a következő kódra van szükséged, ami egy Unix-formátumú idősztringet ad vissza, amelyhez hozzáfűz egy néhány (többnyire hat) számjegyből álló mikroszekundum értéket, vagy bármit, amit a szerver biztosít a nagy pontosságú időméréshez. Az első sor a microtime() használatának legegyszerűbb módját mutatja, a második sorban látható a létrehozott sztring, a további sorok pedig azt mutatják, hogyan formázd ezt a sztringet megjelenítés céljára valamilyen olvasható dátum/idő formátumba. Az utolsó sor az eredményt mutatja:

```
$microtime = microtime ()
,-0.52291300 989829487
list($seconds, $míkroséconds) =  explode(" ", $microtime);
$display_time = date("Y-m-d H:i:s      $seconds)
    . substr($microseconds, 1);
2001-05-14 06:38:07.52291300
```

Ezek után az időmérő keretbe kell illeszteni egy adatbázis-utasításokból álló függvényt, lehetőleg valamilyen értelmes módon, a függvény feldarabolását kerülve, mert minden, az

időmérésből fakadó pótlólagos időráfordítás csökkenti az időmérés pontosságát. Az első Gyors megoldások ODBC-példáiból vettem a kódpéldákat, kezdjük a kapcsolódási kóddal:

```
$micro_start = microtime( ) ;
$connection["vitamins"] = odbc_connect("vitamins", "",
"" ) ; $micro„stop = microtime( ) ;
$event[] = array("description" => "odbc_connect
    vitamins", "start" => $micro_start, "end" =>
    $micro_stop);
```

Figyeld meg, hogy a kód egyszerűen elmenti az időt a memóriába az odbc_connect()-utasí-tás előtt és után, majd az időket egy tömbbe teszi. Ez a módszer minimalizálja az adatbázis eseményméréséből fakadó többlet időfelhasználást, és későbbre halasztja az időmérés kezelésének költségeit. A kapcsolódáshoz szükséges idő és az időmérés többletideje közötti arány megállapításához még egy időmérést kell hozzáadni, mint a következő példa mutatja, ami a microtime() többlet időfelhasználását adja meg, és fűzi hozzá a tömbhöz:

```
$micro_start = microtime( ) ;
$connection["vitamins"] = odbc_connect("vitamins", "", "" ) ;
$micro_stop = microtime( ) ;
$event[] = array("description" => "odbc_connect vitamins",
    "start" => $micro_start, "end" => $micro_stop);
$micro_extra = microtime( ) ; $event[] =
arrayf"description" => "microtime overhead ",
    "start" => $micro_stop, "end" => $micro_extra);
```

A következő lépés, hogy egy lekérdezés végrehajtását ágyazzuk időmérő keretbe. Az egyedüli különbség, hogy megadunk még egy időt, a **$query_start** értékét, amelyet a lekérdezés teljes feldolgozásának mérésére használunk. Figyeld meg, hogyan teszi lehetővé a PHP, hogy egy sorban több változónak megadd ugyanazt az értéket. Ennek a sornak a hatására a $query_start értéke az állítás további részének az értékével lesz egyenlő, ami a $micro_start-változóhoz rendelt érték:

```
$sql - "select * from stock"; $query_start =
$micro_start = microtime( ) ; $result =
odbc_exec($connection["vitamins"], $sql); $micro_stop
= microtime( ) ;
$event[] = array("description" => "odbc_exec " .
    $sql, "start" => $micro_start, "end" =>
    $micro_stop) ,-
```

A következő példa a sorvisszanyerő kód beágyazását mutatja. Mivel a visszanyerés egy while()-utasításban található, a ciklus első lefutásához tartozó kezdeti időt közvetlenül a while()-utasítás előtt kell beállítani, a további lefutások kezdeti idejét pedig majd közvetlenül a while()-ciklus végén kell újra beállítani. Annak érdekében, hogy minimalizáljuk a többlet időráfordítást a visszanyerés közben, a kód a visszanyert sorokat későbbi feldolgozás céljára egy tömbbe menti. Ha az adatbázisod alaposan fel van töltve adatokkal és sok hosszú nyitott kurzorod van, akkor csökkentheted a szerver terhelését, ha a kurzor tartalmát azonnal egy tömbbe olvasod, és az **odbc_free_result()**-paranccsal felszabadítod a kurzort, a későbbi feldolgozást pedig a tömbből végzed:

```
if($result)
```

```
$array = "";
$micro_start = microtime( ) ;
while($row = 0,
odbc_fetch_into($result, $array))

    $micro_stop = microtime();
    $event[] = array("description" => "odbc_fetch_into"
    "start" => $micro_start, "end" => $micro_stop); $rows[]
    = $row; $micro_start = microtimef);
```

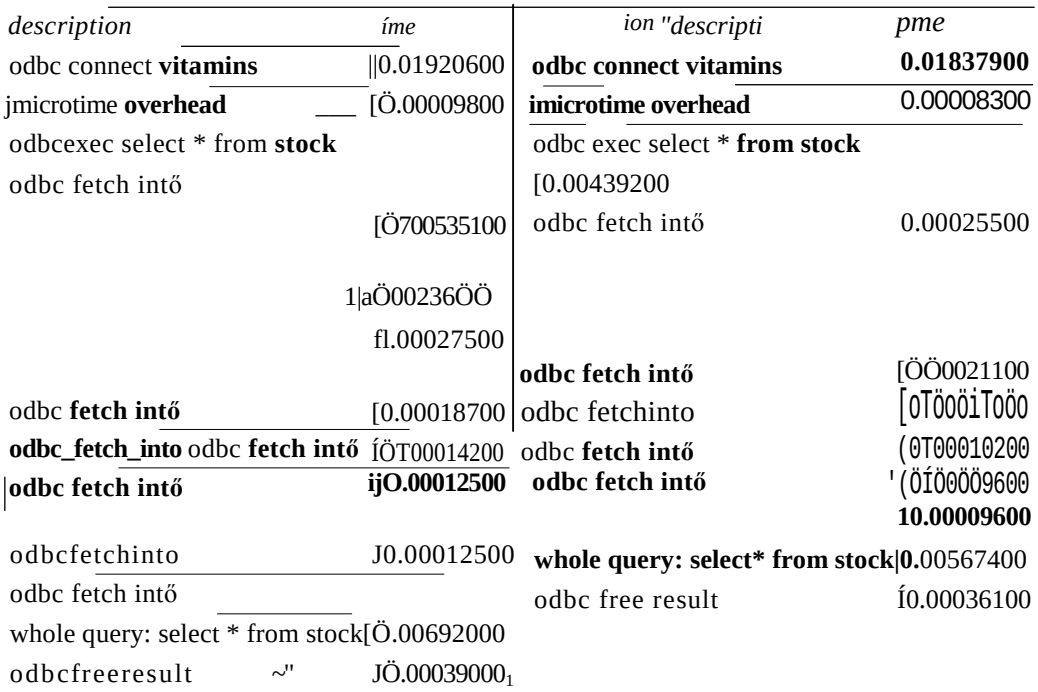
A lekérdezés készen van, tehát a következő kód a teljes lekérdezés idejét rögzíti, majd felszabadítja a kurzort, és rögzíti a kurzort felszabadító utasítás lefutásának idejét. Mivel az erőforrások a szkript feldolgozása után szabadulnak fel, nem biztos, hogy egy rövid szkript végén akarod manuálisan felszabadítani őket. Sokkal valószínűbb, hogy ezt valahol egy hosszú, vagy több lekérdezést tartalmazó szkript elején teszed meg:

```
$micro_stop - microtime();
$event[] - array("description" "whole query: " => $sql,
=> "start" => $query_start, $micro_stop);
"end" $micro_start =
microtime( ) ;
odbc_free_result($result);
$micro_stop = microtime( ) ; "odbc_free_result",
Sevent[] = array("description" $micro_stop);
=> "start" => $micro„start,
"end" ->
```

Miután a lekérdezés befejeződött, megjelenítheted az időmérés eredményeit, vagy későbbi elemzés céljából kiírhatod egy fájlba a merevlemezre. Naplózhatod egy adatbázis elérési időit, de ez lelassítja az adatbázist, és a más oldalak által mért időket is befolyásolja. A következő példakód végiglépked a lementett idők tömbjén, és létrehoz egy könnyen olvasható formázott táblázatot. Figyeld meg, hogy az egyedüli számítást az időkkel a BCMath **bcsub()** utasításával végezzük el, mivel a PHP egész számokra vonatkozó műveletei nem kezelik a nagy számokat, és a lebegőpontos számok műveletei nehezen olvasható jelöléssel adják meg az eredményt. A BCMath - vagy GMP, ha GMP van telepítve - egyszerűen kezeli a nagy számokat. A BCMath benne van a PHP Win32 bináris állományában, a Unix-verziókhoz pedig az - enable-bcmath-paraméterrel kell befordítani a PHP-t:

```
print("<table border=\"3\">"
    . "<tr><td><em>description</em></td><td><em>time</em></tdx/tr>" ) ;
while(list($k, $v) = each($event))
    {
    list($start_microseconds, $start_seconds) =
       explodef" ", $v["start"]);
   $start_time = $start_seconds . substr($start_inicroseconds, 1);
    list($end_microseconds, $end_seconds) = explode(" ", $v["end"]);
    $end_time = $end_seconds . substr($end_microseconds, 1); $time =
    bcsub($end_time, $start_time, 8); print ("<trxtd>" . $v [
    "description" ] . "</td>"  . $time . "</tdx/tr>" ) ;

print("</table>"    );
```

A 6.8. ábra két teszt mérési eredményeit mutatja, tehát a hasonló mérések közötti eltéréseket is megfigyelheted. A kapcsolódási idő 5 százalékot változott a két mérés között, a microtimeQ többlet időráfordítása 18 százalékot változott, az egyes sorok visszanyerése pedig több mmt 100 százalékos eltérést mutat. Vannak olyan változások, amelyeket nyilvánvalóan figyelembe kell venned, és van néhány, amely meglepetést jelent.



6.8 ábra ODBC-visszakeresési idők

A legtöbb időt a kapcsolódás emésztette fel, tehát érdemes volna összehasonlítani egy állandó kapcsolattal. Az állandó kapcsolat nem mindig lehetséges, és néhány adatbázisnál nem takarít meg időt. Ebben a példában az időmegtakarítás nem volna számottevő egy sok lekérdezést tartalmazó oldalon. A legtöbb időt egy olyan tesztoldalon spórolhatnád meg, amelyen minden lehetséges adatbázis-módosítást el lehet végezni.

A teljes lekérdezés egy kicsit gyorsabban futott le a második alkalommal, tehát megérné többször lefuttatni a tesztet, hogy kiderüljön, ez vajon a merevlemez-blokkok memóriába való betöltésének eredménye-e (ami arra utal, hogy az adatbázisszoftver vagy a fájlrendszer jól használja a memóriát) ? Próbáld meg elvégezni a tesztet egy frissítő lekérdezéssel is, hogy lásd, milyen gyorsan működik a lekérdezés. Néhány fájlrendszer, mint az XFS, az NTFS és a Reiser naplózza a frissítéseket, hogy egy esetleges összeomlás esetén helyre tudja állítani a változtatásokat. Ez néhány adatbázist lelassít, pl. a PostgreSQL-t, mivel maga is elvégzi a naplózást, hogy lehetővé tegye a rollback-utasítással a tranzakciók visszavonását. A nagy gyorsítótárat használó és fejlett fájlrendszerek oly mértékben felgyorsíthatják az olvasást, vagy lelassíthatnak bizonyos típusú frissítéseket, hogy fontolóra veheted az adatbázisod áthelyezését egy külön partícióra, egy másik fájlrendszerre.

Az odbc_exec()-utasítást szétvághatod az odbc_prepare()- és odbc_execute()-utasításokra, hogy megvizsgáld, mennyire használható az SQL előkészítő funkciója. Ha az adatbázis egy másik szerveren van, akkor megvizsgálhatod, mennyire konzisztensek a visszanyerési idők, és összehasonlíthatod az adott hálózati kapcsolaton tapasztalt forgalommal.

Hivatkozás:	oldal:
rand(), srand(), és microtime()	62

7. *fejezet*

Környezet

Áttekintés

Érezted valaha, hogy legszívesebben megragadnád a számítógépedet, és jól megráznád, hogy felébredjen végre, és azt tegye, amit szeretnél? E fejezet olyan eszközöket ad a kezedbe, amelyek túlmutatnak a weboldalakon, megkerülik a webszervert, a számítógép mélyére hatolnak, és sebészi pontossággal avatkoznak be a rendszer makacskodó részében. A webszerver által kínált biztonsági megoldásoknál sokkal kifinomultabbak is elérhetővé válnak. Az egyszerű, parancssorból futtatható programokhoz PHP-szkriptből történő meghívással kulturált kezelőfelületet készíthetsz, így a felhasználók egy áttekinthető és felhasználóbarát weboldalt kapnak.

Apache

Az Apache egy nagyszerű webszerver, amelynek lehetőségeit szkriptek segítségével még jobban kihasználhatod. Az Apache nyújtotta különleges lehetőségeket számos módon kiaknázhatod, feltéve, hogy telepítve vannak a rendszereden, ezzel is fokozva a PHP-szkript hatékonyságát. Az Apache ezen túl a különleges változók széles választékát kínálja. Más webszerverek is tartalmaznak hasonló változókat, de nem feltétlenül ilyen széles skálán.

Az első fejezetben példákat láthattunk az apache_lookup_uri()-paranccsal történő URI lekérdezésére, az Apaehe-modulok közti kommunikációra az **apache_note{)**-parancs segítségével, ASCII és EBCDIC közti konverzióra, az összes HTTP-fejléc lekérésére a getallheadersQ-paranccsal, valamint nem PHP alapú oldalak használatára az Apache CGI-interfészén keresztül a virtual()-paranccsal. Az Apache note-rendszere lehetővé teszi, hogy egy szkript futása közben az egyik modul egy másikkal kommunikáljon. Ha azt szeretnéd, hogy egy többszörösen meghívott modul különböző előfordulásai kommunikáljanak egymással, akkor az Apache megosztott memóriáját kell használnod, amit ebben a fejezetben tárgyalunk részletesen.

Mikor egy böngésző lekéri ez egyik oldaladat, egy sor HTTP-fejlécet küld a webszerver-nek, amely értelmezi ezeket, egy részüket továbbadja a PHP-nak, ami futtatja a szkrípted. A szkripten belül a fejlécből származó információk nagy részéhez a környezeti változókon keresztül férhetsz hozzá, az eredeti fejléceket pedig a **getallheaders()**-paranccsal lehet beolvasni. A 7.1 ábra néhány tipikus fejlécet mutat.

Az **Accept** után található a böngésző által fogadott fájlok felsorolása; a */* operátor azt jelzi, hogy a böngésző megpróbál valamit megjeleníteni. A fejléc ugyanezen mezője a PHP **$HTTP_ACCEPT** környezeti változójával érhető el. Ez alapján eldöntheted, hogy egy képet elküldhetsz-e Flash-formátumban, vagy tekintettel a régebbi, illetve a gyengébb, például hordozható eszközökön használt böngészőkre, a JPEG formátumot kell használnod. Sajnos előfordul, hogy a felsorolás pontatlan, és tartalmaz egy "'/"'-operátort, holott a böngésző valójában nem képes bármilyen formátum megjelenítésére. Ezért, ha speciális -például Flash - formátumban jeleníted meg a tartalmat, ugyanakkor egyszerűbb - például

JPEG - formátum is rendelkezésre áll, akkor célszerű az összetettebb tartalom mellé betenni egy, az egyszerűbb formátumra mutató linket, hogy a felhasználó kiválaszthassa, melyik verziót kívánja megtekinteni.

Header	*Vaiue*
Accept	|image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Charset	
Accept-Encoding	gzip
j^ccept-Language	jen
'Authorizaíion	Basic cDV0ZXIfakeZYMAG3
Connection	Keep-Alive
IJHost	[petermoulding. **com**
Referer	"http://petermoulding.coni/phpblackbook/enviroranental/index.html
User-Agent	ÍMozilla/4.77 [en] (WinNT; U) ~

7.1 ábra getallheaders() által visszaadott HTTP-fejrészek

Az **Accept-Encoding gzip** arra utal, hogy a böngésző tömörített fájlokat is fogad, vagyis a PHP gzip-utasítását (lásd 16. fejezet) használva betömörítheted a fájlokat és a weblapokat, ezáltal csökkentve az átviteli időt. Ugyanez a mező a PHP **$HTTP_ACCEPT_ENCODING** környezeti változójával érhető el. A Gzip-tömörítés az Apache-szerver által is meghívható, ezáltal minden kimenő adatot tömörítve kap az erre alkalmas böngésző. Ha az Apache úgy lett konfigurálva, hogy alkalmazza a gzipet, akkor a szkriptjeinkben már nem érdemes másodlagos tömörítést alkalmazni.

Az Accept-Langauge en azt jelenti, a böngészőt úgy állították be, hogy angol nyelvű oldalakat kérjen le, illetve az alapértelmezésben angol nyelvű oldalakat fogad. Az Apache konfigurálásakor be lehet állítani, hogy a szerver a nyelvtől függően különböző oldalakat küldjön. Ez ideális megközelítés abban az esetben, ha a weblapod PHP alapú oldalakat is tartalmaz más oldalakkal keverve. Ha minden oldalad PHP alapú, akkor a nyelvet kezelheted a PHP-n belül is. A nyelv és egyéb nemzetközi kérdések kezelésére vonatkozó különböző megközelítéseket a 12. fejezetben tárgyaljuk. A nyelv a **$HTTP_ACCEPT_LANGUAGE** változón keresztül is elérhető.

Az Authorization-mező a bejelentkezéskor használt biztonsági megoldás típusát jelzi, azonban ezt kis számú felhasználó esetén rendszerint a webszerver kezeli, ezért jelen esetben nem túlzottan érdekes. Amennyiben saját biztonsági megoldás programozására vállalkozol, programod valószínűleg szintén a webszervertől kapja a felhasználói azonosítót és a jelszót, ezért megint csak nem kell ismerned az azonosítás típusát. Vedd észre, hogy módosítottam a hitelesítő sztringet, arra az esetre, ha egy hacker megpróbálná feltörni az oldalt.

A User-Agent-mezŐ vagy a **$HTTP_USER_AGENT**-változó értékéből megtudhatod, milyen böngészővel látogatják éppen az oldaladat, noha ez lehet más program is, például egy kereső motor, amely úgy viselkedik, mintha egy böngésző lenne. Néhányan arra használják az Agent-mezŐt, hogy megállapítsák, a böngésző támogat-e bizonyos opciókat, mint például cookie-k fogadása, azonban megfeledkeznek arról, hogy számos böngészőben ezeket a

beállításokat a felhasználók módosíthatják, ezáltal az Agent-mező szinte teljességgel használhatatlan erre a célra.

A phpinfo()-utasítás megjeleníti az összes PHP-vel és Apache-csal kapcsolatos információt: verziószámokat, konfigurációs beállításokat és az összes, az oldalnak átadott környezeti változót. A 7.2 ábrán láthatunk egy példát. Ha létrehozol egy weblapot, amely tartalmazza a <?php phpinfo()?>-kódot, akkor egy böngészővel meg is nézheted, hogyan működik mindez. A phpinfo()-utasítás közvetlenül a képernyőre ír, azért nincs értelme mutatós képernyő-elrendezést készítened. A megjelenített adatok egy része hasznos a hackerek számára, ezért ne tedd nyilvánossá ezt a képernyőt.

A PHP konfigurálása

A PHP a php.ini-fájlon keresztül konfigurálható, de néha előfordul, hogy bizonyos speciális feladatokat ellátó szkriptek számára módosítanod kell ezeket a beállításokat. A következő részben néhány szkriptből módosítható beállítást ismertetünk.

Bizonyos dolgokat csak a php.ini-ben lehet beállítani, ezeket nem lehet szkriptből felülírni. Mások zárolhatóak, így bár egyes oldalakról módosíthatóak, lehet, hogy nem mindegyikről.

Kiterjesztések

Egyes kiterjesztéseket szkriptből is betölthetsz, így jobban gazdálkodhatsz a szerver erőforrásaival. Képzelj el egy híreket szolgáltató szervert, amely minden hírt egy gyors, olvasásra optimalizált szerverről közvetít. Tegyük fel, hogy írsz egy új szkriptet, amely naponta egyszer kapcsolódik egy Microsoft SQL-szerverhez, és összegyűjti a Microsofttal kapcsolatban megjelent legújabb híreket. Miért tartanád a php_mssql.dll-modult egész nap betöltve, ha naponta csak egy percig használod? A dl()-utasítás révén egy modult a szkriptből is betölthetsz, ahogy az alábbi példa mutatja:

```
dl("php_mssql.dll");
```

PHPVariables

Variable	Value
PHP_SELF	/admin/phpinfo.htrn!
HTTP_GET_VARS['V]	12974
HTTP_COOKIE_VARSfcookie"]	1064996159
HTTP_SERVER_VARS["COMSPEC"]	C \WINNTteystem32\cmd exe
HTTP_$ERVER_VARSrOOCUMENT_ROOT"]	i:/he!pnet/web/root
HTTP.SERVER.VARSfHTTP.ACCEPT[11]]	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png.T
HTTP_SERVER_VARS["HTTP_ACCEPT_CHARSET"]	iso-8859-1.*,utf-8
HTTP_SERVER_VARSrHTTP_ACCEPT_ENCODING"]	gzip
HTTP_SERVER_VARS["HTTP_ACCEPT_LANGUAGE"]	en

7.2 ábra A phpinfoQ által visszaadott képernyő része

A php.ini beállításai

Az ini_get()-utasítással beolvashatod a php.ini egy beállítását, majd ahogyan a következőkben látható, ezt módosíthatod, végül visszaállíthatod a php.ini eredeti beállításait. Ha megváltoztatsz egy beállítást, a változás csak az adott szkriptre vonatkozik, mégpedig a változtatás elvégzésének pillanatától kezdve.

Az ini_get()-utasítás argumentumában a php.ini bármely paramétere szerepelhet, és eredményül a paraméter éppen aktuális érékét adja. Az ini_set()-utasítás argumentumában a paraméter neve és a paraméter új értéke szerepel. Az **ini_alter()** az **ini_set()** egy alternatív neve, tehát régebbi kódokban esetleg találkozhatunk az ini_alter()-utasítással is. Az ini_restore() argumentumában szintén a paraméter neve szerepel, és visszaállítja a php.mi-ben eredetileg szereplő értéket. A következő példában a 0 a hamisat jelenti, az 1 az igazat, vagy azt, hogy az adott opció be van kapcsolva. A példa kikapcsolja a hibák naplózását, megjeleníti a változást, majd visszakapcsolja a hibák naplózását és ismét megjeleníti az eredményt:

```
Ssetting    =    ini_get("log_errors")
; print("<br>"      .    Ssetting);
ini_set("log_errors ",      0 ) ;
Ssetting    =    ini_get{"log_errors")
; print("<br>"      .     Ssetting);
ini_restore("log_errors");
Ssetting    =    ini_get("log_errors")
; print("<br>"      .    Ssetting);
```

A kód egy olyan szerveren volt lefuttatva, ahol a php.ini-ben a hibák naplózása az alapbeállítástól eltérően volt kapcsolva. A következő eredményt kaptuk:

```
1
0
1
```

Időkorlát

Szknptjeid futása egy bizonyos idő után megszakad, hacsak nem állítod az időkorlátot 0-ra, ami lehetővé teszi, hogy a szkriptek tetszőlegesen hosszú ideig fussanak. Minden bizonnyal nem akarod, hogy az éppen egy ciklusban lévő szknptjeid a végtelenségig fussanak, és értékes CPU-időt fogyasszanak, ezért érdemes körültekintően eljárni ezzel a beállítással.

Az alábbi példában látható, hogyan kell beállítani az időkorlátot, ahol a **set_time_limit{)-**utasítással 20 másodperces korlátot határozunk meg:

```
set_time_limit(20) ;
```

Az időkorlátot onnantól kell számítani, amikor a kódban végrehajtódik az ezt beállító sor, tehát ha a szkript ezt megelőzően már 5 másodperce futott, akkor összesen 25 másodpercig futhat. Az időkorlátok nem kumulatívak, azaz nem kapsz 60 másodperces korlátot, ha három egymás után következő sorban 20 másodperces korlátot állítasz be.

Ahhoz, hogy elkerüld a végtelen ciklust, soha ne állítsd be az időkorlátot egy cikluson belül. Képzelj el egy szkriptet, amely létrehoz egy lassú hálózati kapcsolatot, bizonyos műveleteket végez a letöltött adatokon, majd visszatér a ciklus elejére, hogy újabb adatokat szerezzen. Feltéve, hogy egy 10 másodperces korlátot állítunk be a cikluson belül, és a hálózati kapcsolat 9 másodpercet vesz igénybe, valamint 1000 különböző' adatot akarunk letöltetni, a szkript 9000 másodpercig, azaz 2,5 óráig futna. Jobban járunk, ha a ciklus előtt beállítunk egy 100 másodperces tesztkorlátot, és a cikluson belül számoljuk a letöltött adatelemeket, így megállapíthatjuk, meddig jutott a folyamat, mielőtt megszakadt volna.

Környezeti változók

A PHP környezeti változók értékeinek számos forrása van: a PHP, az Apache, a hálózat és a beérkező HTTP-fej lécek. A változókat több különböző módon elérhetjük, köztük a getenvQ-utasítás segítségével, a putenvQ-utasítással pedig értéket adhatunk a változóknak.

A következő példában a getenv()-utasítással kapjuk meg az URL lekérdező sztring értékét. Mindannak, ami a getenv()-utasítás révén elérhető, nagy része előre definiált változóként is hozzáférhető, ha a php.im-ben a track_vars-paraméter be van állítva:

```
$q    =    getenv("QUERY_STRING");
```

A környezeti változókat a putenvQ-utasítással is megadhatod, de ez semmilyen célt nem szolgál az éppen aktuális szkript számára, a következő szkript számára pedig minden visszaállításra kerül. Mire való akkor a putenv()-utasítás? Mikor egy külső programot hívsz meg az execQ- vagy a systemQ-utasításokkal, ezek a programok olvashatják a környezeti változókat, így ezeket a változókat egy meghatározott értékkel adhatod át a külső programoknak.

Biztonság

A munkád minden bizonnyal nagy fontosságú és értékes adataid vannak. Ezért nem szeretnénk, hogy egy utálatos kis 14 éves kölyök betörjön a rendszeredbe és beleirkáljon a fájljaidba. Az alábbiakban megtudhatod, hogyan lehet a fájljaidtól távoltartani a besurranókat, a csalókat és a nyíltan támadó kétbalkezeseket.

HTTP-hitelesítés

A HTTP-hitelesítés akkor érhető el, ha a PHP-t Apache-modulként futtatod, nem pedig CGI-ként. Azok, akik a PHP-t CGI-ként használják, csak az Apache .htaccess- és .htpasswd-fájljait használhatják biztonsági célokra. Tehát az 1. fejezetben említett támadási terv azzal kezdődik, hogy elérjük, hogy a PHP csak a kezdeti tesztelés céljára fusson CGI-ként, majd áttérünk egy Apache-modulra, hogy a haladóbb lehetőségeket is kihasználhassuk kereskedelmi célú weboldalainkon. Az Apache biztonsági eszközeit megtarthatjuk arra a célra, hogy egy adminisztrációs könyvtárat elzárjunk a jogosulatlan felhasználók elől, és a felhasználók egyéni bejelentkezését kezelhetjük PHP HTTP hitelesítésével.

Megjegyzés: Ha további információt szeretnél a HTTP-ről és a fejlécekről, olvasd el az alábbi dokumentumot: www.w3.org/Protocols/rfc2616/rfc2616.

A hitelesítés a header()-utasítással kezdődik, amely egy "Authentication Required" (Hitelesítés Szükséges), majd egy 40l-es fejlécet küld el, amelynek hatására a böngészőben előugrik egy felhasználói nevet és jelszót kérő ablak. A header()-utasítás egy tartományt (reatm) igényel, amely egy, a hitelesítéshez kapcsolt név. Ez azt jelenti, hogy különböző oldalakon különböző bejelentkezéseket használhatsz. A hitelesítés során a tartomány is megjelenik a bejelentkező képernyőn. A felhasználói név a **$PHP_AUTH_USER-,** a jelszó pedig a $PHP_AUTH_PW-változóba kerül.

CHMOD

A 8. fejezetben részletesen ismertetett **chmodQ-, chgrpQ-** és **chown()**-utasítások lehetővé teszik, hogy egyéni és csoportos jogosultságokat biztosíts, valamint meghatározd egy fájl tulajdonosát. Az utasítások Windows vagy NT alatt nem, csak Unix alatt működnek, ahol beállíthatsz új felhasználót, saját könyvtárral és kizárólagos hozzáféréssel. Más operációs rendszerek alatt egy külső program láthatja el a biztonsági feladatokat, vagy a felhasználói adatokat egy adatbázisban tárolva az adatbázis biztonsági megoldásaira hagyatkozva szabályozhatod a hozzáféréseket. A PHP azokhoz a fájlokhoz módosíthatja a jogosultságokat, amelyek a sajátjai, vagy ahol a PHP egy olyan csoport tagja, amely megváltoztathatja az adott fájl hozzáférési jogait.

PHP kötegelt módban

A weboldalak nem lebecsülendők, de miért korlátoznád hozzáértésed és tudásod weboldalakra? Sőt, miért kényszerítenéd magad egy speciális nyelv felidézésére, ha mindössze batch-programokat szeretnél írni, és a PHP-t mindenre használhatod?

A következő néhány trükk révén a PHP-t nemcsak böngészőből hasznosíthatod, hanem egy parancssor-konzolból is.

Az alábbi kód Windows NT parancssor-konzoljából futtatható, és egy PHP-oldal lefutását eredményezi. Az eredmény a 7.3 ábrán látható.

```
"c:/program files/php/php" i:/petermoulding/web/root/test/test.php
```

Ezt a PHP bármely telepítésére alkalmazhatod, mindössze arra van szükség, hogy a parancs a PHP CGI-ként futtatható részére mutasson. Ha a PHP-d csak egy modulként van lefordítva, a telepítési útmutatások alapján fordíthatod le CGI-ként. A Win32-es telepítő alapbeállításként mind a modul-, mind a CGI-verziót tartalmazza, tehát nincs szükség fordításra.

A 7.3 ábrán látható fejléc első három sora elhagyható, ha egy -q-operátort adunk a parancshoz, mint a következő sor mutatja:

```
"c:/program files/php/php" -q
i:/petermoulding/web/root/test/test.php
```

7.3 ábra PHP-oldal parancssorból futtatva

A parancsként futtatható oldalakból az összes HTML-t érdemes elhagyni, egyedüli hasznos formázás az **újsor-(newline-)** karakter, a \n. A PHP-lap futtatás eredményének utolsó sori egy hibát mutat, ahol a kód egy általában az Apache által létrehozott környezeti változót próbál felhasználni, vagyis elhagyhatjuk az összes, Apache-hez kapcsolódó kódot is.

Hogyan lehet adatokat felhasználni a parancsban? Mindent fájlokban adok meg, amelyeket egy weboldalról lehet frissíteni, és amelyeket a parancs a szokásos fájlműveletekkel olvas. Ha Unixot használsz, próbálkozz a readline()-paranccsal, amely a GNU readline-projekt-jétől függ **(cnswww.cns.cwru.edu/~chet/readline/rltop.html).**

A readline()-parancshoz egy sor kiegészítő parancs tartozik, mint **pl.: readline_add_history(), readline_clear_history(), readline_completion_function(), readline_info(), readline_list_history(), readline_read_history()** és **readline_write_history().**

Ezeknek a parancsoknak az ismertetése megtalálható a PHP dokumentációjában, ami letölthető a PHP-val.

A PHP ütemezése

Ha egy PHP-oldalad már parancssorból futtatható, akkor ütemezheted a feladatot, hogy óránként egyszer, egész éjszaka vagy tetszőleges időben kerüljön végrehajtásra. Ha nem szeretnéd hajnali háromkor végignézni a feladat lefutását, helyezd az összes input információt fájlokba, így nem kell begépelni a parancssorba, továbbá az outputokat szintén irányítsd fájlokba, így azok nem vesznek el a parancssorablakban.

Ha egy olyan feladatot akarsz futtatni, amely bemeneti adatokat gyűjt, mondjuk fájlok egy csoportját, akkor készíthetsz egy A könyvtárat a fájlok fogdására valamint egy B háttérkönyvtárat és a **cron-** vagy más ütemező program segítségével beállíthatod, hogy megfelelő időközönként lefusson a feladat. A program az eredeti fájlokat az A könyvtárban találja, míg a feldolgozott fájlokat a B könyvtárban tárolja. Mikor a feladat lefut, csak az új fájlokat veszi fel az A-ba, tehát tetszőlegesen gyakran futtathatod a feladatot. Ahhoz, hogy minden futásról használható és pontos feljegyzésekkel rendelkezz, rögzítened kell a dátumokat, továbbá azt, hogy az adatok hányszor kerültek beolvasásra, és még egy halom információt, és ezeket könnyen olvashatóvá kell tenned.

Azt javaslom, hogy mindent MySQL-táblákban tárolj az 5. fejezetben bemutatott módon, így a naplóállományokat és az eredményeket kényelmesen megnézheted otthonról egy

1

II.
EIM í

weblapon keresztül. Érdemes úgy megírni a programot, hogy hiba esetén e-mailben értesítsen, így otthonról is beavatkozhatsz és kijavíthatod a hibát. Ha igazi szabadságra vágysz, a lefutást befolyásoló paraméterek beállítását megoldhatod egy biztonságos weblapon keresztül, vagyis hajnali háromkor kellemes társaságban kávét kortyolgatva egy menő Internet-kávézóból átkonfigurálhatod a feladatot.

COM

Mindenkinek vannak kedvenc programjai (én imádom az Excelt), és egyes programokat mindenki gyakrabban használ, mint másokat. Mivel az emberek 90 százaléka Windows alapú számítógépeken dolgozik, ezeknek a kedvenc programoknak és az általuk előállított dokumentumoknak a 90 százaléka szintén Windows alapú. Ezért van szükséged a Microsoft Component Object Model (COM) technológiájára. A COM lehetővé teszi, hogy tetszőleges célra felhasználd a programok e 90 százalékát és a hozzájuk kapcsolódó adatokat, beleértve az adatok zárt, saját tulajdonú formátumból nyitott, platformfüggetlen formátumba történő konvertálását. A fejezetben található példában Worddel megnyitunk egy Microsoft Word-dokumentumot, majd RTF-formátumban elmentjük, hogy a Unix-felhasználók számára is hozzáférhetővé váljon.

A COM mint interfész a Microsoft régebbi Dynamic Data Exchange-jének (DDE) utódja, és az ODBC mellett egy másik divatos technológia, de nem olyan tiszta, mint az ODBC. A Microsoft számos COM-variációt kidolgozott, mint pl. az OLE (object linking and embedding) és az ActiveX, ami a COM-fejlesztéseket elég zavarossá teszi a tiszta, átlátható ODBC-vel szemben. (Az ODBC az adatbázisoknál használatos és a 6. fejezetben tárgyaljuk.) Az Gyors megoldások közt, a „Word dokumentumok RTF-formátumba konvertálása COM segítségével" címen egy egyszerű példát találsz a COM alkalmazására, és ennek alapján hasonló COM alapú alkalmazásokat készíthetsz. Ehhez azonban teljes és naprakész dokumentációra lesz szükséged a célalkalmazás objektumainak eljárásairól és tulajdonságairól, vagy pedig végtelen türelmet igénylő tesztelésre.

Régebben dolgoztam Microsoft Access-szel, Excellel és Worddel, mindhármat más programokból vezérelve, először DDE-, majd COM-technológiával, és úgy találtam, hogy e három alkalmazás esetében ez egy egyszerű, könnyen használható megközelítés. Azonban más alkalmazásokhoz nehézséget okozhat megbízható és friss dokumentáció beszerzése és azoknak az apró különbségeknek a kifürkészése, hogy mit tehetsz meg az alkalmazásban közvetlenül a billentyűzettel, és mi engedélyezett egy COM-interfészen keresztül.

Egy dologra kell odafigyelni, ha régebbi COM alapú alkalmazásokkal dolgozol: sok alkalmazás készítői nem gondoltak arra, hogy a program valaha szerver módban fog futni (magára hagyva, anélkül, hogy bárki reagálna a hibaüzeneteire) Ha Windowsban programozod ezeket az alkalmazásokat COM-technológiát használva, akkor látod a képernyőn az előugró hibaüzeneteket, és ennek alapján kijavíthatod a COM-kódot. Mikor ezek az alkalmazások egy Windows NT-webszerver szolgáltatásaként futnak, nem tudják hová küldeni a hibaüzeneteiket. Könnyen úgy járhatsz, hogy az alkalmazás érthetetlen módon leáll, mert szeretne megjeleníteni egy előugró hibaüzenet ablakot, de ezt nem teheti meg.

könnyebb átnevezni a végső felhasználás előtt, és ha a fájlnevek számodra mé
értelemmel bírnak, készíthetsz róluk egy másolatot, amit átnevezel, és azt kül
használónak.

A rövid nevek könnyen kezelhetők, és minden platform fel tudja dolgozni ők
könnyen lehetnek félreérthető'en rejtélyesek. Képzeld el, hogy beküldesz egy
talmazó fájlt a következő névvel: „Kárigény bejelentés 2002. január 6. -ával vé
re". Az "sO 1062002" vagy valami hasonló könnyen begépelhető nevet használ
amely nem tartalmaz olyan karaktereket, amelyek összezavarják az operációs r
Az a probléma, hogy "s01062002" a világ jelentős részén 2002. június 1. -jét je
dátumokkal dolgozol, válassz egyértelmű formátumot, mint pl. "s20010106",
ÉÉÉÉHHNN-formátumot mindenki felismeri, és senki nem használ ÉÉÉÉN
azt szeretnéd, hogy a PHP hasonló fájlnevet produkáljon a mai dátummal, ak
kozz az alábbi kóddal, és ha többet szeretnél tudni a dátumokról, lapozd fel a

```
$file_name    =          date("Ymd"
"s'
```

Hagyd ki a szóközöket, és ne használj aláhúzást helyettesítésre. Noha a Kári
bejelentésből adódik a Kárigényjbejelentés, bizonyos helyzetekben az aláhú
például, ha valaki egy linkről vagy aláhúzással kiemelt fejlécről másolja le a fáj
Kárigénybejelentés könnyen olvasható, de néhány operációs rendszer még m
fogja megtalálni a fájlt, ha valaki kárigénybejelentés-re keres; tehát maradjunk
zileg is biztonságos kárigénybejelentés (karigenybejelentes '.Ékezet nélkül!)
A kötőjel (-) és pont (.) karakterek biztonságosak minden általam használt op
szerben, ám még nem használtam BeOS-t és néhány hordozható eszközt.

Linux és Unix

A Unix-szal az a legnagyobb gond, hogy képtelen felismerni a kis- és nagybet
csenek megfelelő eszközei a kis- és nagybetűk megkülönböztetése nélküli fájl
Úgy tűnik, a fejlesztők imádják a kis- és nagybetűket vegyesen tartalmazó ne

hordozható médiát, mint pl. Zip-lemezt Unix-formátumra megformázni.

POSIX

A Portable Operating System Interface (POSIX) a legmegbízhatóbb megoldást ki
különböző platformokon futtatott alkalmazások és különböző operációs rendsze
zötti kommunikációra. Noha nem minden operációs rendszer támogatja a POSIX
nem mindegyik rendelkezik a POSIX funkciói által igényelt eszközökkel, mégis
funkciók nagyobb eséllyel működnek együtt egy új operációs rendszerrel, mint ha
ósrendszer-specifikus parancsokat használnánk, például a PHP külső program int
keresztül. A POSIX úgy is felfogható, mint az operációs rendszerek közötti kapc
ODBC-je (ha még nem használtál ODBC-t, lásd a 6. fejezetet).

Minden Unix alatt futó program egy folyamat és az Apache alatt minden weblap
folyamatnak van egy szülő folyamata. Ha egy szoftver elszáll és megöli a folyamat
fő Apache-folyamatot ez nem zavarja; szakadatlanul új folyamatokat indít az új we
számára. A posix_getpid()-utasítással hozzáférhetsz egy folyamat azonosítójához
beavatkozhatsz egy folyamat életébe, a posix_kill()-utasítással pl. megölhetsz egy
tot. Az elvetemültebbek használhatják a posix_getppid()-parancsot, amely megsz
szülő folyamat azonosítóját és azonnal megöli.

A biztonságot a felhasználói azonosítók és csoportok ellenőrzik, és a
posíx_getuid()-utasí-tással megkapod a folyamatod által használt valódi felhaszná
azonosító numerikus azonosítóját. A posix_getpwuid()-parancs argumentumában
numerikus felhasználói azonosító szerepel, és egy tömböt ad eredményül, amely a
felhasználóval kapcsolatos információkat tartalmazza, mint pl. a neve vagy a titko
jelszava. A folyamatokhoz hozzárendelhetsz egy effektív felhasználói azonosítót;
posix_geteuid()-utasítás az effektív felhasználói azonosítóhoz tartozó numerikus
adja meg, és a posix_setuid() az effektív azonosítót állítja be. Az Apache alatt a
rendszerint nobody felhasználói azonosítóval fut, és semmilyen jogosultsággal ne
rendelkezik. Mikor valaki bejelentkezik, jogosultsá-

gokat kap az Apache biztonsági rendszerétől, és ekkor használhatóvá válnak ezek az utasítások. A posix_getlogin() a felhasználó login nevét adja meg és a posix_getpwnam() a név alapján szolgáltat információkat egy felhasználóról.

A posix_getgid()- és posix_getegid()-függvények egy folyamat csoportját adják meg, a posix_setgid() a csoport azonosítóját állítja be, a posix_getgroups() pedig az aktuális folyamathoz hozzárendelt csoportokat sorolja fel. Az erőforrások területén a posix_times() az eddig elhasznált időt, a posix_uname() a rendszer nevét, a posix_getrlimit() pedig a rendszer számára rendelkezésre álló erőforrások felsorolását adja. Néhány további POSIX-utasítás leírása a www.php.net/manual/en/ref.posix.php címen található meg.

Ha Windows NT-t használsz és szeretnéd kiterjeszteni a POSIX-szal való együttműködési készségét, akkor a RedHatnek van egy Cygwin nevű terméke számodra (http://sources.redhat.com/cygwin). A PostgreSQL a Cygwin segítségével ülteti át a Unixos kódját Windows NT-re. A PostgreSQL-fejlesztők a Cygwint használják arra, hogy Windows NT-n felállítsanak és karban tartsanak egy PostgreSQL-szervert. A Cygwin használatával elvileg számtalan Unix-kódot vihetsz át Windows NT-re. Azonban számos terméknek problémát jelenthet a Windows-interfész és számos Windows-felhasználónak a Cygwin/Unix-interfész, azért nem kell a Unix teljes grafikus kezelői felületét átpakolni Windows-ra. A PHP szolgálhat felhasználói felületként!

Gondoljunk bele a lehetőségbe! Vegyünk egy Unix alapú programot, ami valami különlegeset tud, cseréljük le a felhasználói felületet PHP-ra és HTML-formokra és adjuk közre a programot Windows-on, PHP-t és Apache-ot használva működési környezetként. A Cygwin lehetővé teszi, hogy lefordítsuk a programot Windows alá, és a PHP külső fájlokat kezelő funkciói lehetővé teszik, hogy PHP-ből futtassuk a fájlt. Az Apache Windows-os 1.3.20 verziója előre felinstallált Cygwin-interfészt foglal magában, és alig várom hogy teszteljem és kiderítsem, mennyire használható ez a kombináció.

A programok kommunikációjának elősegítése

Mit teszel, ha szeretnéd közölni minden weboldalaidon futó programmal az andorrai frank - malawi kwacha keresztárfolyamot? Egy fájlt használnál, adatbázist vagy valami gyorsabbat? A megosztott memória a leggyorsabb megoldás, és ha szinkronizálni kell a frissítést, akkor használhatsz szemaforokat.

A kis fájlokhoz való hozzáférés rendszerint gyors, és ha minden alkalmazás ugyanazt a fájlt szeretné olvasni, akkor az operációs rendszer eltárolja a memóriában, ahonnan a hozzáférés gyakorlatilag azonnali. A fájlrendszerek akkor lassulnak le, ha a fájl frissítésére kerül sor; valójában bármely fájl bármely frissítése minden fájl minden hozzáférését lassítja. Ezért ha az oldalunkon sok fájl gyakran frissül, akkor nem a legjobb választás egy fájlban eltárolni az árfolyamot.

Az árfolyam-adatbázis jól hangzik, de az adatbázisok időigényesek, és nem minden adatbázis képes cache-ben tárolni a gyakran lekért információt. A MySQL jó olvasási hozzáférési

idővel és különleges adattáblákkal rendelkezik, de ha nem használjuk ki maximálisan az adatbázis lehetőségeit, akkor még mindig sok CPU-időt pazarolunk el.

Megosztott memória

A megosztott memória növeli a rendszer teljesítményét és alkalmazkodóképességét, ezen túl rendkívül jól mutat a szakterületek felsorolásában az önéletrajzban az egyenlő részek biztonsági rései (equal parts secunty holes), a megbízhatósági problémák (rehabihty problems) és a bővítéskorlátozások (expansion limitations) mellett. Az alábbiakban röviden körvonalazzuk, hogyan skálázzunk Mount Everest-nyi számítást anélkül, hogy megcsúsznánk a jégen.

A megosztott memória a Unix-rendszerek V-jéből származik, tehát alaposan tesztelt; de a megosztott memóriához való korlátlan hozzáférés azt jelenti, hogy bármely szkripted tudja olvasni és módosítani az adatokat. A megosztott memória C-, Perl- és más programok számára is elérhető, úgy, hogy nem csak frissíthetik az adatokat, de véletlenszerűen át is strukturálhatják oly módon, hogy a szkripted nem fogja érteni. A PHP 4.0.3 feletti verzióiban a megosztottmemória-utasításokat az shp_ -tag előzte meg, a 4.0.5-ös verzió óta pedig az shmop_ -előtag.

Az shmop_open()-utasítás argumentumában egy kulcs, egy hozzáférés/létrehozás indikátor, egy oktális hozzáférési mód (ugyanaz, mint chmod használata esetén) és egy méret szerepel, míg visszatérési értéke a megosztott memóriaszegmens azonosítója, ami pont olyan, mint egy fájlazonosító. A hozzáférés kulcsa egy 8 bájt méretű egész szám, így ez lehet véletlenszerűen generált, vagy a megosztott memória valamely más területén tárolt kulcsszámláló eggyel megnövelt értéke. A hozzáférés/létrehozás indikátor értéke c, amikor létre akarod hozni a megosztott memóriát, és a, mikor korábban létrehozott memóriához szeretnél hozzáférni. A hozzáférés módja leggyakrabban 0755 vagy valami hasonló, ha létrehozzuk, és 0, mikor használjuk a már létrehozott memóriát. A méret 0, mikor hozzáférünk a megosztott memóriához, és valahol 1 és 131,072 bájt közt van, mikor létrehozzuk.

A shmop_close() argumentumában a shmop_open() által eredményül adott megosztott memóriaszegmens-azonosító szerepel és lezárja de nem törli a szegmenst. Ha végül szeretnéd törölni a szegmenst, akkor meg kell nyitnod, és a shmop_delete()-parancsot kell kiadnod, majd bezárnod. A shmop_delete() argumentumában a szegmens azonosítója szerepel, és megjelöli a szegmenst mint törölhető memóriát. Ettől a ponttól kezdve semmilyen más folyamat nem tudja megnyitni a szegmenst. Maga a törlés akkor kerül végrehajtásra, ha minden felhasználó lezárta a hozzáférését. A szegmens örökké a memóriában maradna, ha nem törölnénk ki, vagy valaki nem zárná le a hozzáférését. A szegmenst csak a tulajdonosa vagy egy root jogokkal rendelkező felhasználó tudja kitörölni. Az árfolyam példájánál maradva, egy felhasználó hozná létre, frissítené és törölné a megosztottmemória-szegmenst, az összes többi felhasználó olvasná.

Mit tehetsz egy megnyitott szegmenssel? Megvizsgálhatod a méretét a shmop_síze()-utasí-tással, írhatsz bele a shmop_write()-tal és olvashatsz belőle a shmop_read()-del. A shmop_read()-utasításnak a szegmens azonosítójára van szüksége, egy nullától számított kezdőértékre, és az olvasásra kerülő tartomány hosszára (0 mindent elolvas és a

shmop_size()-zal megtudhatjuk a méretét.) A shmop_write()-nak a szegmens azonosítóját kell megadni, a beírandó sztringet, és egy viszonyítási pontot, ahonnan elkezd írni.

Most már tehát birtokában vagyunk a fájlok megfelelőinek a memóriában, amelyek megoszthatók, megvédhetők a kitörléstől és a felülírástól, de nem lehet megbízhatóan frissíteni őket. A megosztott memória mérete, a szegmensek száma és az egy folyamat által megnyitott szegmensek száma korlátozott, ezért körültekintően kell megtervezni és használni a memóriában található adattáblákat, mint pl. amit a MySQL-ben használhatunk, nehogy elárasszuk a megosztott memóriát. A többszörös frissítés védelmét a szemaforok biztosítják, ezt a következő bekezdésben tárgyaljuk.

Szemaforok

A szemaforok révén ellenőrizhetjük a megosztott memória szegmenseinek frissítését. Mielőtt frissítésre megnyitsz egy szegmenst, a sem_get()-utasítással kérhetsz egy szemafor azonosítót, amely a szegmens azonosítóján alapul, majd a frissítés elvégezése után a sem_release()-utasítással engedheted el a változót. A sem_get() argumentumában először meghatározhatod a kulcsot, majd egy l-es használati korlát következik (tehát a folyamatod lesz az egyetlen, amely hozzáfér megfelelő szegmenshez), végül a chmod-ból ismert oktális mód, ami megakadályoz másokat a szegmens frissítésében. Ha más folyamatok birtokában vannak a szemafornak, add ki a sem_aquire()-utasítást> amely meggátolja az új folyamatokat abban, hogy hozzáférjenek a szemaforhoz, majd várjuk meg, míg a létező folyamatok elengedik a szemafoft.

Külső programok

És most felnyitjuk Pandora szelencéjét, hogy a benne található titkos rendszerparancsokkal tönkretegyük a webszerverünket, vagy esetleg mindenkit bámulatba ejtsünk tökéletes programozói tudásunkkal. A következő rövid részben néhány egyszerű lépést végigkövetve temérdek új paranccsal ismerkedhetünk meg, és arra nézve is útmutatást kapunk, hogyan válasszuk ki közülük, melyik a megfelelő és megbízható.

Az exec()-utasítás argumentumában egy szöveges parancs szerepel - igen hasonló ahhoz, amit a parancssorba gépelünk be -, valamint egy opcionális tömbnév és egy opcionális változó, amelyben az eredmények eltárolhatok. Miután a parancs végrehajtása megtörtént, visszatérési értékül a parancs outputjának utolsó sorát adja. Ha a megszokott visszatérési státust szeretnéd, akkor harmadik paraméterként meg kell adnod még egy változót, és ezt a változót kell ellenőrizni (ez az utasítás nyilvánvalóan nem illeszkedik a standard PHP 4 utasítások formájához). Ha a teljes outputra szükséged van, akkor ezt a második paraméterként megadott tömb révén érheted el. A „Külső programok futtatása" című Gyors megoldásokban megtalálható passthru()- és system()-parancsok az exec() alternatív nevei.

Az escapeshellarg()-parancs az argumentumában szereplő sztring köré félidézőjelet tesz, valamint a sztnngben szerelő félidézőjelekhez és speciális karakterekhez is félidézőjelet vagy visszaperjelet ad. Mint a következőkben látható, ez lehetővé teszi az exec()-hez hasonló utasításokban a paraméterként használt sztnng formázását. Ha megengeded a felhasználók-

nak, hogy nyers adatokat gépeljenek be, és ezeket táplálod be paraméterként a parancsok számára, akkor a felhasználók utasításokat tartalmazó adatokkal behatolhatnak a rendszerbe. Az **escapeshellarg()** biztosítja, hogy a sztring egy egyedülálló sztringként kerül feldolgozásra, ezáltal csökkenti a behatolás veszélyét. A következő példában saját magunk adhatunk idézőjeleket és perjeleket a sztringhez, de a valóságban a sztring egy adatbázisból vagy valamely más forrásból származhat, ahol nem tudjuk megváltoztatni az adatokat.

```
$parm = " -Tt /dev/hda" ; "^
exec("hdparm " . escapeshellarg($parm));
```

Az **escapeshellargQ** által visszaadott sztring:

```
[1]-Tt /dev/hda'
```

Az **escapeshellcmd()** az **escapeshellarg()** egy korábbi változata, de nyilvánvalóan nem működik olyan jól. Nem hasonlítottam őket össze forráskód szinten, és úgy készítettem el weboldalaimat, hogy a felhasználók ne tudjanak sztringeket begépelni közvetlenül parancsok paramétereként.

Extra információ

A PHP szüntelenül újabb hasznos információkat bocsát rendelkezésünkre. Ha egy képernyőn egy erőforrás azonosítóját sztringként használtad, akkor korábban néha egy sima egész számot láttál, pl. 3, ami nem segített a képernyőt előállító utasítás hibájának felismerésében. Most az üzenetet a **Resource id#3** formában kapjuk, így azonnal észrevehetjük, hogy a select eredménye helyett a forrás azonosítóját jelenítettük meg (ahelyett, hogy kinyertünk volna egy sort, és az került volna megjelenítésre):

```
$result = mysql_query("select * from
countries"); print($result);
```

A JPEG-fájlok tartalmazhatnak olyan mezőket, amelyek alapján megállapítható, milyen beállítások voltak érvényben a kamerán, amellyel rögzítették a digitális képet. A PHP-ban most már szerepel egy utasítás, a **getexifQ,** amely megjeleníti ezeket az információkat. A JPEG-ekről szóló és a JPEG-ekből származó információk megjelenítéséhez lapozzuk fel a 8. és a 11. fejezeteket.

mnoGoSearch-függvények

Az mnoGoSearch-oldalról **(www.mnogosearch.ru)** letölthető mnoGoSearch-szoftver Unix-platformon ingyenesen használható a Win32 bináris verzió viszont pénzbe kerül, ami korlátozza a program elterjedését. Az indexelésre alkalmas szoftver információkat gyűjt a fájlokról, oldalakról, honlapokról, tetszőleges ODBC adatbázisban tárolja őket, és egy PHP-mterfészt biztosít az adatbázisban való keresésre. Ezek a funkciók a mi honlapunkon is hasznosak lehetnek, feltéve hogy a weblap Unix-platformon működik és a jövőben is azon fog működni. A következő felsorolás az egyes utasítások rövid leírását tartalmazza. (A szoftver egyelőre nincs feltelepítve a saját weblapomra, ezért itt példákat nem közlök.

Az **mnoGoSearch** weboldalán tesztelhető a keresés.)

- udm_alloc_agent() Egy mnoGoSearch-sessiont indít.
- **udm_free_agent()** Lezárja a sessiont.
- **udm_add_search_limit()** Korlátozásokat ad a kereséshez, így az nem fut a végtelen ségig. ^
- **udm_clear_search_limits()** Eltávolítja a keresési korlátozásokat.
- **udm_find()** Végrehajtja a keresést.
- **udm_get_res_field()** A keresési eredményeket adja.
- **udm_free_res()** Felszabadítja a kereséshez rendelt erőforrásokat.
- **udm_cat_path()** A legfelső kategóriától az éppen aktuális kategóriára mutató útvona lat adja. (A kategóriákat úgy kell elképzelni, mint a **pl.** a **Yahoo.com** kategóriáit.)
- **udm_cat_list()** Felsorolja az aktuális kategóriával egy szinten lévő kategóriákat.
- **udm_api_version()** Megjeleníti az kereső verzióját, így megbizonyosodhatunk, hogy a legfrissebb verziót használjuk.
- **udm_errno()** Visszatérési értéke egy hibakód.
- **udm_error()** Visszatérési értéke egy hibaüzenet, így meghatározhatjuk a problémát.
- **udm_get_doc_count()** Megadja az adatbázisban található dokumentumok számát, így megbizonyosodhatunk arról, hogy a dokumentumok száma összhangban van az **indexelni** kívánt honlappal.
- **udm_load_ispell_data()** Betölti **a** nyelvi ellenőrző adatokat (ispell).
- **udm_free_ispell_data()** Eltávolítja a nyelvi ellenőrző adatokat.
- **udm_set_agent_param()** Beállítja a keresőügynök paramétereit.
- **udm_get_res_param()** Visszatérési értékei az eredmény paraméterei.

Gyors megoldásokok

A régi fájlok eltávolítása

Szeretnél egészséges környezetet biztosítani a honlapod számára, ezért ki kell takarítanod az elavult fájlokat. Az operációs rendszerekben is fellelhetők bizonyos eszközök és módszerek a fájlok kirostálására, de ezek nem kifejezetten a becses honlapod fájljaira vonatkoznak. Egyszer elérkezel arra a szintre, mikor egyedi kitisztító alkalmazásra lesz szükséged. Ez az igazán elegáns megoldás, amelyet olyan tetszetős kezelőfelülettel ruházhatsz fel, amilyennel csak szeretnél.

A kód a fájlok egy tömbben való felsorolásával kezdődik, majd kitörli a fájlokat. Az 5. fejezetet elolvasva felépítheted a listát tartalmazó adatbázist, és létrehozhatsz egy csinos formot a lista karbantartására. Az adatbázis a felhasználók hozzáférését is kezelheti, vagyis más honlapépítők is megadhatják a törlési igényeiket. Sőt, a törlési dátumot is meghatározhatjuk, így a fájlok eltávolítása már a hozzáadáskor meghatározható.

A kitörlendő fájlok felsorolásával kezdjük. A következő példákban először egy tesztfájlt és egy tesztkönyvtárat nevezünk meg egy ideiglenes könyvtárban, majd a következő, sokkal találóbb példában egy dátumhoz kötődő reklámkép és -banner fájljait tekintjük át. Mikor felrakjuk az online-boltunk apák napi akciójához kapcsolódó képeket, meghatározhatjuk, hogy apák napja után az összes apák napi zoknit, pólót és papucsot ábrázoló kép eltávolításra kerüljön a szerverről:

```
$delete[] = "t:/Copy of New Text
Document.txt"; $delete[] = "t:/Copy of temp";
```

A kód számos üzenetet jelenít meg azonos formátumban a ptr()-utasítás révén, amelynek argumentumában két érték szerepel, és azokat egy HTML-táblázat sorában helyezi el:

```
function ptrí$v, $m) { print ("<trxtd>" . $v .
    "</tdxtd>" . $m . "</tdx/tr>") ;
```

A következő kód beolvassa a kitörlendő listát, fájlokat töröl és könyvtárakat olvas be, hogy újabb kitörlendő fájlokat és alkönyvtárakat találjon. Egy alkönyvtárakat tartalmazó könyvtárat nem könnyű kitörölni, ezért a kód minden egyes alkönyvtárt egyedileg töröl. Mikor a kód megtalál egy alkönyvtárát, hozzáadja a kitörlendő lista végéhez. Ez az eljárás a főkönyvtártól az alkönyvtárak felé tartó listát állít elő. A kód egy következő része megfordítja a felsorolás sorrendjét, és a legalsó alkönyvtárral kezdve a megadott főkönyvtárig bezárólag mindent kitöröl.

A kód első sora visszaállítja a tömbváltozót, arra az esetre, ha a mutató nem a tömb első elemére mutatna. Ezt követően megjeleníti a HTML-táblázat elejét, a kód végrehajtása során további sorokat ad a táblázathoz, majd végül megjeleníti a HTML-táblázat végét jelző tag-et.

ÉI

A while()-ciklus végigmegy a tömb minden elemén. Ehhez az each()-megközelítést használja, ezáltal a folyamatot nem befolyásolja, hogy a cikluson belül a kód hozzáad, illetve elvesz sorokat a táblázatból. Az első if() ellenőrzi, hogy létezik-e a fájl, arra az esetre, ha soha nem is létezett, vagy már törölték. A második if() megvizsgálja, hogy a fájl könyvtár-e. A kapcsolódó elseif() megvizsgálja, hogy a fájl fájl-e, az else()-ág pedig azt az esetet fedi le. ha a tömb adott eleme se nem fájl, se nem könyvtár. Ha az adott elem fájl, az unlink()-pa-ranccsal töröljük. Ha a fájl nem létezik vagy nem könyvtár, illetve ha már töröltük, akkor az adott elemet az unset()-paranccsal eltávolítjuk a $delete-tömbváltozóból.

A könyvtárakat az opendir()-, readdir()- és closedir()-parancsokkal olvassuk be (lásd a 8. fejezetet). A readdirQ minden elemet betesz a törlendő elemek tömbjébe, kivéve, ha az adott elem .-ot vagy ..-ot tartalmaz. Az egyes pont az aktuális könyvtárra vonatkozó mutató, míg a dupla az eggyel magasabb szintű könyvtárra mutat:

```
reset($delete);
print("<table bordér=\"3\">");
ptr("<em>Item</em>", "<em>Result</em>");
whiledist ($k, $v) = each ($delete) ) {
    if(file_exists($v))
        {
        if(is_dir($v)) {
            $dir = opendir($v);
            if(substr($v, -1) != "/") {
                $v .= -/"; }
            while($file_name = readdir($dir)) {
                if($file_name != "." and $file_name != "..") {
                    $delete[] = $v . $fi1e_name;


        closedir($dir);

        elseif(is_file($v))

        if(unlink($v))

               ptr($v, "deleted"};

            else

            ptr($v, "not

        deleted"),-unset($delete[$k]);

        else

            ptr($v, "not a directory or file");
```

```
            unset($delete[$k]


    else

        unset($delete[$k]);
        ptr($v,     "not
        found");
```

Miután a szkript végrehajtotta az előbbi kódot, az összes könyvtárnév, felülről lefelé, benne lesz a $delete-tömbben; a végső kódnak alulról felfelé kell elvégeznie az eljárást. Az array_reverse()-parancs megfordítja a tömb elemeinek sorrendjét, ezzel megoldja a problémát:

```
$delete    =    array_reverse(Sdelete);
```

Az alábbi kód végigmegy a $delete-tömbön és kitörli az elemeit. Az array_reverse()-parancs miatt a legalsó szintű alkönyvtárral kezdi, és legutoljára törli ki az általunk megadott főkönyvtárat. Az rmdirQ-parancs eltávolítja az argumentumában szereplő könyvtárat, feltéve, hogy a könyvtár üres és rendelkezünk a megfelelő jogosultsággal.

```
while(list($k,     $v)    =
    each($delete)) {
    if(file_exists($v)) {
        if(is_dir($v)) {
            if(ürmdir($v)) {
                ptr($v,
                "removed"); }
            else   {   ptr($v,            "not
                removed");



    else {
ptr($v,    "does   not   exist"); } }
print("</table>");
```

A 7.4 ábra a kis tesztlistára vonatkozó törlési táblázatot mutatja, a 7.5 ábra pedig a második futtatás eredményét, ahol láthatjuk a hiányzó fájlok eredményezte üzenetet. A lista rövidebbé tehető, ha úgy módosítjuk, hogy ne jelenjenek meg az alsóbb szinteken lévő elemek. Az alkönyvtárakon végiglépkedve keletkező bejegyzéseket eltárolhatjuk egy másik tömbben, ami párhuzamosan kerülhet feldolgozásra. Mikor egy ilyen rendszert tesztelünk, érdemes mindent megjeleníteni, hogy napvilágra kerüljenek a kiválasztás hibái.

\Rem		lassult \
|t:/Cop	of New Text Document.txt	deleted
y	Jfe. oftemp/hs~l.tmp	deleted
jt:/Copy	of temp/Copy (2) of temp/hs~4 .tmp	deleted
|t:/Cop	of temp/Copy (2) of temp/New Text Document.txt	deleted
jfc/Copy	of temp/Copy (2) of temp	jremoved.
(tiZCop	of temp	removed

7.4 ábra A törölt fájlokat és könyvtárakat tartalmazó fájllista

Item	\Result
t:/Copy ofNew Text Document.txt	not found
Í:/Copy of temp	jnot found

7.5 ábra Fájllista a már törölt könyvtárakon futtatott szkriptből

ötlet: Egy külön tipp Windows felhasználóknak: Mikor ezt a kódot tesztelitek, és a Windows Intézőt vagy valamilyen hasonló fájlkezelőt használtok a könyvtárak ellenőrzésére és a törlési folyamat követésére, letilthatjátok az rmdirQ-parancsot az egyik könyvtárra. Az rmdirQ-parancs nem fogja letörölni a fájlkezelő ablakban éppen megnyitott könyvtárat. Az olyan alkalmazások, mint a *Microsoft Word, szintén beragadva hagyhatnak itt-ott egyes fájlokat. Ha Wordot használunk a tesztfájlok létrehozására és nem tudjuk törölni a fájlokat vagy a fájlokat tartalmazó könyvtárt, akkor zárjuk be az alkalmazást, amellyel létrehoztuk* a *tesztfájlt.*

A modern számítógépekben olcsó a tárkapacitás de a nagyméretű képek és az audiovizuális fájlok gyors terjedésével még mindig könnyen megtölthetjük a winchestert. Biztosítani kell a lehetőséget a weblap építésében közreműködő társainknak, hogy a saját, elavult fájljaik kitakarításáról rendelkezzenek. A legjobb alkalom erre, mikor feltöltik a fájlokat. Ezeket az információkat összekapcsolhatod egy kirostáló alkalmazással, amelynek segítségével a felhasználók megtalálhatják az elavult fájlokat. Az már hab a tortán, ha egy külön mezőben megadhatják, hogy egy adott fájlnak mi a szerepe és egy másik mezőben meghatározhatják a dátumot vagy egyéb feltételeket a fájl törlésére vonatkozóan.

Hivatkozás:	oldal:
Az adatbázisok listázása	**144**
Lemezterület-kimutatás	**267**
Képek listázása	**384**

Még egy területen láttam hasznát ennek a kitakarító kódnak, mikor a PHP COM eszközét használtam különböző Microsoft Office-alkalmazásokkal. Az Office-alkalmazások néha rejtélyes módon lefagynak, és mindenféle ideiglenes fájlokat és félbehagyott tesztfájlokat hagynak maguk után, amelyeket kézzel kellett kitakarítanom. Bemásoltam a kitakarító kó-

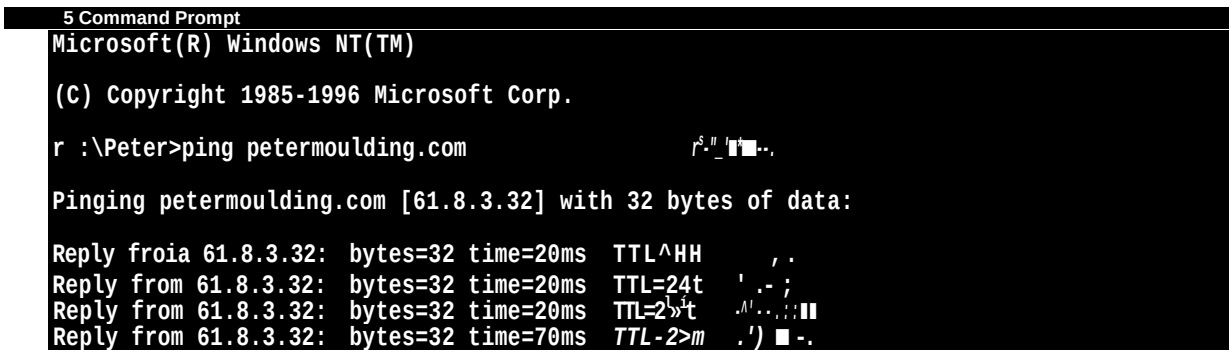
dot a tesztszkriptjeim elejére, és ezt használtam tisztogatásra. Egy fájlt hozzáadni a tömbhöz még mindig sokkal egyszerűbb, mint kézzel kitörölni vagy egy fájlparancs kódjába begépelni.

A következő" lépés a fájltisztogatás kiterjesztésében egy $copy-tömb hozzáadása volt. Ennek révén az eredeti tesztfájlokat a forráskönyvtárból átmásoltam egy tesztkönyvtárba, így az esetlegesen sérült fájlokat helyettesíthettem egy sértetlen másolattal. Bár ez a másolgatás és törölgetés a PHP-val egészen egyszerűnek és magától értetődőnek tűnhet, dolgoztam olyan projekten, ahol emberek több mint 50 ezer dollárt költöttek egy olyan szoftverre, amelyik alig csinált többet, mint fájlokat másolt és törölt a felhasználók által készített lista alapján. A legdrágább szoftvernek még a paraméterek beállítására szolgáló online-formja sem volt; az egész szövegfájlokon alapult, amelyeket kézzel szerkesztettek. Ha fájlkezelő rendszert készítesz a honlapod számára, kérlek, biztosíts a végfelhasználónak könnyen átlátható és könnven kezelhető formokat az adatok bevitelére.

Külső programok végrehajtása

Bizonyos dolgokat még a PHP sem tud, ezért szükséged lehet külső kiegészítő programok futtatására. Itt egy példa, hogyan futtathatsz külső programokat, a világ legegyszerűbb programját a pinget használva.

Aki valaha is használt pinget, az tudja, hogy kéréseket küld IP-címekre és felsorolja a válaszadásig eltelt időt. A 7.6 ábrán egy Sydneyben található munkaállomásról pingeltünk egy szintén Sydneyben lévő webszervert: a körutazás 20 ms-ot vett igénybe. Az Egyesült Államokban található Yahooo.com szintén Sydneyből pingelve 171 ms alatt válaszolt, ami nem rossz idő egy nemzetközi kapcsolat esetén, míg egy másik szerverből több mint 500 ms-ba telt kiszedni a választ, ami egy aktív kereskedelmi weblapnál jócskán meghaladja az elvárható értéket.



7.6 ábra Ping a parancssorban

Ha egy szkriptből szeretnél pingelni, használd az exec()-utasítást, mint az alábbi példa mutatja. A sztring a futtatni kívánt program neve. Ha a program nem a system könyvtárban

található, akkor a sztringnek tartalmaznia kell az elérési utat is. A $result-változó értéke a program outputjának utolsó sora és a print()-utasítással tudjuk megjeleníteni. Sajnos a pingelés eredményének utolsó sora üres sor:

```
$result = exec("ping");
print("<br>" . $result);
```

Ahhoz, hogy a program teljes outputját megkapd, adj hozzá egy tömbváltozót az exec()-parancshoz, mint a következő példa mutatja. Az output megjelenítését pedig a **print()** utasítás és a while()-ciklus segítségével oldhatod meg:

```
$result = exec("ping", $output);
while(list($k, $v) =
each($output))

    print("<br>" $k $v );
```

A kód outputja itt látható. Ez egy help oldalt tartalmaz, mert nem adtunk meg IP-címet vagy URL-t (néhány sort elhagytunk, hogy takarékoskodjunk a hellyel):

```
    Usage: ping [-t] [-a] [-n count] [-1 size] [-f] [-i TTL] [-v TOS] [-r
    count] [-s count] [[-j host-list] I [-k host-list]] [-w timeout]
    destination-list

    Options:
    -t Ping the specified hőst until
interrupted. <cut>
17: -w timeout Timeout in milliseconds to wait for each
reply. 18:
```


Próbáljuk ugyanezt a pinget újra ugyanezzel a szkripttel úgy, hogy megadjuk egy weboldal nevét. Nyugodtan próbálgassunk:

```
$result = exec("ping petermoulding.com", $output); while(list($k,
$v) = each($output))
    print("<br>" $k $v);
```

Az új output egy formás elrendezésben mutatja be a szerver válaszadási idejét, amit megjeleníthetünk a weblapunkon is, vagy kivághatunk belőle részeket, hogy a PHP-szkriptben további vizsgálatnak vessük alá:

```
19
20      Pinging petermoulding.com [61.8.3.32] with 32 bytes of
21      data:
22
23      Reply from 61.8.3.32: bytes=32 time=20ms TTL=244
24      Reply from 61.8.3.32: bytes=32 time=30ms TTL=244
25      Reply from 61.8.3.32: bytes=32 time=20ms TTL=244
        Reply from 61.8.3.32: bytes=32 time=60ms TTL-244
```

Most adjunk hozzá egy további kódrészt, mint a következőkben látható:

```
reset{$output);
while (list ($k, $v) = -ea,ch {$ output) {
    print("<br>" . $k . ": " . $v
```

A reset()-parancs visszaállítja a tömbpointert a tömb elejére, és meglepő módon az eredmény mindkét exec() outputját tartalmazza. A execQ az új outputot a tömb végéhez fűzi, így az outputokat naplószerűen összegyűjtheted, és később összesítve dolgozhatod fel:

```
Usage: ping [-t] [-a] [ -n count] [-1 size] [-f] [-i TTL] [-v TOS] [-r
count] [-s count] [[-j host-list] I [-k host-list]] [-w timeout]
destination-list

Options:
-t Ping the specified hőst until
interrupted. <cut>
-w timeout Timeout in milliseconds to wait for each reply.


Pinging petermoulding.com [61.8.3.32] with 32 bytes of data:

Reply from 61.8.3.32: bytes=32 time=20ms TTL=244
Reply from 61.8.3.32: bytes=32 time=30ms TTL=244
Reply from 61.8.3.32: bytes=32 time=20ms TTL=244
Reply from 61.8.3.32: bytes=32 time=60ms TTL=244
```

A passthru() az exec() egy alternatív neve; lássunk egy gyors demonstrációt. A következő kód megpingeli a Yahoo.com-ot és egy opcionális második paramétert biztosít a program visszatérési kódjának. Nincs lehetőség arra, hogy mint az execQ esetében, a teljes outputot eltároljuk egy tömbben, sem az output utolsó sorát nem kapjuk vissza. A teljes output közvetlenül a képernyőre kerül. Abban a verzióban, amit teszteltem, az output kétszer került elküldésre, kiegészítve a help-képernyővel, amely a kettő közt jelent meg:

```
passthru("ping                  $return_code);
yahoo.com"                      .
print("<br>return_code:        $return_code)
"
```

Mivel a passthru() ellenőrizetlenül és formázatlanul küldi az outputot a képernyőre, főleg olyan programokhoz használatos, amelyek bináris adatot, pl. grafikát vagy mozgóképet produkálnak. Küldd el a bináris outputnak megfelelő fejlécet, és futtasd a programot a passthru()-paranccsal. A bináris outputnak HTML- vagy egyéb output nélkül kell követnie a fejlécet.

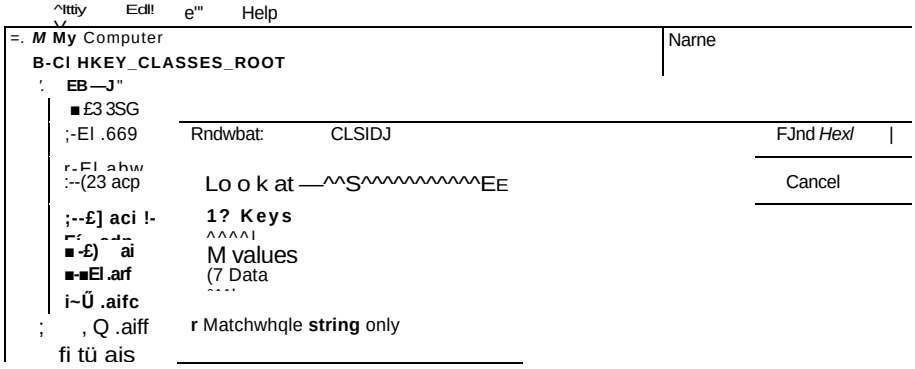
A system() az exec() és a passthruQ egy alternatív neve, lássunk erre is egy gyors szemléltetést. Az alábbi kód megpingeli a Yahoo.com-ot és egy opcionális második paramétert biztosít a program visszatérési kódjának. Itt sincs lehetőség arra, hogy mint az ececQ esetében, a teljes outputot eltároljuk egy tömbben. A $result-változóban az output utolsó sorát kapjuk vissza. A $return_code-változóban O-t kapunk, ha sikeres volt a pingelés, és 1-et, ha valamilyen hiba lépett fel:

```
$result = system("ping yahoo.com", $return_code);
print("<br>result: " . $result);
print("<br>return_code: " . $return_code);

result: Reply from 216.115.108.243: bytes=32 time=171ms TTL=244
return code: 0
```

Word-dokumentumok Rtf-formátumba konvertálása COM segítségével

Az alábbi példa bármilyen Microsoft Office-termékkel vagy tetszőleges COM-kompatibil alkalmazással használható Windows vagy Windows NT alatt. Az egyetlen nehézség annak elérése, hogy az egyes alkalmazások pontosan azt az utasítást hajtsák végre, amit szeretnél.

Először a célalkalmazás COM-azonosítójára lesz szükségünk, amelyet megtalálhatunk a Visual Basic for Applications (VBA) dokumentációjában vagy a Windows Regisztrációs Adatbázisában, A Regisztrációs adatbázist a regedit-paranccsal megnyitva böngésszük a HKEY_CLASSES_ROOT-könyvtárat vagy keressünk rá a CLSID-re. A *7.7* ábrán a Regisztrációs adatbázis megnyitott HKEY_CLASSES_ROOT-könyvtárát láthatjuk és a Keresés ablakot, amint éppen a keresés elején tartunk. A 7.8 ábra a Word CLSID bejegyzését mutatja (a Regisztrációs adatbázisban rengeteg CLSID-bejegyzés található). A számunkra szükséges név a CLSID-könyvtárat tartalmazó könyvtár neve, ami példánkban Word. Application.



7.7 ábra Windows Registry HKEY_CLASSES_ROOT

Kíváncsi voltam, milyen opciók működnek a saveas- (Mentés másként) utasítással, ezért létrehoztam egy saveas-makrót VBA-ban, ami alább látható, és teszteltem a megfelelőjét PHP-ban. A Word verziója 97 SR-2 volt, a PHP-é pedig 4.0.4pll. A FileName működött első paraméterként és a FileFormat másodikként, de a ReadOnlyRecommended-nek mint

hetedik paraméternek nem volt hatása. A PHP COM interfésze gyorsan fejlődik, így mire a COM-ot PHP-bó'l használod, könnyen lehet, hogy sokkal több működő és dokumentált opciót találsz:

7.8 ábra Windows Registry CLSID eleme

```
ActiveDocument.SaveAs    FileName:="comtest.rtf",
    FileFormat:=wdFormatRTF, LockComments:=False,
    Password:="", AddToRecentFiles:=True,
    WritePassword    :="",
    ReadOnlyRecommended:=False,
    ErabedTrueTypeFonts: =False,
    SaveNativePictureFormat:=False,
    SaveFormsData:=False, SaveAsAOCELetter:=
    Falsé
```

Ha mindezt saját kezűleg szeretnéd kipróbálni, a kívánt Office-termékben válaszd az Eszközök Makró | Uj makró rögzítése menüpontot. Végezd el a megfelelő műveleteket, amelyeket COM-on keresztül szeretnél megvalósítani, állítsd le a rögzítést, és nézd meg a makrót a Microsoft Office-hoz biztosított Visual Basic-szerkesztőben. A Visual Basic-szerkesztőben az objektumok tulajdonságainak, mint pl. a wdFormatRTF-nek a definíciói is megtalálhatók, tehát beállíthatod a kívánt értékeket, vagy PHP-ban is létrehozhatod a megfelelő definíciókat. A következő kód a fájlformátum definíciók felsorolását tartalmazza:

```
define("wdFormatDocument",      0);
define("wdFormatTemplate",      1);
define("wdFormatText",      2 ) ;
define("wdFormatTextLineBreaks",      3 ) ;
define("wdFormatDOSText",      4);
define("wdFormatDOSTextLineBreaks",      5);
def ine ("wdFormatRTF",      6 ) ,-
define("wdFormatUnicodeText",      7);
```

Az alábbiakban bemutatott, a COM-műveletek végrehajtását szolgáló kód megtévesztően egyszerű. Az első sor a $w-változóban létrehozza a com-objektum egy új előfordulását a word-alkalmazás számára és megnyitja a kommunikációs csatornát az alkalmazás számára. Az if()-utasítással ellenőrizzük, hogy ez sikerrel járt-e. Az én tesztgépemen a Word néhány alkalommal nem válaszolt, mikor egy dokumentum már meg volt nyitva szerkesztésre. A Microsoft Office-alkaímazások nem reagálnak, ha éppen meg van nyitva egy párbeszéd-

ablak, mint például a Mentés másként ablak. Az alkalmazások akkor is kudarcot vallanak, ha már használatban van az a fájl, amit a COM-on keresztül próbálunk elérni, és esetenként képtelenek megszakítani a kapcsolatot egy-egy fájllal. Ha Worddel szerkesztesz egy állományt, lehet hogy fizikailag be kell zárnod a Wordot, mielőtt más alkalmazással megnyithatnád az adott fájlt.

A $w->Vistble = 1-utasítás hatására az alkalmazás ablakának aktívvá kellene válnia, így Iá: hatóvá válnak a COM-műveletek, miközben tesztelünk. Néhány embernek működik, másoknak nem.

A Sw->Documents->Open()-parancs megnyit egy dokumentumot a Wordben, és nagyjából megegyezik a Word Fájl | Megnyitás parancsával. A **$w->Documents[l]->saveas()-pa-**rancs elmenti a dokumentumot az argumentum második paraméterben megadott valamelv. az előbbi felsorolásban szereplő formátumban. A $w->Quit()-utasítással kilépünk az alka^ mázasból:

```
if($w = new com("word.application") )
    {
    print("<br>Opened Word version " . $w->Version) ;
    if($w->Visible = 1)
        {
        print("<br>Word should be visible.");
        }
    else
        {
        print("<br>Visible failed.");
        }
    if($w->Documents->Open("t:/comtest.doc"))
        {
        print("<br>Document opened."); }
    else     {     print("<br>Open
        failed.");

    $w->Documents[1]->saveas("t:/comtest3.rtf"
    $w->Quit(); }                                    wdFormatRTF)
else  {  print("<br>Word  failed  to
    open");
```

A következő felsorolás az előző részben nem szereplő COM-utasításokat tartalmazza. Használatuk az alkalmazástól függ. Néhányuk még mindig nagyon megbízhatatlanul működött, mikor próbáltam:

- **COM()** COM-osztály.
- **VARIANTO** VARIANT-osztály.
- **com_load()** Új referenciát hoz létre egy COM-komponenshez.
- **com_invoke()** Meghív egy COM-komponenshez tartozó eljárást.

com_propget() Egy COM-komponens tulajdonságának értékét adja vissza.
com_get() Egy COM-komponens tulajdonságának értékét adja vissza.
com_propput() Egy COM-komponens tulajdonságának értékét állítja be.
com_propset() Egy COM-komponens tulajdonságának értékét állítja be.
com_set() Egy COM-komponens tulajdonságának értékét állítja be.
com_addref() Növeli egy komponens-hivatkozás számlálóját.
com_release() Csökkenti egy komponens-hivatkozás számlálóját.

HTTP hitelesítés példa

Ez a példa egy egyszerű HTTP-hitelesítést alkalmaz egy demonstrációs weblapra, de mindezt alkalmazhatod egy honlap összes oldalára vagy akár adminisztrációs panelek és opciók bonyolult összességére is. A hitelesítés lényege a bejelentkező oldal és a felhasználónév valamint a jelszó ellenőrzése. Ha van egy működő' HTTP-hitelesítésed, szabadon alkalmazhatod weblapok tetszőleges kombinációjára.

Először a PHP header()-utasítására lesz szükséged, amely tetszőleges HTTP-fejlécet küld. Jelen esetben egy "Authentication Required" (Hitelesítés szükséges) és egy 401-es fejlécet kell küldened, mivel ennek hatására a böngészőben előugrik egy ablak, amely felhasználói nevet és jelszót kér, mint az a 7.9 ábrán átható. A tartomány hozzárendel egy nevet hitelesítéshez, amely szintén megjelenik a bejelentkező képernyőn. A 7.9 ábrán a tartomány neve **birds,** amit a doménnév követ.

7.9 ábra Bejelentkezés képernyő

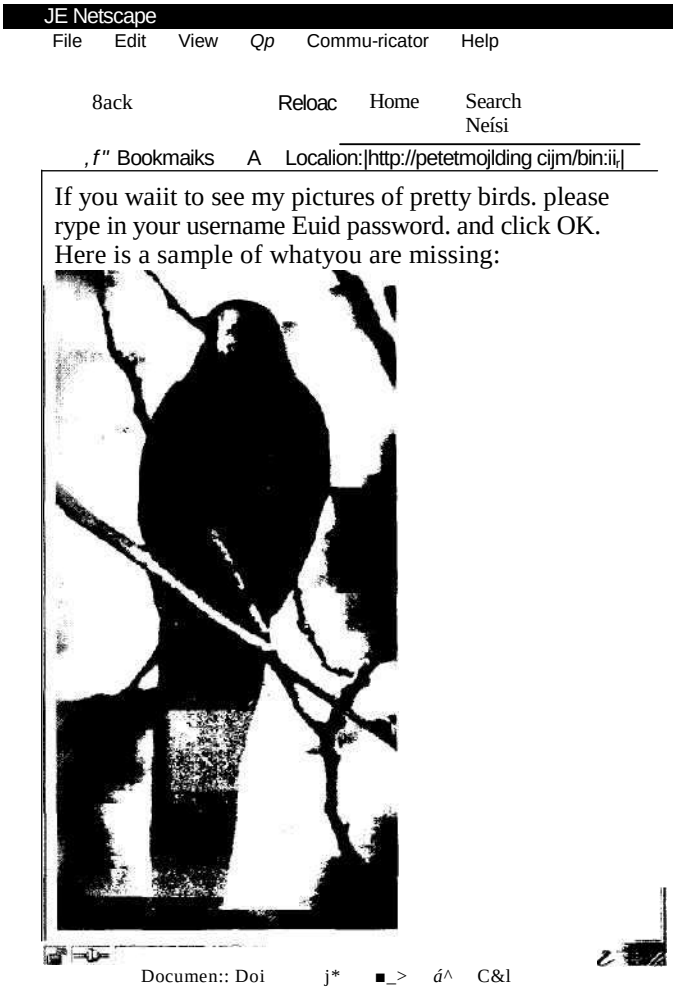
A következő kód egy egyszerű, tesztelésre alkalmas bejelentkező oldalt mutat a **petermoulding.com/birds/** címen. A kód először a $PHP_AUTH_USER-változó létezését ellenőrzi, majd beengedi a látogatót, vagy a két fejléc elküldésével megjeleníti a bejelentkező ablakot. Ha a Mégse vagy az OK gombra kattintunk, a böngésző ugyanannak az oldalnak küldi vissza az információkat, amelyik a fejléceket küldte, tehát a szkript másodjára is lefut. Másodjára a szkript látja a felhasználói nevet, vagy elhagyja a bejelentkező ablakot és végrehatja a bejelentkezés utáni utasításokat. A 7.10 ábra egy sikertelen bejelentkezés utáni böngésző ablakot mutat:

```
if(isset($PHP_AUTH_USER)) {
   print("Hello " . $PHP_AUTH_USER
       . ",<br>You are logged on with password
   if(isset($PHP_AUTH_TYPE))                                    .
                                                          $PHP_AUTH_PW)

      print("<br>You are using authentication type      $PHP_AUTH_TYPE^■

   print("<p>Here is my first pretty bird, a King Parrot, sitting on" .
       " a branch outside my office window. The King Parrots arrive"
         " when the noisy Eastern Rosellas are not covering the" .
       " branches.</p>");

else

   header ( "WWVJ-authenticate : basic realm=\ "birds\ "" ) ;
   header( "HTTP/1.0 401 Unauthorized");
   print( " I f you want to see my pictures of pretty birds,
   please type"
       . " in your username and password, and click OK. Here is
       a sample"
       . " of what you are missing:<br>");

print(img("kingparrot.jpg" , 2 50, 50 0, "King Parrot"));
```


Ez a biztonsági szint éppen arra elegendő, hogy szemléltessük a fejléceket. A valóságban a felhasználói azonosítót és jelszót egy adatbázis bejegyzéseivel vetnéd össze. A jelszót titko-sítanád, mielőtt elmented az adatbázisba, nehogy mások ki tudják olvasni az adatbázisból. Mikor valaki bejelentkezik, ugyanazzal a kulccsal titkosítod a jelszavát, és ezt hasonlítod össze az adatbázisban tárolt titkosított jelszavakkal. A böngészőben JavaScriptet használhatsz a jelszó titkosítására, mielőtt átadnánd a szervernek. Ez csökkenti annak a veszélyét, hogy valaki elolvassa, míg keresztülhalad a hálózaton.

Ha egy felhasználó bejelentkezett, a sessionje egy cookie vagy egy sessionazonosító segítségével tartható fenn az URL-jében. Mivel mindkettő lemásolható, egy titkos kódot kell elhelyezni a cookie-ban, aztán ellenőrizni kell a beérkező lekéréseket, hogy a megfelelő titkos kódot tartalmazzák-e. A titkos kódnak tartalmaznia kell valamit, ami minden bejelentkezésre egyedi, pl. a microtime()-utasítás által megadott időpontot, továbbá tartalmaznia kell valamit, ami alapján két böngésző megkülönböztethető, pl. a böngésző IP címét. (Egy proxy-szerver mögötti böngésző IP-címének megszerzése elég bonyolult lehet; némelyik az eredeti IP-címet egy HTTP_X_FORWARDED_FOR-fejlécként küldi el.) Ezen túl rövid időtúllépési időszakokat kell biztosítani arra az esetre, ha valaki elmegy ebédelni és nyitva hagyja a böngészőjét.

Tehetsz még néhány további óvintézkedést attól függően, hogy mennyire vagy paranoiás. Hiába teszed azonban bombabiztossá honlapodat, ha a szervered egyike az ugyanazon a gazdaszerveren futó sok kis virtuális szervernek, és egy gyenge biztonságú közös adatbázison osztozik a többi honlappal, vagy ha a gazdaszerveren más biztonsági rések vannak, például FTP-hozzáférés.



7.10 ábra A hibás bejelentkezéskor kapott képernyő

Hogyan igazítsuk a kódot a környezethez?

A phpinfoQ-utasítással megtekintheted a PHP környezetét, de nem befolyásolhatod, hogyan fusson le a kódod a phpinfo() alapján. Az alábbi utasítások elő'segítik, hogy a kódod jobban igazodjon a környezethez.

Hibanaplózás

Először is minden szkríptváltozást naplózni kell ahhoz, hogy lásd a teszt eredményeit. Néhány változás hatása függhet a böngésző típusától, ezért nem árt minden nap megvizsgálni az eredményeket, hogy felfedezd, mi történik, ha valaki egy elavult böngészővel látogatja az oldalaidat. Néhány változás a lap megjelenítése előtt következik be, néhány a lap megjelenítése után, néhány pedig olyan lapokon, amelyek soha nem kerülnek megjelenítésre, például átirányító oldalakon, ezért minden beérkező üzenetet naplóznunk kell.

A legegyszerűbb módja, hogy rögzítsük a PHP naplóállományát az error_log()-utasítás használata, amit a következő példa szemléltet. Az error_log()-parancs a php.ini beállításaitól függően a PHP vagy a rendszer naplóállományába ír, de tetszés szerint írhat egy külön fájlba, vagy e-mailt is küldhet. Úgy találtam, hogy az error_log()-utasítás üzenetei nem mindig jutnak el a naplóállományba, ha a PHP vagy az Apache valamely belső hibával küszködik, ezért írtam egy külön naplóparancsot:

```
error_log("Changed code to fit odd browser");
```

A logtofile()-parancs argumentumában tetszőleges sztring szerepelhet, megnyit egy naplózásra szánt fájlt, hozzáfűzi a sztringet, a dátumot és az időt, majd végezetül bezárja a fájlt. Az idő a standard rendszeridő helyett a microtime()-utasítástól származik, így a fájl arra is használható, hogy feljegyezze az időt egyazon szkript különböző bejegyzései közt, és kimutassa azokat az eseteket, mikor a szkript egy része szokatlanul sok időt emészt fel. A logtofileQ-utasítás fopen()-parancsában nevezz meg egy saját tulajdonban lévő könyvtárat, amelyhez korlátlan hozzáférésed van, és bármikor eltávolíthatod a régi feljegyzéseket. Az fopen()-parancs a paraméterének hatására az fwrite()-parancs egy létező fájlhoz fűzi hozzá a bejegyzéseket, ha a fájl nem létezik, az fopen() létrehozza a fájlt.

Ennek a parancsnak a kódját a következő példában láthatjuk. Mikor egy szkriptet tesztelsz, tégy egy naplóüzenetet a szkript azon részei elé, illetve mögé, ahol külső erőforrásokat, például adatbázis-lekérdezéseket használ, így fogalmat alkothatsz arról, hogy normális esetben mennyi ideig fut le a szkript. Ha valamit változtatsz a környezeten, például indexelést adsz az adatbázishoz, vizsgáld meg az időket, hogy lemérd a hatását.


```
function Logtofile($text="")
    {
    $self = "";
    if(isset($PHP_SELF))
        {
        $self = $PHP_SELF;
        } if($log = fopen("t:/log.txt",
    "a"))
        {
        list($m, $t) = explode(" ", microtime());
        fwrite($log, date("Y-m-d H:i:s", $t)  . substr($m, 1) . "
            " . $self . " " . $text . "\n");
        fclose($log);


logtofile("New test"
```

Az eredmény például így néz ki:

```
2002-06-07 16:11:05.61830400 New
test
```

Menekülő látogatók

Van a honlapodon egy olyan oldal, amelyiken sokáig töltődik be a fő információ az oldal tetején? Az ilyen oldalakon a látogatók gyakran az első néhány sor elolvasása után a Vissza gombra kattintanak. Az oldalakon a képernyőket részekre oszthatod, és külön-külön vizsgálhatod a látogatók reakcióit. Először közölnöd kell a PHP-val, hogy hagyja figyelmen kívül, ha a látogatók bezárják az ablakot vagy elkattintanak, mint azt a következő kódrész mutatja:

```
ignore_user_abort(true);
logtofile("Start of very   long database search page")

if(connection_aborted())
    {
    logtofile("User left the page");
    }
else
    í
    // continue with code
```

Az ignore_user_abort()-utasítás immúnissá teszi a szkriptet az elkattintással szemben. A php.ini-ben beállíthatod, hogy a PHP hagyja figyelmen kívül a látogatók bármiféle elkattin-tását, de ez nem volna túl jó ötlet, tekintve, hogy a webszerver elég sok időt töltene olyan oldalak elküldésével, amelyeket senki nem néz meg. Ha egy szknpted figyelmen kívül hagyja az elkattintást, a connection_aborted()-paranccsal ellenőrizheted, hogy elkattin-tott-e a látogató.

Hová kell elhelyezni az ellenőrző pontokat? Nagy erőforrás igényű műveletek közé, mint például adatbázis-lekérdezések. Ha egy szkript megjeleníti az árbevétel szerinti első tíz terméket, majd ezeket követően az összes többi terméket kategóriánként csoportosítja, akkor ehhez minden bizonnyal két lekérdezést használ, és mindkettő jelentős erőforrásokat vesz igénybe. Az eltávozó látogatók valószínűleg a tízes toplista megtekintése után teszik ezt, tehát a két lekérdezés közé érdemes egy ellenőrzési pontot tenni. Ha lassú adatbázis-kapcsolattal dolgozol, pl. ODBC-t használsz, akkor minden egyes rekord beolvasása elé tehetsz egy ellenőrzést, hogy meghatározd, érdemes-e elvégezni a műveletet.

Szkript időtúllépés

Néha a szkriptek túllépik az időkorlátot, mielőtt az összes adatot feldolgozták volna, és a PHP megszakítja a futásukat. A következő kóddal felismerhetjük és feljegyezhetjük ezeket az eseteket. Először írjunk egy függvényt, amely naplózza, ha egy oldal végrehajtása megszakad. A logshutdownQ nevű példaprogram a logtofile()-utasítást használja a feljegyzés rögzítésére. A feljegyzés tartalma a connection_timeout()-utasításon alapszik, amely IGAZ értéket ad, ha a szkript túllépte az időkorlátot. Ez az utasítás csak akkor lesz hasznos, ha azután fut le, hogy a PHP befejezte a kérdéses szkriptet, ezért közölnünk kell a PHP-val, hogy ezt a szkriptet minden más után futtassa, erre a register_shutdown_function()-utasítást használjuk:

```
function logshutdown()

   if(connection_timeout())

       logtofile("The script timed out"); else

   logtofile("The script shut down normally"); }

register_shutdown_function("logshutdown");
```

Az adatbázis kiválasztása

Ha az egyik honlapod PostgreSQL-t használ, a másik MySQL-t, az extension_loaded()-utasítással ellenőrizheted, melyik van feltelepítve. Ez az ellenőrzés bármilyen PHP-kiter-jesztésre, pl. grafikus vagy matematikai függvényekre is elvégezhető, így megállapíthatod, mit használhatsz a kódot futtató szerveren. Ehhez tudnod kell, hogy a kiterjesztés milyen néven szerepel a php.ini-ben. A Unixos .so kiterjesztések Windows alatt dll névre hallgatnak, ezért mindkettőt ellenőrizheted, mint a következő kód mutatja:

```
if(extension_loaded("php_mysql.dll") or
extension_loaded("php_mysql.so"))

   $database = "my";
   logtofile("Using MySQL");

elseif(extension_loaded("php_pgsql.dll" )
    or extension_loaded{"php_pgsql.so"))

 $database = "pg";
   logtofile("Using PostgreSQL");

elseif(extension_loaded("php_mssql.dll")
    or extension_loaded("php_mssql.so"))

   $database = "ms";
   logtofile("Using Microsoft SQL Server");
```

Ez a példaprogram a $database-változó értékét adja meg. A $database-t az if ()- vagy a switch()-utasításokkal használva meghatározhatod, hogy a szervertől függően melyik kód fusson le. így a kód könnyedén eldöntheti, hogy BCMath-t vagy GMP-t, Flash-t vagy PNG-t használjon. Gondoskodj róla, hogy a kiválasztás rendelkezzen hibakezeléssel, mint pl. egy figyelmeztető üzenet vagy e-mail küldése, ha egyik opció sem elérhető. Úgy is jár-hatsz, hogy feltelepítesz egy alkalmazást, kisétálsz az ajtón, a rendszergazda pedig letörli az alkalmazáshoz szükséges kiterjesztést.

Böngésző-alapú kódok

Az alábbi egyszerű módon módosíthatod a programjaidat, hogy jobban illeszkedjenek a böngészőhöz. Csak illeszd be a következő' kódot, és a böngészők információit illetően hagyatkozz a browscap.ini-fájlra, amit rendszeresen frissíts, például a www.cyscape.com/asp/browscap/ oldalról:

```
Sbrowser = get_browser();

if(Sbrowser->javascript)
    {
li'  logtofile("Generating JavaScript");
    // create JavaScript here
```

A get_browser()-utasítás által megadott objektum felsorolja azokat az opciókat, amelyek valószínűleg elérhetőek a szkriptet olvasó böngészőben. Nem véletlenül fogalmaztam úgy, hogy *valószínűleg.* Ha a browscap-fájl nem friss, akkor félrevezethet egy bizonyos böngészővel kapcsolatban, vagy az is előfordulhat, hogy rosszul ismeri fel a böngészőt. A browscap a böngésző által küldött $HTTP_USER_AGENT-sztringen alapul és egy forgalmas oldal napjában több száz szokatlan agent-sztringet kap, amelyek közül sok teljesen megfejthetetlen.

A böngésző lehetőségeinek statikus listája alapján nem tudod megállapítani, hogy fel vannak-e telepítve bizonyos opciók, vagy néhányat esetleg kikapcsoltak a telepítés óta. A browscap-fájlok csak azt tudják megmondani, hogy a böngésző támogatja a CSS2-t, de azt nem tudják, hogy a CSS2 mely része működik az adott böngészőben, vagy hogy milyen komoly következményekkel járhat, ha másodosztályú CSS2-t alkalmazol.

A következő példát a PHP 4.0.5 browscap-fájljából vettük. Kiderül belőle, hogy a böngésző támogatja a cookie-kat, de az nem, hogy engedélyezve vannak-e. Az sem derül ki, hogy a webTV-képernyők korlátozott méretűek és felbontásúak, ezért kisebb képeket igényelnek. Általános szabályként használható, hogy ha a browscap szerint valamely funkció nem támogatott, akkor ne használjuk az adott funkciót, míg ha egy funkciót határozottan elérhetőnek tüntet fel, akkor ez csak a lehetőséget jelenti. Ha az oldalunk megtekintéséhez egy nem szokványos böngészőkiegészítő szükséges, akkor tájékoztassuk a látogatót a követelményekről, és biztosítsunk számára egy tesztelési lehetőséget, hogy ellenőrizhesse böngészője képességeit, valamint lássuk el instrukciókkal arra az esetre, ha böngészője nem megfelelő.

```
[WebTV 1.0] brows
er =WebTV
Version=l.0
majorver=#l
minorver=#0
frames=FALSE
tables=TRUE
cookies=TRUE
backgroundsounds =TRUE
vbscript=FALSE
```

```
javascript=FALSE
javaapplets=FALSE
ActiveXControls=FALSE
Winl6=False
beta=False
AK=False
SK=False
AOL=False
```

Utasítások ellenőrzése

Minden utasítás rendelkezésre áll a honlapodon? Ha az alkalmazásodhoz egy új függvény szükséges, ellenőrizheted a PHP-verzióját vagy a function_exists()-utasítással egyszerűen meggyőződhetsz arról, hogy létezik-e az utasítás, mint a következő kód mutatja. A példaprogram ellenőrzi, hogy az egyik kiterjesztett matematikai utasítás elérhető-e, ha nem a naplófájlban rögzít egy hibaüzenetet. Az utasítások ellenőrzése akkor célszerű, ha a megfelelő PHP-verziót használod, és a szükséges opciók is fel vannak telepítve, csak éppen elfelejtetted beágyazni a megfelelő fájlokat az include-paranccsal.

```
if(function_exists("beadd"))

    // perform calculation using BCMath,

else
    {                                                    tt ]
    logtofile("BCadd missing"); }                        -31
```


Ellenőrzés a fejlécek elküldése előtt j

Ha a kódodat valaki más használja a szkriptjében, akkor előfordulhat, hogy nem megbízható a HTTP-fejlécek küldése. Mielőtt átirányító fejléceket vagy cookie-kat küldesz, a headers_sent()-paranccsal ellenőrizd, hogy megbízható-e a fejlécek küldése:

```
if(headers_sent())
    {
    logtofile("Too iate to send headers");
    }
else
    {
    //  send  the  headers
```

A HTTP-fejlécek egy csoportban kerülnek elküldésre az oldal elején, abban a pillanatban, amikor az oldal bármilyen outputot produkál. A PHP még egy szimpla, az első PHP-tag előtt véletlenül leütött szóköz vagy egy beágyazott fájl utolsó PHP-tag-je után szereplő karakter hatására is elküldi a fejléceket. A véletlenszerűen elküldött fejlécek elkerülése végett *győződj* meg arról, hogy nincsenek szóközök vagy újsor-karakterek az első <? előtt, és törölj ki minden karaktert az utolsó ? > után.

A PHP ellenőrzése

Vannak helyzetek, mikor ellenőrizned kell a PHP-verzió számát. A PHP 4.0.4 Win32 bináris állományai jól működő COM-támogatást tartalmaztak, de a 4.0.5-ös verzióban már problémák voltak ezen a területen. Ha olyan kódot írsz, ami a COM-hoz hasonló sajátosságokat használ, akkor lehet, hogy egyes PHP-verziók esetén hibaüzenetre lesz szükség.

A **phpversionQ** értéke a verzióra vonatkozó sztring, amely valahogy így néz ki: 4.0.7-dev. Az eredménytől függően futtathatsz különleges kódrészeket, amelyek az egyes verziók hibáit kezelik. A következő példa lekérdezi a verziót, és megvizsgálja, hogy szükséges-e különleges eljárás. Egy jó indok arra, hogy mindent naplózzunk, beleértve a verziószámot is, hiszen ez segít a távoli szerverekről küldött naplófájlok értelmezésében, amelyeket egy helyi rendszergazda esetleg újrakonfigurált.

```
$version    =
phpversion();
logtofile(Sversion);
if($version    ==
"4.0.5")
    {
    logtofile("Sorry, too COMplicated");
    }
else
    {
    // COM code here
```

Ha a kódod függ a verziótól, akkor telepítési és frissítési részek elé helyezd a verzió ellenőrzését, hogy az emberek az előtt szembesüljenek a problémával, mielőtt a telepítés vagy a frissítés olyan pontra érkezik, ahonnan nem léphetnek vissza.

A memória ellenőrzése

Mielőtt egy új alkalmazást beindítanál, szeretnél mindent alaposan tesztelni. Mi történik, ha a tesztadatbázis csak 100 sort tartalmaz, vagy ha a legnagyobb kép is csak közepes méretű? Hogyan tesztelheted a rendszer teljesítményét azt modellezve, hogy egyszerre számos szkript nagy fájlokat olvas?

Próbáld ki a **leakQ**-utasítást, amely szándékosan fogyasztja a memóriát. A **leak()** azt teszteli, hogyan képes a PHP szükség esetén erőforrásokat felszabadítani. A teszt php.ini fájlomban 30 MB van beállítva, tehát minden leak()-utasítás 29 MB memóriát fog lefoglalni. A tesztgép 600 MB virtuális memóriával rendelkezik, tehát csak 25-ször kell lefuttatnom, hogy kiderüljön, működik-e a PHP kitisztító mechanizmusa. 25-ször párhuzamosan meg-híva a szkriptet, letesztelhetem, hogyan kezeli a virtuális memória a terhelést:

```
leak(30408704);
```

Furcsán hangzik, hogy a memóriát egy alkalmazásból teszteled, miközben a szkripted többszörösen beágyazott különböző szoftverekbe, köztük a PHP-ba, a webszerverbe (amely maga is számos réteget tartalmazhat) és az operációs rendszerbe, amelyek mind korlátozásokat szabnak a memória méretére. A rétegek nagy része visszafogja a teljesít-

ményt, ahogy közeledsz a maximális memória-mérethez. Próbáld ki, meddig tudsz elmenni (közben figyeld, hogyan változik a teljesítmény).

Szokatlan formátumok megjelenítése

Ha új típusú tartalommal kísérletezel, és a szerveredet olyan emberek tartják karban, akik megkésve adják hozzá az új MIME-típusokat a konfigurációs fájlokhoz, akkor meg kell kerülnöd a szokásos rendszert, hogy tetszőleges tartalmat nyújts a látogatóknak. Ez a megoldás megmutatja, hogyan tudod megjeleníteni a friss tartalmat az fpassthru()-utasítás révén, és nagyon hasonlít a következő megoldás kódjához és utasításához.

A következő kód egy tesztoldal, amely három változót vesz át egy adatbázisból (vagy egy formból vagy egy URL-ről, vagy akárhonnan, ami kényelmes a teszt számára). Ebben a pél- i dában magában a kódban gépeltük be az értékeket. Az fopen()-utasítás megnyitja az új fájlt j írásra és olvasásra, ha már létezett, akkor törli a tartalmát, ha nem létezett, akkor létrehozza. A középső részben a kód létrehozza az új fájlformátumot. A rewind()-parancs visszaállítja a fájl pointerét a fájl elejére, és az fpassthruQ-parancs elküldi a fájlt a böngészőnek és bezárja a fájlt:

```
$directory = "/tmp/";
Sname = "roo.jpg";
$type = "image/jpeg";
$file = fopen($directory . Sname, "w+");

// lots of code to generate the new file

header("Content-Type:      $type);
rewind($file);
fpassthru($file);
```

Ha a tesztfájlgeneráló kódot ebbe a keretbe foglalod, akkor azonnal láthatod az outputot a böngészőn. Az egyetlen szépséghiba, hogy a tesztfájl létrehozása után el kell küldeni egy fejlécet. Ez azt jelenti, hogy nem küldhetünk hibafelismerő üzeneteket a képernyőre, mert blokkolnák a fejlécet. Hozzáadhatunk egy algoritmust, ami csak akkor küldi el a fejlécet és a tesztfájlt, ha nincsenek hibaüzenetek (lásd fentebb a headers_sent()-utasítást), vagy szét vághatjuk a létrehozást és a megjelenítést két külön oldalra. Két oldallal minden szükséges üzenetet megjeleníthetsz, és a második oldalon a readfileQ-paranccsal jelenítheted meg a tesztfájlt, amint azt a következő megoldásban bemutatjuk. ;

4

Képek biztonságos megjelenítése

Vannak helyzetek, mikor egy képet vagy egyéb titkos információt csak egy felhasználónak szeretnél megmutatni, de nem akarod a webszerver standard biztonsági megoldását használni. Mindez megoldható, ha a biztosítani kívánt fájlokat a webszerver könyvtárán kívül helyezed el, és az fpassthru()-parancshoz hasonló readthruQ-val továbbítod a felhasználónak, amit a következő Gyors megoldásokban ismertetünk.

Először szükséged lesz egy oldalra, amely átveszi a kép nevét és megjeleníti a képet. A következő példakód pontosan ezt teszi a rendkívül egyszerű readfileQ-utasítás révén. A readfile() argumentumában bármilyen fájlhivatkozás szerepelhet, URL-ek vagy akár FTP-hivatkozások is. A tesztelés megkönnyítése érékében a példában a fájl nevét egy nem biztonságos URL-lekérdezés paraméteréből veszi át, de egy működő honlap esetén érdemes volna sessionöket beállítani a 19. fejezet alapján, és egy sessionrekord egyik mezőjében átadni a fájlnevet. A végfelhasználó nem férhet hozzá a sessionrekordhoz, így nem cselezheti ki a rendszert, nem veheti rá arra, hogy ne a megfelelő fájlt küldje el, vagy elküldjön egy fájlt, mikor nem kellene.

A fájl típusa szerepel a kódban, de átadható a sessionrekorddal. A tartalom típusát el kell küldeni a HTTP-fejlécrekordban, és itt kézzel kell beállítani; de a sessionrekorddal szintén továbbítható, ezáltal teljesen általános fájlokat is megjeleníthetünk:

```
<?php
if(isset($image))

    header ("Content-Type : image/jpeg") ,-
    readfile("i:/usr/home/petermoulding/secure/" . $image      •jpg")
```

A kép fejlécekkel együtt jelenik meg egy oldalon, formázással, és csak egyetlen HTML-tag-re van szükség a kép megjelenítéséhez, mint a következő kódban látható. A teszteléshez a kép nevét egy URL-lekérdezés image=roo-paramétereként adjuk át, de a biztonságos rendszerben ez egysession rekordon keresztül történne.

Ahhoz hogy jól formázott HTML-t kapjunk, az <img> tag-nek be kell állítani a magasságát (height), a szélességét (width) és egy alt-paramétert. Mikor egy különleges képet (vagy bármilyen egyéb fájlt) helyezel el egy oldalon, valószínűleg létrehozol egy adatbázisbejegyzést, amely a fájl leírását tartalmazza, és kereskedelmi megoldásoknál egy árat is hozzárendelhetsz; a méretet és az alt-információt ezen a szinten érdemes hozzáadni. Mikor lekérdezed a leírást és az árat, a méretet és az alt-információt is visszanyered, és mindezt a session-rekordban tárolod a megjelenítendő oldal számára:

```
print("<img src=\"./readfile.html?image=roo\""
    . " width=\"280\" height=\"250\" border=\"0\" alt=\"\">" );
```


Képinformáció kiíratása 385

Ezt a technikát bármilyen fájlra alkalmazhatjuk. Mivel a fájltípus az adatbázisban van eltárolva és dinamikusan küldöd el, olyan fájltípust is elküldhetsz, amelynek a továbbítására nincs konfigurálva a webszerver. Ha a webszerver nem úgy lett konfigurálva, hogy felismerjen egy fájltípust, a fájl típusa nem fog megjelenni a böngészőben a böngésző adatlekérésének következményeként.

A webszerver által nem ismert fájltípusok és a rajta nem hozzáférhető helyen tárolt fájlok által kínált dupla biztonsági szint meghaladja számos nagy weblap biztonsági szintjét. Még tovább mehetsz, ha olyan oldalakat alkalmazol, amelyek egy tűzfal mögött elhelyezett szerverről veszik a fájlokat, ezáltal elérheted a köznyelvben paranoiásnak nevezett szintet.

8. fejezet

Fájlok

Áttekintés

Az adattárolás alapegysége minden operációs rendszerben a fájl. A fájlok számtalan formátumban és funkciókban fordulnak elő a különböző operációs rendszerekben. A fájlattribútumoknak két szintje létezik, egyik az operációs rendszeren, másik az alkalmazáson **belüL** Jelen fejezet az alapvető fájl típus okkal, az összes fájltípusra érvényes attribútumokkal, a fájlokkal való műveletekhez szükséges PHP-kódokkal foglalkozik, valamint az egyéb fejezetek-í i hez (például a 19. fejezetben leírt session-ök) szükséges fájlműveletekalapjait tartalmazza.

Minden fájlt lehet mérettel jellemezni, de nem minden fájl tartalmaz megjeleníthető grafikát, így az operációs rendszereknek, illetve az alkalmazásoknak el kell tudniuk igazodni abban, hogy melyik alkalmazás tudja az adott fájlt futtatni. Windows alatt ez a fájlkiterjesztés, azaz az utolsó pont után lévő néhány karakter alapján dől el. Más operációs rendszerek más módszert alkalmaznak annak eldöntésére, hogy az adott fájlhoz milyen alkalmazást társítsanak, de bármely rendszer hibázhat. Ezeket a hibákat használják ki az e-mail-en keresztül terjedő vírusok. Ha egy vírusnak olyan neve van, mint a futtatható fájloknak, néhány levelező program gondolkodás nélkül futtatni is fogja azt.

Könyvtárak

A **PHP** az **opendir(), readdirQ** és **closedirQ** könyvtárkezelő függvényekkel, és a **is_dir()** fájltípustesztelő függvénnyel segíti a könyvárakkal való munkádat. A **PHP dirname()** függvényével elérési utakból fájlokba emelhetsz át könyvtárneveket. A **pathinfoQ** ugyanazt az információt nyújtja, mint a **dirname()** és más függvények együttesen, a **realpath()** a relatív könyvtárlinkeket abszolút elérési úttá változtatja, az **mkdirQ** új könyvtárat hoz létre, az **rmdir()** pedig törli a könyvtárat.

Apache-nézet

Amikor az Apache vagy a használt webszerver megszámlálja a weboldalak könyvtárait, a webszerver a honlapod alapkönyvtárától számlál.
Az én tesztgépem **test.petermoulding.com** honlapja az i:/petermoulding/web/root könyvtárban van. Ezen a könyvtáron belül jelen könyv tesztoldalai a /phpblackbook könyvtárban vannak (az Apache szerint), ezen fejezet tesztoldali és fájljai pedig a phpblackbook /files könyvtárában. A 8. fejezet tesztoldalainak indexére mutató URL a **/phpblackbook/files/index.html.**

PHP-nézet

A PHP közvetlenül az operációs rendszer fájlhozzáférési rutinjaival áll kapcsolatban, így **a** könyvtárakat a szerver a fájlstruktúra elejétől számlálja.
A teszthonlapom az i:/petermoulding/web/root könyvtárban van, jelen fejezet tesztoldalai a /phpblackbook/files könyvtárban vannak, a PHP-fájfüggvényeknek nyújtott elérési út pedig az i:/petermoulding/web/root/phpblackbook/files.

Nézetek váltása

A jelenleg használt könyvtárat a getcwd()-vel találhatod meg, és a **chdir()-rel** válthatsz a könyvtárak között. Nem tudom, hogy ezt ténylegesen hányszor fogod majd megtenni, hiszen a gyakran használt PHP-fájlfüggvények mind elfogadnak elérési utat, így elég csak az elérési úttal dolgozni. Sokkal valószínűbb, hogy könyvtárat akkor váltasz, ha olyan rendszerfüggvényt használsz egy folyamat futásához, amely az elérési út paramétert nem ismeri.

Fájltípusok

Vannak operációs rendszerek, amelyekben a fájltípust a fájlok végződéséből állapítod meg, például a JPEG fájlok .jpeg vagy -jpg végződésűek. Más esetekben az operációs rendszernek külön szabályai vannak. A PHP-ban van olyan függvény, amellyel megállapíthatod, hogy a fájl normál fájl **(is_file()),** könyvtár (is_dir()), fájlra mutató Unix-típusú link **(is_link())** vagy futtatható fájl **(is_executable()).** A **is_readable()** segítségével megállapíthatod, hogy olvasható-e, az is_writeable()-val azt, hogy írható-e, az **is_uploaded_file**Q-val pedig azt, hogy a fájl vajon egy HTTP POST fájlfeltöltés eredménye-e. A l i . fejezetben található függvényekkel ellenőrizheted az image-fájlokat is, és különböző PHP-függvényekkel megnyitva azokat, további speciális fájlokat ellenőrizhetsz. Ne felejtsd el a @ jelet használni a hibák elrejtéséhez, és ellenőrizd, hogy a függvény érvényes eredményt adott-e, vagy a hamis eredménnyel hibát jelez.

Minden operációs rendszernek megvan a saját mechanizmusa a fájl használatának megállapítására. A legtöbb PHP-függvénnyel eredetileg Unix-függvényeket hívtak be, így lehet, hogy azok Unix alatt működnek a legjobban, és a te operációs rendszeredre nem fordítha-tóak le. Először minden egyes függvényt tesztelj le, és ha valamelyik váratlan eredményt ad, minden egyes PHP-frissítésnél teszteld azt újra, mert előfordulhat, hogy valaki belemélyedt a kódba, és megváltoztatta, ahogy a függvény az operációs rendszeredtől kapott információkat értelmezi.

Fájlok megjelenítése

Ha már ismered egy fájl típusát, a legmegfelelőbb mechanizmussal megjelenítheted, és - ha kétségeid vannak - a Gyors megoldások „Adatok megjelenítése" című részében bemutatott speciális hexadecimális megjelenítő használatával megjelenítheted a fájl egy részét. Ebben a fejezetben olyan fájlműveletekről lesz szó, amelyek bármely fájlra alkalmazhatóak. A különleges fájlok, így a képek és adatbázisok egyedi követelményeinek megfelelő függvényeket másik fejezetekben tárgyalom.

Fájlok létrehozása és megváltoztatása

Ha egy fájl szöveg vagy valamilyen nyers bináris formátum, a PHP-fájlfüggvényekkel létrehozhatod vagy megváltoztathatod, azonban bármilyen más formátumú állomány esetén jobb inkább a fájl típusának megfelelő függvényt használni. írhatsz ugyan bináris írásmódot használó saját rutinokat JPEG- vagy PNG-fájlok léthozására, de akkor állandóan változtat -gatnod kell a kódot, hogy az egyedi és a szabványostól eltérő esetekben is megfeleljen.

Ezért hát sokkal jobban jársz a 11. fejezetben bemutatott függvények használatával. Ugyanez igaz az adatbázisok, Flash-fá)lok, levelekhez csatolt állományok vagy bármilyen egyéb egyedi fájlformátum esetén: mindig az adott fájltípushoz készített szoftverrel hozd létre és változtasd a fájlokat.

Fájlok másolása

A copy(tí, *b)* segítségével a-ból b-be másolhatsz egy fájlt, az **unlink(a)** pedig törli a fájlt. A **copy()** és az **unlink()** együttesen lehetővé teszi a fájlok áthelyezését (mivel sajnos a PHP-ben nincsen move()-függvény). Végtelen a vita azzal kapcsolatosan, hogy milyen biztonsági előírásokat kell betartani egy áthelyezett vagy másolt fájl esetén. Mivel az eredményt **a** webszervered operációs rendszere határozza meg, hozz létre az operációs rendszerednek egy tesztoldalt, hogy a pontos eredményt megtekintsd.

Ideiglenes fájlok

Az ideiglenes fájl *{temporary)* egy egyedi típusú fájl, mert soha nem kell a nevével vagy **a** törlésével foglalkoznod. Csak akkor létezik, amikor megnyitod, ha bezárod, törlődik, olyan könyvtárban van, amelyről csak az operációs rendszernek kell tudnia, és olyan neve van, melyet soha nem kell látnod. A **tmpfile()** pontosan úgy nyit meg egy ideiglenes fájlt, mím az **fopen(),** pontosan úgy ad vissza egy állományazonosítót, mint az **fopen(),** és pont úgy záródik be a fclose()-tól, mint az **fopen().** Ugyanúgy végezhetsz vele PHP-fájlbeoivasást, írást és más fájlkezelő műveletet. Az egyetlen különbség a fájl élettartamában van. Őszintén szólva a várható élettartama pont annyi, mint egy pizzának a Sydney User Group (PHPSydney.com) ülésén. A **tempnam()** az ideiglenes fájl nevét adja eredményül, így saját nevet adhatsz neki, de mi történik, ha két szknpt verseng ugyanazért az ideiglenes névért? A PHP a 4.0.3-tól ezt a problémát az elhelyezési fájl létrehozásával oldja meg, ami azt jelenti, hogy azonnal törölheted vagy újból használhatod az ideiglenes nevet, vagy ami logi-kusabb, az elejétől használhatod a **tmpfile()-t.**

Fájtok feltöltése

A HTTP-vel végezhető fájlfeköltés, és a feltöltött fájl a php.ini-ben meghatározott ideiglenes könyvtárba kerül. A feltöltés méretét az **upload_max_filesize** specifikációban lehet korlátozni. íme a php.ini erre vonatkozó része:

```
file uploads = On
upload_tmp_dir = t:\upload
upload__max_f ilesize = 8M
```

Mikor a szkripted megkapja a fájlfeltöltési információt, ellenőrzésképpen le kell a fájlt másolnia és állandó tárhelyre áthelyezni, különben a fájl elvész. A PHP legutolsó verziójában található move_uploaded_file()-függvénnyel áthelyezheted a feltöltött fájlt, amelyre a Gyors megoldások „Fájlok feltöltése" című részben találsz példát.

Fájllista-cache

A clearstatcache() olyan különleges függvény, amelyre akkor van szükséged, amikor a fájlkönyvtárakat és az olyan fájlattribútumokat, mint például a méret, újra beolvasod, hiszen mindezek a fájlattribútumok a cache-ben mentődnek el, és a lemez helyett onnan kerülnek beolvasásra. Az operációs rendszertől függően a cache percekig vagy akár napokig is tárolja az eredményeket, amely általában túl hosszú a scriptnek, amely felfrissíti a fájlt, majd azonnal az új fájlhosszt próbálja beolvasni. Ha írsz egy fájlhoz, és utána megpróbálod beolvasni a fájlattribútumokat, végre kell hajtanod az írást, lefuttatnod a clearstatcache()-t, majd utána beolvasni a fájlattribútumokat.

Engedélyezés/Jogosultságok

Számos operációs rendszerben és fájlrendszerben a hozzáférés-kezelést leegyszerűsítheted a fájlok csoportokhoz való hozzárendelésével, majd az egyéni felhasználók csoportokba sorolásával. A legkifinomultabb rendszerekben egyidejűleg vannak jelen mind az erőforrás-, mind pedig a felhasználói csoportok, így az ilyen rendszerek által kínált hozzáférés-kezelés egy jól strukturált relációs adatbázishoz hasonlatos. A PHP olyan biztonsági függvényei, mint a chgrp(), közvetlen kapcsolatot jelentenek a Unix és Unix-féle operációs rendszerek C-függvényeihez, így nem tartalmazzák a hozzáférés-kezelés kifinomultabb pontjait, és csak Unix vagy Linux alatt működnek. Talán hosszú távon a PHP-függvények is képessé válnak arra, hogy más operációs rendszereken is használják az ekvivalens függvényeket, vagy ezek az operációs rendszerek képesek lesznek a Unixot a PHP-val való kommunikációban felülmúlni.

A chmod() segítségével PHP-ból a Unix chmod műveletét végrehajtva megváltoztathatod a fájlhoz való hozzáférést. Ehhez a legfontosabb helyiértékre nullát kell tenned, hogy oktá-lisnak értelmezze a módot. Ha nem konyítasz a módértékekhez, fogj egy jó könyvet a Unixről vagy Linuxról, és olvasd el a chmod-ról szóló részt. A chown() segítségével PHP-ból a Unix chown-utasítást végrehajtva megváltoztathatod a fájl tulajdonosát, és az csak akkor fog működni, ha megfelelően magas szintű hozzáférésed van a fájlhoz. A chgrp() az egyik felhasználói csoportból egy másikba helyezi át a fájlt. Mivel a Unixnál nincsenek forráscsoportok, egy fájlcsoport megváltoztatása a chgrp()-t egyszerre egy fájlon használva unalmassá válhat. Javaslom, készíts egy olyan adminisztrációs oldalt, amellyel fájlok vagy könyvtárnevek listáját tömegesen tudod az egyik csoportból a másikba helyezni. Ezen oldal kódja a listában vagy egy könyvtárban található összes fájlra alkalmazza a chgrp()-t.

Az umask() beállít egy alapértelmezett engedélyezési mód maszkot, amely minden olyan engedélyezési sztringre vonatkozik, amely logikai ES-t (AND) használ új fájlokra. (A logikai ÉS-t bitenként alkalmazzák, és akkor eredményez 0-t, ha az argumentumok valamelyike 0, és akkor 1-et, ha minden argumentum 1.) Az umask(OOl) a 0777-es engedélyezési módot 0766-ra konvertálja, ami gyakorlatilag azt jelenti, hogy csak a fájl tulajdonosának van végrehajtási hozzáférése a fájlhoz. A hozzám hasonló egyszerű embereknek jó néhányszor el kell olvasni egy Linux kézikönyvet ahhoz, hogy megjegyezzék, hogyan működik a chmod(). Ezért én inkább maradok a GUI-alapú alkalmazásoknál, amelyekben egyszerűen csak be kell pipálni, hogy milyen jogosultságokat akarsz adni.

Gyors megoldások

Könyvtárak listázása

Ez a megoldás azt mutatja meg, hogyan kell a könyvtár struktúrában a fájlokon és könyvtárakon lépkedni, hogyan kell a kódot különböző célokra megváltoztatni, valamint hogy ez a kód az alapja a következő Gyors megoldások némelyikének. Az ismétlés elkerülése érdekében az egyszer bemutatott kódot és függvényt nem ismétlem meg még egyszer, hanem az adott megoldásnál a korábban bemutatott kódra, illetve függvényre hivatkozom.

Közös kód

Jelen fejezet összes gyors megoldásában feltételezem, hogy a fájlok egy tesztkönyvtárban vannak, amelyet te állítasz be, és a $path-változót arra használom, hogy az alapkönyvtár elérési útját tartalmazza. Az én tesztgépemen a **test.petermoulding.com** teszthonlap a i:/petermoulding/web/root könytárban található, és a fejezet tesztoldalai ezen belül a /phpblackbook/files könyvtárban vannak. így az operációs rendszer és a **PHP**-fájlfüggvények számára az összes fájl az i:/petermoulding/web/root/phpblackbook/files címen érhető el:

```
$path = "i:/petermoulding/web/root/phpblackbook/files";
```


Az is_dir() megközelítés

A következő kód a Spath-ban elnevezett könyvtárból olvassa be a könyvtárstruktúrát, megszámolja a talált könyvtárakat és fájlokat, majd megjeleníti az összefoglalást. Ez a legegyszerűbb kód, amely bemutatja az **opendirQ, readdir()** és **closedir()** könyvtárkezelő-és az **is_dir(), is_file()** és **is_link()** fájltípustesztelő függvényeket. Az első sorokban számlálókat hozunk létre, a következő utasítással egy könyvtárat nyitunk meg, majd a kód egy ciklussal végigmegy a könyvtáron, ellenőrizve annak tartalmát. A . és .. nem számítanak be, mert azok az aktuális szülőkönyvtárra való hivatkozások:

```
$directories_found = 0;
$files_found = 0;
$links_found = 0;
$others_found = 0;
$path_id = opendir($path);
while($file_name = readdir{$path_id))
    if ($filename          and $filename !=     .")

        if (is__dir ($path          $f
                                   ilename)
```

```
            $directories_found++;
            } elseif(is_file<$path
        . "/"                          $filename)

            $files_found++; }
        elseif(is_link($path . "/"
            í $links_found++;          $file name))

        else

            $others found++;


closedir($path_id);
print("<br>Directories found: " . $directories_found);
print("<br>Files found: " . $files_found);
print("<br>Links found: " . $links found);
print("<br>Others found: " . $others_found);
```

Lent látható az eredmény. Remélem, azokban a rendszerekben, amelyekben ezt a kódot használod, nem lesznek Others-típusok. Csak azért hagytam az **othert** benne, mert az olyan operációs rendszerek, mint például a BeOS, vadonatúj fájltípusokat tartalmaznak. A kódot a többszörös if()-utasítások switch()-csel, valamint a négy változó tömbbel való helyettesítésével kisebbé és szorosabbá teheted, de ezek a változtatások az PHP-t először használó embereket összezavarhatják:

```
Directories found: 2
Files found: 3 Links
found: 0 Others found:
0
```

A filetype() megközelítés

A következő példában az **is_dir()** és az ehhez hasonló függvényeket az egyszerű **filetype()**-függvénnyel helyettesítettem, amely sztringként adja vissza a fájl típusát, sztring a **dir, fiié** vagy **link** egyikét tartalmazza. Ez alkalmassá teszi arra, hogy a fájltípusokat egy asszociatív tömbben tároljuk, amely a filetype() eredményeként megjelenő érték alapján kulcsindexelt. Ezután egy ciklussal megjeleníthetjük a talált fájltípusok listáját. Habár ez a kód ideális arra, hogy az egyes fájltípusokon ugyanazt a műveletet hajtsuk végre, az *előző* kóddal különböző műveletet hajthattunk végre minden fájltípuson az egyes if()-utasításokon belül:

```
$path_id = opendir($path);
while($file_name = readdir($path_id))
    {
    if($file_name != "." and $file_name != "..")
        {
        $file_type = filetype($path . "/" . $file_name);
```

4.1

```
        if (!isset($found[$file_type]))
            {
            $found[$file_type] = 0;
            }
        $found[$file_type]++;


closedir($path_id);
reset($found);
while (Üst ($k, $v) = each ($found) )
    {
    print("<br>Found " . $k . ": " . $v)
```

Alább látható a kód eredménye, amely nem túl sokatmondó. Vagyis olyan, amit egy adminisztrációs vezérlőpanelben szívesen használnál, nyilvános alkalmazásban viszont már nem. Ha végigolvasod a 3. fejezet tömbös példáit, rájössz, hogyan használhatod ezeket a rövid neveket indexként egy olyan tömbhöz, amely már leíróbb jellegű neveket tartalmaz.

```
Found file: 5
Found dir: 2
```


A get_directory_file() megközelítés

Az előző példák egy könyvtár egyetlen szintjén mentek végig. Gyakrabban fordul elő azonban, hogy azt szeretnéd, hogy a kód a kiválasztott könyvtár alkönyvtáraiban is végigmenjen a fájlokon, így a következő kód a korlátozás nélküli, többszintű feldolgozásra mutat példát. Az előző megoldások könyvtárolvasó kódja egy get_directory_file() nevű függvénybe volt beágyazva, amely függvény a könyvtár elérési útját kéri. Van egy kis különbség e között és a között a példa között, mely a fájlattribútumokat találja meg. Az egyetlen eredményként megjelenő fájlattribútum - a kód pontosságának bizonyítására - a fájltípus, de bármilyen szükséges fájlattribútumot hozzáadhatsz, így a fájlméretet vagy bármely, a 11. fejezetben leírt képattribútumot. Néhány későbbi példában arra használom ezt a függvényt, hogy csupán könyvtár- és fájlneveket gyűjtsön, majd az e függvény által létrehozott tömb elemei alapján gyűjtsön további fájlattribútumokat.

Ahogy a kód végigmegy az alkönyvtárakon, egyre hosszabb és hosszabb elérési utakat hoz létre, ezért a kód első része eltávolítja az utolsó / jelet az elérési útból. A következő kódrész azután visszafűzi a perjelet, amikor a fájlnév hozzáadódik az elérési úthoz. A perjel eltávolítása olvashatóbbá teszi a könyvtár elérési útját, amikor a táblázatban van tárolva, illetve bizonyos függvényekben így kell az elérési utat megjeleníteni, amikor azt könyvtárnévként használjuk. Amikor a get_directory_file() úgy látja, hogy egy fájl a könyvtár, az alkönyvtár feldolgozására a get_directory_file() behívja önmagát, és az array_merge() függvény összefésüli az alkönyvtárból érkező tömböt a $found elsődleges tömbbel. Ha a függvény üresnek találja a könyvtárat, nem *hozza*, létre a $found-ot, így a kód utolsó sorai egy üres $found-változót hoznak létre, hogy a return utasítással visszatérjen:

```
function get_directory_file(Spath)
   {
   $path_id = opendir($path);
   while($file_name = readdir($path_id))
       { I      if($file name != "." and $file name !=
"..")
f          i    ~
           $file_type = filetype($path  . "/" . $file_name);
           $found[$path][$file_name] =  $file_type;
           if($file type == "dir"}
               <
*              $file_array = get_directory_file($path . "/" . $file_name);
               $found = array_merge($found, $file_array);


   closedir($path_id);
   if ( !isset($found))
       {
       $found   =   arrayO;
       }
   return($found);
```

Formázott fájllista

A hátralévő' kódpélda a get_directory_file()-függvény eredményét veszi, és a 8.1 ábrán látható formázott fájllistát jeleníti meg. A kód első sora meghívja a **get_directory_file()-t** a gyökérkönyvtárból kiinduló elérési úttal, majd a könyvtár és fájl szerint kulcsindexelt $found-tömböt kapja vissza, végül az ábrán látható táblázatban megjeleníti az eredményt. Mivel a **Sfound** kétszintű indexeléssel rendelkezik, két whileQ-ciklus van, az első a könyvtárakon, a második a fájlokon fut végig:

Dtrectory	*File*	*[Type*
i: /petennouldinsAveb/root/phpblackbook/files/	images	**dir**
i:/petemioulding/web/root/phpblackbook/files/	index, html	**(file**
i:/petennoulding/web/root/phpblackbook/files/	listingdirectories.htm	i
!i:/petermoulding/web/root/phpblackbook/ílles/	**listingfiles.html**	**file**
ir/petermoulding'web/root/phpblackbook/files/	test	**[dir**
i:. petennouldingí web/root phpblackbooky'files/images	london.gif	**[file**
|i:/petermoulding/web/root/phpblackbook/files/images	peters.jpg	**Ifile**
i:/petermoulding\veb/root/phpblackbook/files/images	php4-small.gif	**|file**
;i:/petermouldingweb/root'phpblackbook,/files/images	santaclara.gif	{file

8.1 ábra Könyvtárak listája

```
$found = get_directory_file($path);
reset($found) ;
print("<table bordér=\"3\"><tr><td><em>Directory</em></td>"
. "<td><em>File</em></td><td><em>Type</em></tdx/tr>") ; while
(üst ($d, $dv) = each ($found) ) {
    if(is_array($dv)) {
        ~
        while (üst <$f, $fv) = each($dv)) {
            print ("<trxtd>" . $d . "</tdxtd>" . $f . "</td>
                        . $fv . "</tdx/tr>") ;


    else
        {
        print ("<trxtd>" . $d . "</td><td>" . $dv . " .
            "<td> </tdx/tr>") ;


print("</table>");
```

Van jónéhány formázási lehetőséged: végigmehetsz a könyvtárakon és létrehozhatsz egy csak alkönyvtárakból álló listát, vagy kiemelheted az egyes alkönyvtárak elején álló első könyvtárak nevét. Az egyetlen szokatlan elem a **test** alkönyvtár, egy olyan könyvtár, amelyet a főkönyvtár fájllistája tartalmaz, de a könyvtárlistában nem szerepel könyvtárként, mert a test-nek nincsen tartalma, nincsenek fájljai vagy alkönyvtárai.

El kell döntened, hogyan kezeled az üres alkönyvtárakat. Listázhatod úgy a könyvtárakat mint a könyvtárkulcsban lévő bejegyzések összességét, vagy listázhatod az állományokat egy könyvtártípussal is. Bármilyen könyvtárak megtalálására megvannak az eszközeid, a 9. fejezetben bemutatott könyvtárkereső űrlapot is használhatod, és a következő kóddal - a keresést elősegítendő - a fájlokra jellemző további attribútumokhoz is hozzáférsz.

Egyéb könyvtárfüggvények

A dirnameQ-függvénybe a fájl elérési útját írva az elérési út könyvtárrészét kapjuk eredményül, míg a basename() a maradékot, a fájlnevet eredményezi. A következő három gyors példa ezen függvények legjobb és legrosszabb tulajdonságait mutatja be. A **$path** használó teszt pontosan úgy működik, ahogy azt várnád:

```
print("<br>Dirname: " . dirname(Spath . "/index.html") ) ;
```

Az eredmény:

```
Dirname: i:/petermoulding/web/root/phpblackbook/files
```

A következő kód

```
print("<br>Basename:       basename($path . "/index.html"));
```

eredménye:

```
Basename: index.html
```

Az /a/b-vel lefolytatott teszt azt mutatja, hogy a b-t fájlnévként értelmezi, holott a b lehet könyvtár- és fájlnév is, és az URL-ekben a b valószínűleg inkább könyvtár, mint fájl:

```
print("<br>Dirname:      "      .
     dirname("/a/b") .     ",    basename:     "     .
     basename("/a/b") ) ;
```

Az eredmény:

```
Dirname:     /a,    basename:    b
```

A harmadik /a/b/-teszt meglepő, mert itt a b könyvtár helyett fájl. Próbáld ki a honlapodon ezt a tesztet a PHP-val:

```
print("<br>Dirname:      "      .    dirname("/a/b/"
     ) .     ",     basename:     "     .
     basename("/a/b/") ) ;
```

Az eredmény:

```
Dirname:     /a,     basename:     b
```

A pathinfoQ ugyanazt az információt adja, mint a **dirname()** és a **basename()** összekapcsolva. Az első példa ugyanaz, mint az *előző* rész első példája. A **$path . "/index.html"-t** egy **whileQ** -ciklus követi, hogy az eredményül kapott tömböt megjelenítse:

```
$info    =    pathinfo($path    .
"/index.html"); while (üst ($k,      $v)      =
each($info))
     {
     print("<br>Pathinfo:     k:     "     .     $k     .     ",     v:     "     .     $v     ) ;
```

Az eredmény ugyanaz a basename-re és dirname-re, mint az *előző* részben, illetve megjelenik a fájl kiterjesztése is:

```
Pathinfo: k: dirname, v: i:/petermoulding/web/root/phpblackbook/files
Pathinfo: k: basename, v: index.html Pathinfo: k: extension, v: html
```

A második példa az /a/b ugyanolyan értelmezését adja, természetesen itt nincsen kiterjesztés:

```
$info = pathinfo("/a/b");
```

Ennek eredménye:

```
Pathinfo:     k:     dirname,     v:
/a Pathinfo:     k:     basename,
v:     b
```

A harmadik teszt megpróbál egy URL-t összetevőire bontani, de mint látjuk, nem sikerül a lekérdező sztringet kezelnie: ,

```
$info     =
   pathinfo("http://www4.ncdc.noaa.gov/cgi-win/wwcgi.dll?wwAW~MP~PUB");
```

Az eredmény így néz ki:

```
Pathinfo:     k:     dirname,     v:
http://www4.ncdc.noaa.gov/cgi-win Pathinfo:     k:     basename,     v:
wwcgi.dll?wwAW~MP~PUB Pathinfo:     k:     extension,     v:
dll?wwAW~MP~PUB
```

A **realpath()** a relatív könyvtárlinket abszolút elérési úttá konvertálja, amelyet akkor használsz, amikor egy URL-ből származó fájlnevet egy PHP-függvénybe akarsz beírni. A következő' példa egy oldallekérésből származó relatív fájlnevet, a **./index.html-et** abszolút fájlnévvé konvertálja, amely alkalmas a PHP-függvényekben való használatra:

```
print("<br>Realpath: " . realpath("./index.html"));
```

Az eredmény:

```
Realpath: i:\petermoulding\web\root\phpblackbook\files\index.html
```

Könyvtárak létrehozása és törlése

Az **mkdir()** könyvtárnevet vagy teljes elérési utat fogad el, új könyvtárat hoz létre, sikeres művelet esetén igaz, hiba esetén hamis eredményt adva vissza. Az **rmdir()** könyvtárnevet vagy teljes elérési utat fogad el, törli a könyvtárat, sikeres művelet esetén igaz, hiba esetén hamis eredményt adva vissza. íme egy egyszerű példa a két függvényre:

```
$new = "./anothertest";
if(mkdir($new))

    print("<br>:-) Created directory: " . $new); if
    (rmdir() )      /^

    print("<br>:-) Removed directory: " . $new); } else
        {
        print("<br>:-( Failed to remove directory: " . $new);
        }

else
    {
    print("<br>:-( Failed to created directory: " . $new);
```

Az eredmény a következő:

```
Warning: Wrong paraméter count for mkdir()
:-( Failed to created directory: ./anothertest
{Figyelem: Rossz paraméter megadása az mkdir() számára. Az
anothertest könyvtár létrehozása sikertelen.)
```

Figyeld meg, hogy az **mkdir()** nem működött, mert nem tartalmazta az opcionális második paramétert, a]ogosultsági beállítást, amelyet a Unixszal használnak, de egyéb operációs rendszerekben figyelmen kívül hagynak. Windows NT alatt a PHP 4.0.5-ben az **mkdirQ** igényli a második paramétert, de amennyire én tudom, a Unix-stílusú jogosultsági beállításokat nem fordítja le Windows NT-s megfelelőjükre. Ha az effajta kódot webszervereken való éles használat céljából írod, gondoskodnod kell arról, hogy az bármely PHP-változatta használható legyen. Ezért a második paramétert vagy változóként kell megadnod, aminek

értéke a szknpt elején beállítható, vagy pedig egyetlen közös beágyazott elemként kell használnod, ami a honlap összes szkriptjében megtalálható, és megadható neki egy olyan alapértelmezett beállítás, ami az összes operációs rendszerrel működni fog. A következő példa a 0777 általános beállítást tartalmazza, amely bárki részére hozzáférést biztosít. Csak azt az egyetlen kódsort mutatom, amelyet az előzőhöz képest megváltoztattam:

```
if(mkdir(Snew,     0777))
```

Az eredmény:

```
:-) Created directory:   ./anothertest
:-) Removed directory:   ./anothertest
```

Mindkettő **nobody** felhasználóként fut a webszerveren (hacsak a szerver alapértelmezett beállításait nem változtatták meg), így nemigen van arra jogosultságuk, hogy könyvtárakat hozzanak létre és töröljenek. Ez problémákat vet fel ha olyan weboldalt próbálsz írni, amely az új felhasználók számára új saját könyvtárat hoz létre. A legjobb megoldás az, ha a PHP-t Apache-modulként futtatod, biztonsági igazolást és SSL-t telepítesz, az összes megfelelő Apache biztonsági opciót alkalmazod felhasználói információk könyvtárban (pl. LDAP) vagy adatbázisban (pl. MySQL) való tárolásnál, majd az adminisztrációs oldaladon a bejelentkezéshez felhasználói nevet és jelszót kérsz. Kevés felhasználó esetén a leggyorsabb megoldás az Apache .htaccess- és .htpassword-fájljainak használata. Ha a honlapodhoz való hozzáférés biztonságos, lazíthatsz a fájlhozzáférési engedélyek rendszerén. Bármelyik rendszert választo~d~is, szükséged lesz egy az Apache-ról szóló jó könyvre, mint a *Apache Server for Windows Little Black Book* (Greg Holdén, The Coriolis Group, Inc.), és egy az operációs rendszer adminisztrációjáról szól, mint a *Linux System Administration Black Book* (Dee-Ann LeBlanc, The Coriolis Group, Inc.).

Fájlok listázása az attribútumokkal együtt

Ez a megoldás azt mutatja meg, hogyan fuss egy könyvtárstruktúrában végig a fájlokon a fájlattribútumok kiíratásával. Ez a megoldás „Könyvtárak listázása" című előző rész kódján alapul.

Általános kód

Az első lépés annak eldöntése, honnan akarod a kimutatást kezdeni. Annak érdekében, hogy mind a feldolgozás idejét, mind a kimutatás méretét kordában tartsuk, legyen a **$path-ban** lévő kiindulási pont egy kevés fájlt tartalmazó alkönyvtár. Egy a 9. fejezetben megmutatott űrlap segítségével megadhatod a kiindulási könyvtárnevet:

```
$path    =    "i:/petermoulding/web/root/phpblackbook/files" ;
```

i A get_directory_file() kiterjesztése

Az *e\Ő7.6* részben használt **get_directory_file()-függvényt** kiterjesztjük, hogy több, ese-

tünkben a hagyományos PHP-függvényekkel elérhető összes fájlattribútumot adja eredményül. A megváltoztatott szakaszt kiemeltem. Ez tartalmazza a fileatime()-függvényt, amely azt mutatja meg, hogy mikor volt a fájlhoz történő legutolsó hozzáférés (feltéve, hogy az operációs rendszer és a fájlrendszer lehetővé teszi ezt). A filemtime() azt mutatja meg, hogy mikor változott meg legutoljára a fájl mérete, a fllectime() pedig azt, hogy mikor frissítették legutoljára a fájl valamelyik rekordját. A fileowner()-, fileperms()-, filegroup()- és fileinode()-függvények ezeket az információkat Linux alatt nyújtják, viszont Windows NT alatt nem adják eredményül a megfelelő Windows NT fájlattribútumokat:

```
function
    get_directory_file($path) {
    $path__id    =    opendir ($path) ;
    while( $ f ile_name    =    readdir($path_id))

        if ( $ f ilename              and    $filename
        !=                          ! =

            $file["type"] = filetype($path . "/" . $file_name);
            if($file["type"] == "dir")

               $file_array = get_directory_file($path. "/"
                   . $file_jxame) ;
             ~'"if (isset ($found) )

                   $found = array mérge($found, $file array);

               else

                   $found = $file array;


            else
                {
               $file["accessed"] = fileatime($path  . "/" . $file name);
               $file ["changed"] = filectime($path  "/" . $file name!; /" .
               $file["group"] = filegroup($path .   $file name); /" . $file
               $file["inode"] = fileinode($path .   name); . "/" . $file
               $file["modified"] = filemtime($path . name); . $file name);
               $file["owner"] = fileowner(Spath . "/"  "/" . $file name);
               $file["permissions"] = fileperms($path $file name);
               $file["size"] = filesize($path . "/"
               $found[$path][$file_namel = $file;



    closedir ($path_id)-;
    if ( !isset($found))
        í
    $found = array(); }
    return($found);
```

Formázott fájllista

A következő kódrész a get_directory_file()-függvénnyel az összes könyvtárat és fájlnevet beolvasva, majd az eredményül kapott tömbből, a $found-ból ciklussal táblázatba kiíratva a 8.2 és 8.3 ábrán látható fájl- és attribútumlistát eredményezi. Az első printutasítás az oszlopfejlécet, a második az egyező mezőket, a harmadik — abban a meglehetősen valószínűtlen esetben, ha a fájlbejegyzés nem egy attribútumokból álló tömb - csupán a könyvtár - és fájlneveket, a negyedik - abban a szintén valószínűtlen esetben, ha a könyvtárbejegyzés nem tartalmaz fájlbejegyzéseket - a könyvtárakat jeleníti meg, míg az utolsó a HTML-táblázat tag-et zárja le. A könyvtár- és fájlnevek normál tag-ként jelennek meg, a dátumokat pedig a 2. fejezetben bemutatott dateQ-függvénnyel formáztam.

Directory	*File*	*Type*	*Ac*
i:/petermoulding/web/root/phpblackbook/files/images london.gif		file	12001-05-17 11:27:12]
i:/petennouldingweb/rootphpblackbookyfiles/images peters.jpg		file"	[2001-05-17 1Í727:12;
i:/petennoulding/web/root/phpblackbook/files/images	jphp4-small.gif	ifile	2001-05-
i:7petermouÍding/web/roofphpb 1 ackbook/fiíes/images	santaclara.gif	jfile	2001-05-17 13:29:42
i:/peteniioulding web/rootphpblackbook/files/	index.html	Ifile	2001-05-17 13:15:29i
f | i **:/petermoul** dingv w eb/rBr6t7plq)bÍ ackb o okj'fi 1 es/	listingdirectories.html	ifile	
i:'petennoulding/web/root/phpblackbook/files/	listingfiles.html	Ifile	12001-05-17 13:28:25

8.2 ábra Fájlok és attribútumaik listája, A rész

Changsd	*Group*	*Inode*	*\Modified*	*Owner\Permissions\Size*		
2001-05-17 11:27:12		**?**	2000-09-11 21:16:41	**0**	33206	**21940!**
2001-05-1711:27:12	0	0	2001-03-16 09:46:13	**o**	**33206**	**6530** ;
2001-05-17 11:27:12	0	0	2001-01-30 04:36:00	**0**	33206	**4528**
2001-05-17 11:27:12	0	0	**2001-03-15 21:50:59**	**0**	**33206**	**2543** Oi
2001-05-17 07:14:39	0	0	2001-05-17 12:22:29	0	**33206**	**483**
2001-05-17 07:15:32	0	0	**2001-05-17 13:15:29**	0	**33206**	**5365]**
2001-05-17 12:03:44	0	0	2001-05-17 13:28:25	0	**33206**	**5342** |

8.3 **ábra** Fájlok és attribútumaik listája, B rész

Az első while()-ciklus könyvtárszinten fut végig a **$found-on,** a második **while()**-ciklus pedig a fájlok szintjén fut végig a könyvtárak tartalmán. Arra az esetre, ha a get_directory_iile()-függvény fájl nélküli könyvtárat vagy attribútum nélküli fájlt eredményezne, beraktam az if(is_array())-tesztet. Jóllehet még soha nem találkoztam attribútum nélküli fájllal, fájl nélküli könyvtárral néhányszor már igen:

```
$found = get_directory_file($path, "");
print ("<table border=\"3\"XtrXtdXem>Directory</emX/td>"
    . "<tdXem>File</emX/tdxtdXem>Type</emx/td>" .
    "<td><em>Accessed</emx/tdxtd><em>Changed</em></td>" .
    "<td><em>Group</em></td><td><em>Inode</emx/td>" .
    "<td><em>Modif ied</em></td><td><em>Owner</emX/td>" .
    "<tdXem>Permissions</emx/tdxtdxem>Size</emx/td>"

while (üst ($d, $dv) = each ($found) )
```

```
if(is    array($dv)) {  ~whiledist < $f,
    $fv)    =    each($dv))
        {
        if(is_array($fv))

                                                        $f
            print("<tr><td>" . $d . "</td><td> . "<td>" .
                $fv["type"] . "</td>" . "<td>" . date("Y-m-d",
                $fv["accessed"]) . "snbsp;" . date("H:i:s",
                $fv["accessed"]) . "<td>" . date("Y-m-d",
                $fv["changed"]) . " " . date("H:i:s",
                $fv["changed"]) >" . $fv["group"] .
                          $fv["inode"] .
                  "<td>" date("Y-m-d", $fv["modified"])
                  "Snbsp;     . date("H:i:s", $fv["modified"])
                . "<td>"$fv["owner"] .
                . "<td>"          $fv["permissións"]
                . "<td>"$fv["size"] .
            }           else
             {
             print ("<trxtd>"   .   $d   .   "</tdxtd>"    .   $f   .
             "</td>"
            .  "<td>"   -   $fv   .   "</td><td> /</tdxtd> </td>"
                 .    "<td>  </tdxtd>Snbsp; </tdxtd>  </td>"
               .   "<td> </tdxtd> </tdxtd> </tdx/tr>"    );


    else
        {
        print ("<trxtd>"    .    $d    .    "</td><td>"    .    $dv    .
            "</td>" .    "<td> 
            </tdxtd> </tdxtd>  </td>" .
            "<td> </tdxtd> </tdxtd> </td>" .
            "<td> </tdxtd> </td><td> 


print("</table>");
```

További attribútumok

A következőkben bemutatott függvényekkel bizonyos operációs rendszerekben további attribútumokhoz juthatsz, de nem szabad elfelejtened, hogy az ezen függvények által adott attribútumok jelentése is operációs rendszerenként eltérő lehet - különösen ami a futtatható fájlokat illeti. Figyeld meg, hogy a **is_writeable()** az **is_writable()** aliasa:

```
$file["executable"]    =    is_executable($path    .    "/"    .
$file_name); $file["readable"]    =    is_readable($path    .    "/"    .
$file_name); $file["writeable"]    =    iswriteable($path    .    "/"
.    $file_name); $file["writeable"]    =    is_writable($path    .
"/"    -    $file_name);
```

Most már bármilyen attribútumról készíthetsz kimutatást, amennyiben az operációs rendszered biztosítja az információt, a PHP-dben pedig megvan az ehhez szükséges függvény. A PHP4-et nem nehéz új függvényekkel kibővíteni, így bárki, aki ért a C-hez, további függvényekkel további információt szerezhet be az operációs rendszertől.

Lemezterület-kimutatás

Amikor egy bonyolult, erőforrás-igényes fájlt alkotsz, nem örülnél neki, ha az az utolsó pillanatban a lemezterület hiánya miatt fuccsolna be. Jobb, ha előre ellenőrződ a rendelkezésre álló lemezterületet, és csak azután kezdesz neki a feladatnak, ha már eltakarítottad a szemetet. Az itt bemutatott kóddal könnyen megtalálod a legnagyobb könyvtárakat és fájlokat, és egyszerűen átalakíthatod, hogy keresse meg a legrégebbi fájlt, valamint törölje ki a nem használt fájlokat.

Az operációs rendszered nyomon követ bizonyos fájlinformációkat, és lehet, hogy feljegyzi azt is, hogy egy adott fájlt mikor használtak a legutolsó alkalommal. Az operációs rendszered a fájlban lévő bájtok számáról is készít kimutatást, ami nem feltétlenül egyezik meg a fájl által ténylegesen elfoglalt bájtok számával, így a PHP által közölt eredmények csak útmutatók, amelyeket az operációs rendszerednek vagy a fájlrendszerednek megfelelően ki kell igazítanod. Vegyük a következő példát. A Linux ext2 fájlrendszere a fájlokat a 8,192 bájt (8KB) többszörösébe osztja ki, így 1,000 darab 200 bájtos fájl 8 a 200,000 helyett 8,192,000 (8MB) területet foglal el. A változó allokációjú fájlrendszerek, így a Reiser vagy a NTFS kevesebb területet használ a kisebb fájlokra, de még így is az 512 bájtos lemezszektor többszörösére kerekít fel. Minden fájlrendszer többletráfordításokat használ fel a könyvtárakra, a biztonsági és naplózási információkra. A legtöbb fájlrendszer feljegyzi, hogy a fájlt mikor hozták létre, de a jó fájlkezeléshez arra is szükség van, hogy mikor olvasták be a fájlt legutoljára, a többletráfordítások miatt ezt a dátumot viszont csak néhány fájlrendszer jegyzi fel.

Általános kód

A kódnak csak egy kiindulási pontra van szüksége, hogy hol kezdjen el írni a lemezre. Ahogy az a $path-utasításban látható, én a rendszerpartíciót a c:-munkaállomásomon kezdem el, mert a c: számtalan kisebb és nagyobb fájlt és könyvtárat tartalmaz. A kimutatáshoz egyszerűen adhatsz partíciótáblát és további meghajtókat, de nem árt tudni, hogy bizonyos fájlfüggvények nem működnek a hálózatban lévő meghajtókon:

```
$path = "c:";
```

A get_directory_file() kiterjesztése

A példában az előzőkben használt get_directory_file()-függvényt bővítjük ki a fájlattribútumokkal. Azért, hogy ne kelljen a függvény teljes leírását elolvasnod, az input a $path, az output pedig egy a könyvtár-, majd fájlnevekkel indexelt tömb lesz, amely tar-

talmazza a fájlméreteket, ezt a példa későbbi részében még használjuk, az utolsó beolvasás idejét, amit szintén használni fogsz, valamint további attribútumokat, amelyekre nincsen szükség.

Üres lemezterület

Mennyi szabad terület van még a partíción? A következő' példában a PHP diskfreespace()-függvényének használatával megjelenítjük, hogy mennyi lemezterület van még a partíción. A 8.4 ábra néhány eredményt mutat. Az első sor a partíció gyökérkönyvtárából visszakapott nyers számot mutatja, a második ugyanezt az értéket number_format()-tal formázva, a harmadik pedig a szabad lemezterületet mutatja, ha a dískfreespace() egy alkönyvtárba mutat - ez az eredmény persze nem más, mintha a főkönyvtárba mutatna, hiszen a szabad lemezterület az egész partícióra vonatkozik:

diskfreespace("/")	'1277777920
(Disk free space for /	1,277,777,920:
e space for /mysql/bin	l,277,777,920

8.4 ábra Szabad lemezterület

```
print("<table border=\"3\">");
print(trow(tdl("diskfreespace(\"/\")")  -  tdl(diskfreespace("/"))));
print (trow (tdl ("diskf reespace (\"A") ") . tdn (diskf reespace ("/"))));
print(trow(tdl("diskfreespace(\"/mysql/bin\")")
    . tdn(diskfreespace("/mysql/bin")))) ;
print("</table>");
```

Táblázatfüggvények

A „Gyors megoldások" megjelenítő kódrészleteiben a HTML táblázattag-ek begépelésének csökkentésére használd az itt látható egyszerű formázófüggvényeket. A tdd() egy táblázat egy cellájában jeleníti meg a dátumot, a tde() kiemelt szöveget jelenít meg a fejrész egy cellájában, a tdl() balra igazított szöveget jelenít meg a cellában, a tdn() egy formázott és jobbra igazított számot jelenít meg egy cellában, a tdreQ egy jobbra igazított, kiemelt szöveget jelenít meg egy cellában, a trow() pedig cellák sora köré töri be a táblázatsortag-eket. Ha a táblázat egy cellájában nincs input, a kód egy sortördelést letiltó karaktert szúr be, a -t, így biztosítva azt, hogy a cella minden böngészőben helyesen jelenjen meg:

```
function tdd($text="")

    if(strlen($text))

        Stext = date("Y-m-d", Stext) . " " . date("H:i: s", $text); el

    se
```

```
        Stext   =   " ";

    return("<td>" . Stext . "</td>");

function tde($text="")

    if ( ! strlen (Stext) )    ; ■■ ■ ;; ■

        Stext = " ";

    return("<td><em>" . Stext . "</em></td>");
    í function
tdl($text="")

    if{!strlen(Stext)) {
        Stext = " ";

    return("<td>" . Stext . "</td>");

function tdn ($text="")                                          .·.!./ .  :.·▮

    if (strlen (Stext))                                            ;

        Stext = number_formát(Stext) ;

    else                                                  :

        Stext = "Snbsp;";

    return("<td align=\"right\">" . Stext . "</td>");

function tdre($text="")

    if (! strlen (Stext) )   ^^^

        Stext = " ";

return("<td align=\"right\"><em>" . Stext . "</em></td>");

function trow($text="") if ( !strlen(Stext)}

        Stext = tdl("Snbsp;");

    return("<tr>" . Stext . "</tr>");
```

Fájllista és elfoglalt lemezterület

Ha a fájllistát egy magasabb szinten lévő vagy egy grafikai munkát tartalmazó könyvtáron futtatjuk le, az hatalmasra nőhet. (Egyszer egy projektemnél egyetlen könyvtárban 49,00C eget, felhőt és napfelkeltét ábrázoló kép volt.) A következő kód a $print_limit-ben beállított megjelenítési limitet tartalmaz, amely korlátozza a böngészőnek küldött sorok számát. Az outputot elküldheted egyszerűen egy adatbázisba (a MySQL-t javaslom), hogy az SQL-szolgáltatások használatával az eredményeket - kezelhető blokkokban megjelenítve — onnan bogarászd ki.

Hivatkozás: **oldal:**

Adatbázis létrehozása **160**

A kód a get_directory_file()-függvényt használja az attribútumokat is tartalmazó fájllista és az elfoglalt lemezterület kigyűjtésére, az előző megoldásban alkalmazott ciklushoz hasonlóan. Azután egy ciklussal végigfut az eredményül kapott tömbön, a $found-on, majd a korábban definiált cellaformázó függvényeket használva megjeleníti az eredménysorokat. Két whileQ-ciklus fut a $found-on végig: az első a könyvtárszinten (az első kulcs), a második pedig a könyvtáron belül. A megjelenítési hosszúság korlátozása érdekében a középső printutasítások if($print_limit)-utasításokba vannak ágyazva, és a fájlok méretét egy külön kód adja hozzá a $directory_total-mezőhöz. Ez a mező az egyes ciklusok végén kiíratásra kerül, megadva a könyvtár által elfoglalt teljes lemezterület nagyságát.

A $parts = explode('V", $d)-vel kezdődő kódrész veszi a könyvtáron belül elfoglalt teljes területet, és a könyvtáron belül talált minden szinten hozzáadja a $size-tömbhöz. A /usr/Iocal/bin/ könyvtárat például usr-, local- és bin-részekre bontja, majd /usr-, /usr/local-és /usr/local/bin-könyvtárakat épít fel, a területet mindhárom könyvtár esetén hozzáadva a tömbhöz. A $size-t későbbi listákrajiasználja (az eredmény a 8.5 ábrán látható):

Directory	*\Mle* _____	*Size*
c:	'boot.ini	289
c:	BOOTSECT.DOS	512
c:	(COMMAND.COM	32,768
c:	'IO.SYS	98,304
c:	MSDOS.SYS	32,768
c:	INTDETECT.COM	26,816
c:	íntldr	156,496
c:cygwin/bin	aclocal	10,253
c:cygwin/bin	addftinfo.exe	33,792
c.xygwin/bin	addr2line.exe	337,408

8.5 ábra Fájllista és a felhasznált lemezterület

```
$print_limit = 10;
$found = get_directory_file($path) ;
print("<table border=\"3\">" . trow(tde("Directory") . tde("File")
    . tde("Size")) ) ; while(list($d,
$dv) = each($found))

    if (is_array($dv))

        $directory_total t 0;

        while(list($f, $fv) = each($dv))

            if(is_array($fv) ) if

                ($print_limit)

                    print (trow(tdl($d) . tdl($f) . tdn($fv["size"]
                    $print_limit--;

                $directory_total += $fv["size"];

            else

                if($print_limit)

                    print(trow(tdl($d) . tdl($f) . td($fv)));
                    $print_limit--;
                    \

                            \

        $parts = explodeC1/", $d) ;
        $acc_dir = "";
        while(list($pk, $pv) = each ($parts))
            {
            if (strlen ($acc dir))
                {
                $acc_dir .= "/";
                }
            $acc_dir .= $pv;
            if(isset($size[$acc_dir]))
                {
                $size[$acc_dir] += $directory totál;
                }
            else
                {
                $size[$acc_dir] = $directory_total;



    else {
        print(trow(tdl($d) . tdl($dv) . tdl() ) ) ;


print ("</table>" );
```

A könyvtárak által elfoglalt lemezterület kiíratása

A következő kód az előző példában létrehozott $size-tömböt rendezi és jeleníti meg, miközben a fájlok által használt terüjet listáját íratja ki. Az ebben és a következő kódokban használt ksortQ és a többi tombréndező függvényt a 3. fejezetben mutattam be.

A kód első része ellenőrzi, hogy a $size-tömb létezik-e és valóban tömb-e, arra az esetre, ha az előző kód megváltozott. A ksort()-tal rendezi a tömböt, majd a while()-t használva végigfut rajta. Mivel minden könyvtárhoz egy elem tartozik, a lista egyszerű, csak a tesztelés során megjelenítendő sorok korlátozására beiktatott külön kód miatt tűnik bonyolultnak. Ha már működik, megszüntetheted a korlátozást, de előtte jó sok fájllal teszteld le a böngésződ. A Netscape 4.76 egy 10,000 fájlból álló listával már nem boldogul, és minden nyitva lévő Netscape-ablakot bezár:

```
if(isset($size)     and     isarray($size))

    $print_limit    =    10;
    ksort($size);
    print ("<tcLble border=\ "3\ ">" . trow(tde("Directory")
        . tde("Size"))); while (üst ($d,
    $s) = each($size))

        if($print_limit) í
            print(trow(tdl($d)
            $print     limitr-;      .
                                  tdn($s)))

        }
    print
    }
```

Az eredmények a 8.6 ábrán láthatók.

\Ekrectory	*Size*
|c:	**347,953**
|c:KPCMS	**6,679,805**
|c:KPCMS/CMSCP	**625,012**
'c:KPCMS/DCPDB	**6,054,793**
c:ProgramFiles	**437,524,776**
c:ProgramFiles/ACDSee32	**2,060,879**
c.Program Files ACDSee32/Shortcuts	**1,248**
c:Prograin Files/Adobe	**6,072,513**
c:PrograniFiles/Adobe/Acrobat 4.0	**6,072,513**
;c:Program Files/Adobe/Acrobat 4.0/Help	**350,478**

8.6 ábra Könyvtárként felhasznált lemezterület

A legnagyobb könyvtárak által elfoglalt lemezterület kiíratása

A következő kód és lista az előző könyvtárlista egy változata, azzal a nagy különbséggel, hogy az arsort()-tal rendez. Az **arsort()**-függvény az asszociatív tömböket a kulcs szerint fordított sorba rendezi, így az elfoglalt lemezterület alapján listázva a könyvtárakat a legnagyobb lesz legelöl. A **$print_limit** lehetővé teszi, hogy csak a 10 vagy 100 legnagyobb könyvtárad jelenjen meg.

A 8.7 ábrán látható eredmény három sor összetett, számtalan alkalmazást tartalmazó könyvtárak mutat, majd a cygwin-könyvtárat, amely a legnagyobb, egyetlen alkalmazást tartalmazó könyvtár, majd a nyílt forráskódú StarOffice-alkalmazást, kicsit lemaradva tőle. Mindkét program esetében hasznos lenne a telepítésre jobban odafigyelni. Ha figyelembe veszem, hogy a Cygwint csak azért telepítettem a tesztgépre, hogy segítse a PostgreSQL telepítését, akkor a felhasznált lemezterület nagysága nevetségesen sok. Ezen kimutatás alapján újratelepítem a Cygwint, és mindent kikapcsolok, amire nincsen szükségem. A terjedelmes StarOffice telepítését is felülvizsgálom. A használt kód a következő:

íDírectory	*pze*
c:ProgramFües	**437,524,776**
|c:WINNT	**210,260,683**
|c: WINNT/system3 2 *1*	[151,858,259
˙c:cygwin 7	ijl30,321,558
c:Program **Files/StarOffice** '	I|l26,229,469
|c:Program Files/StarOffice/program	77,192,911
c:cygwia'usr	_j[**75,921,580**
ic:Program Files/TMG	1 50,275,431
ciProgram Files/JavaSofl	**j 43,485,512**
c:Program Files/JavaSoft/JRI *	**43,485,512**

8.7 ábra Felhasznált lemezterület, legnagyobb könyvtárként listázva

```
if (isset($size)    and    ís_array($síze))
    {
    $print limit = 10;
    arsort($size);
    print("<table border=\"3\">" . trow(tde("Directory")
        . tde("Size")));
    while (üst ($d, $s) = each($size)) {
        if($print_limit) {
    print(trow(tdl<$d)    .    tdn($s)));
    $print_limit--; }íprint("</table>");
```

A legnagyobb fájlok által elfoglalt lemezterület kiíratása

A következő feladat a legnagyobb fájlok felkutatása. Ez legalább olyan hasznos, mint az *előző,* a legnagyobb könyvtáraké volt. A következő kód újra felkeresi a $found-tömböt é> egy, az elfoglalt területet tartalmazó $file_size nevű új tömböt hoz létre, amelyet könnyer. rendezhetsz és egyszerűen feldolgozhatsz. Két while()-ciklus könyvtáranként, majd fájlonként fut végig a $found-tömbön. A fájlok szintjén minden egyes fájlbejegyzés hozzáadod:* a $file_size-hoz, méret, könyvtár és fájl szerint indexelve. Átkerülnek az utolsó hozzáférés adatai is, így látható, hogy mikor használták utoljára az adott fájlt. Az operációs rendszertől függően azonban előfordulhat, hogy az utolsó hozzáférés adatai nem azt mutatják, hogy mikor olvasták be utoljára a fájlt, pedig igazából erre lenne szükséged:

```
reset($found);
while(list($d,   $dv) = each ($found))

    if(is_array($dv))

       while(list($f, $fv) = each($dv))

           if(is_array($fv))

               $file size[$fv["size"]] [$d] [$f] = $fv["accessed"];

              }
       }
```

A $file size méret szerint listázva tartalmazza a fájlokat. A következő példában a krsortQ méret szerint fordított sorrendbe rendezi a tömböt, így a legnagyobb fájl jelenik meg legelőször. A kód többi része majdnem teljesen ugyanaz, mint az előző két tömböt listázó ciklus, azzal a különbséggel, hogy ez egy harmadik while()-ciklust is tartalmaz, hiszen a tömbnek a korábbi kettő helyett három kulcsa van. A külső ciklus és a fő szekvencia méret szerint indexelt, így a belső printutasítás először a méretet jeleníti meg. A $print_limit 10 sorra korlátozza a belső printutasítást, és a ciklus azután fut végig a fájlokon, hogy a megjelenítés leáll, így további feltételként minden while()-hoz hozzáadhatod az and $print_limit feltételt:

```
$print_limit = 10;
krsort($file_size);
print ("<table border=\"3\">" . trow(tde("Size")  . tde("Directory")
    . tdeC'File")  . tde("Last accessed"}));
while(list($s, $sv) = each($file_size))

    if(is_array{$sv))

       while(list($d, $dv) = each($sv))

           if(is_array($dv))

               while(list($f, $fv) = each($dv)) í
```

```
if ($print_limit)
    í
    print                    . tdl(Sd)   . tdl($f)   .
    (trow(tdn($s)            tdd(Sfv)))
    $print_limit--;
```

```
print ("</table>") ;
```

Az eredmény:

	5Í2K *Directory* \	*File*	*Last accessed*
jl8,103,692 c:Program Files/Sta^OflFice/help/01	shelp.dat	2001-01-30 20:05:00
(16,777,216c: cygwin/usr/1 ocal/pgs ql /data'pgxlog	0000000000000000 Í	2001-05-14 19:57:47
11,646,640 jc:Program Files/JavaSoft/JRE/1.3.0J)l/lib	rt.jar	2001-05-19 17:23:51
11,646,454 c:Program Files/JavaSoft/JRE/1.3/lib	rtjar	2001-05-02 21:33:04
9,089,327 ;c:Program Files/Opera/Mail/Apachetect	|Trash.MBS	2001-04-13 07:33:44
6,111,232 jc:Program Files/Iomega/Iomegaware	IIOGUREG.EXE	ToOl-01-17 22:45:44
j 6,094,848 c:Program Files/StarOffice/user/store	**outscs**	2001-01-30 19:52:14
| 6,047,744 c:WINlsn7system32	**.QuickTime.qts**	2000-12-04 17:35:40
5,767,168 |c:Program Files/StarOöice/program	applicatrdb	2000-05-08 05:20:00
|| 5,760,054 !c:WINNT	|Carolyn.bmp	2001-05-19 14:49:16

8.8 ábra Felhasznált lemezterület, legnagyobb fájlonként listázva

Képfájlok attribútumainak kiíratása

Ebben a megoldásban arról lesz szó, hogyan mehetsz végig úgy a fájlokon, hogy a képfájlok attribútumait kilistáztasd. Egyaránt használjuk az *előző* fejezet Gyors megoldások részében látott $path-t és get_directory_file()-függvényeket. A képattribútumokat a getimagesizeQ-képfüggvény használatával gyűjti össze, melyet - a többi képfeldolgozó függvénnyel egyetemben -a 11. fejezetben magyarázok el teljes részletességgel. Ha egyszer elsajátítottad az alábbi példában szereplő kódot, képes leszel képeket listázó és megjelenítő oldalakat létrehozni. Ötvözve ezt az 5. fejezet MySQL-adatbázis kódjával, akár képkatalógust is készíthetsz, a 11. fejezet kódjával kibővítve automatikusan hozhatsz létre kisméretű előnézeti képeket, és adhatsz hozzájuk címet és szerzői jogi figyelmeztetést is.

A getimagesize() fájlnevet és egy további, opcionális paramétert tartalmazhat, amit itt $extra-nak nevezek. Eredményként tömböt ad - a példában ezt $att-nak nevezem -, amely a képre vonatkozó információkat tartalmaz. A $att-ba négy hagyományos érték kerül, illetve JPEG-képek esetén további kettő, de a vizsgált képek között egyikről sincs további információ, és a két hagyományos érték közül csak kettő, a magasság és szélesség az, amelyik

hasznos. így a kód az összes értéket eltárolja, de csak a magasságot és szélességet jeleníti meg. A Sextra-ba egy a képben tárolt információkat tartalmazó tömb kerül - ez a fajta információ jelenleg csak a JPEG-képekről érhető el, és a vizsgált fájlok között nem volt JPEG. Jóllehet a kód összegyűjti és egy megjelenítésre alkalmas sztringbe helyezi az információt, a sztringet nem helyeztem a kimeneti HTML-táblázatba. A több szerveren lefuttatott teszt csak néhány olyan JPEG-fájlt talált, amelyek többletinformációt tartalmaztak, és ez a többlet információ is csupán annyi volt, hogy Photoshopból származtak.

Képinformációk kigyűjtése

A következő kód úgy néz ki, mint az *előző* Gyors megoldásokban használt fájlfeldolgozó kód, a képek többletinformációinak kigyűjtése érdekében néhány helyen megváltoztatva. Az első változtatás a belső feldolgozó ciklus módosítása, hogy az csak képeket olvasson be. A másik változtatás hatására a kód a képattribútumokat a $images-tömbbe helyezi el. Mivel a kód valójában az új képattribútumokat a már létező fájlattribútumokkal vegyíti, olyan további megjelenítő oszlopokat szúrhatsz be, mint például a létrehozás ideje:

```
$found = get
directory_file($path); reset($found) ;
while (üst ($d, $dv) = each ( $f
    otmd) ) {
   if(is_array($dv)) { ~while (üst ($f,
       $fv) = each($dv))

          $x5 = strtolower(substr($f, -5));
          $x4 = substr($x5, -4); if($x4 ==
          ".jpg" or $x5 == ".jpeg" or $x4 == or $x4 ==
          ".png" or $x4 == ".swf")              ".gif"

             $att = getimagesize($d      /'    $f, $fv["extra"])
             if(isset($att[0] ))  {$fv["width"] = $att[O];}
                                 if(isset ($att[l]))
             if(isset ($att [2])) {$fv["height"] = $att[l];}
             if(isset($att[3])) {$fv["it"] = $att[2];}
                                  {$fv["htral"] = $att[3];}
             if(isset($att[4]))  {$fv["channel"] = $att[4];}
             if(isset ($att[5])) {$fv["bits"] = $att[5];}
             if(is_array($fv["extra"]))

                while (üst ($ek, $ev) = each ( $fv [ "extra" ])) if

                   (strlen(Sextra))

                      $extra .= "<br>" .              .
                      $ek .                                $ev;

                   else

                      Sextra = $ek . ": " .
                      $ev;


          if(isset ($extra)     and
          strlen($extra))
```

```
            $fv["extra"] = Sextra;

        else { $fv["extra"] =
            "";


$images[$d][$f] = $fv;
```

A belső ciklus akkor fut, ha egy fájl .jpg vagy más, az if()-utasításban felsorolt végződések egyikével rendelkezik. Ha rengeteg fájlt kell beolvasnod, amelyek között csak néhány kép van, helyezd a fájltípus-ellenőrzést a get_directory_file()-függvénybe, így rengeteg felesleges műveletet takaríthatsz meg. Az olyan operációs rendszerekben, amelyek nem használnak fájlvégződéseket, úgy döntheted el egy fájlról, hogy kép-e, ha a getimagesizeQ-függvényt lefuttatod. Csak azt kell megnézned, hogy az eredmény tömb-e vagy pedig hamis érték.

Képinformációk megjelenítése

Most, hogy a képattribútumok immár a $images-ben vannak, az előző Gyors megoldásokban használt megjelenítő kódhoz hasonló ciklussal fuss végig a tömbön, csupán a megjelenítendő mezőket kell megváltoztatnod. A következő kód a tömb használata előtt ellenőrzi, hogy a tömb létezik-e, arra az esetre való tekintettel, ha valaki az előző kódot megváltoztatta volna. A kód beállítja a tesztre vonatkozó megjelenítési korlátot, ez jelen esetben 10, majd két while()-ciklus fut végig a könyvtárakon és fájlokon, az elemeket megjelenítve. Ha a legnagyobb, a legkisebb vagy bármilyen egyedi tulajdonsággal rendelkező képet keresel, további if()-utasításokat szúrhatsz be, vagy rendezd a tömböt valamilyen más sorrendbe, az előző Gyors megoldásokban a 10 legnagyobb lemezterület elfoglaló fájl megjelenítésére mutatott trükkök használatával:

```
if(isset($images)     and
    is_array($images)} {
    $print_limit    =
    10; reset($images);
    print("<table    border=\"3\">"      .    trow(tde("Directory")      .
    tde("File") .    tde("Size")     .     tde("Height")     .    tde("Width")     .
    tde("Channel") .    tde("Bits")     .     tde("Extra"))     ); while (üst ($d,
    Sdv)    =    each ($images) } {
        if(is_array($dv)) {
            while(list($f,    $fv)    =
                each($dv)) { if(is    array($fv))
                    { ~
                    if($print limit)
```

```
print(trowftdl($d)    .    tdl($f)    .
    tdl($fv["size"]) .    tdl($fv["height"])    .
    tdl($fv["width"]) .    t41($fv["channel"])    .
    tdl($fv["bits"]) .    tdÍTSnsxtra))    );
$print_limit--;
```

```
print ("</table>"    );
```

Az eredmények:

Directory	*Size*	*Heigh*	*Width*
i:/petennoulding/web/root/phpblackbook/fíles/images jbs.gif	255	*11*	61
i:/petennoulding/web/root/phpblackbook/files/images copy.jpg	6530	100	80
lii/petennouldingí'web/root/phpblackbook/files/images kingparrot.jpg	60810	500	250
|i:/petermoulding/web/root/phpblackbook/files/images me.jpg	6530	100	80
|i:/petermoulding/web/root/phpblackbook/files/images notme.jpg	50024	250	280
|i:/petermoulding/web/root/phpblackbook/files/images peter.jpg	6530	100	80
[i :/petermoulding/web/root/phpblackbook/files/images php-syd.gif	6405	100	190
|i:/petermoulding/web/root/phpblackbook/files/images roo.jpg	50024	250	280

8.9 ábra Képfájlok listája az attribútumaikkal

Bármilyen típusú adat megjelenítése

A következő példa célja olyan kód létrehozása, amellyel bármilyen fájlt megjeleníthetsz a képernyőn anélkül, hogy a böngésződ szétdurranna. A példa egy szövegfájlt (php.ini), egy HTML-fájlt (test.html) és egy képfájlt (me.jpg) tartalmaz. A PHP-fájlfüggvényekkel bármilyen, az operációs rendszered és a biztonsági beállításaid alapján engedélyezett fájlt beolvashatsz és megjeleníthetsz.

Általános kód

A kezelhetőség érdekében, ahogy azt az előző példában is tettem, a **$print_limit** használatával 10 sorra korlátozom a megjelenítéseket, a sorokat pedig 70 karakterre csonkolom, ahogy azt a **$line_limit-ben** beállítottam. Az első példa a fileQ-függvényt használja a teljes fájl memóriába történő beolvasására, majd pedig a **$print_limit-et** alkalmazza a megjelenítéskor. Ez akkor ideális, ha megjelenítés előtt a kód eldob néhány sort, te pedig ugyanannyi sort akarsz megjeleníteni. A második példa azt feltételezi, hogy a kis tesztfájlocska egy masszív szövegfájl, az **fgets()**-függvénnyel egyszerre a fájl egy sorát olvassa be, és a beolva-

sást korlátozza, nem a megjelenítést, ami ideális megoldás a nagy fájlok elejének megtekintésére. A sorhossz alkalmazható a hosszú sorok kezelésére, és lehetőséget ad neked például egy fájl egyes rekordjai elejének megtekintésére:

```
Sline^limit = 70;
$print limit = 10;
```

Szövegfájl megjelenítése

A PHP php.ini fájlja körülbelül 50 sornyi beállításból és rengeteg megjegyzéssorból áll, így elég hosszú arra, hogy példaként használjuk. Ezen kívül tisztában vagyok vele, hogy semmi olyan speciális karaktert nem tartalmaz, amely felizgatná a böngésződet. A kód a **file()**-függvényt használja arra, hogy a a fájlt egy tömbbe beolvassa, minden newline-karakter után új elemet kezdve. A newline-karakterek megmaradnak, így az összes elem, talán az utolsót leszámítva, newline-karakterrel végződik. A nem szövegfájlok a bemeneti fájl összes bájtját tartalmazó egyetlen nagy elemmé válhatnak. A while()-ciklus végigfut a tömbön annak végéig, vagy ameddig a **$print_limit** nullára nem csökken. A kód többi része egyszerű megjelenítés. A substr()-függvény a sor hosszúságát a $line_limit-nek megfelelően korlátozza, és az első prmtsor kis fejlécet hoz létre a lista azonosítására:

```
$file = "php.ini";
$line_limit = 70;
$print_limit =10;
$text = file($path . "/" . $file);
print("<br><em>File: " . $path . "/" . $file .
while (üst ($k, $v) = each($text) and $print_limit)

    $v = substr($v, $line limit)
    0, print("<br>" .
    $v)
    $print_limit--;
```

A kód eredménye:

```
File: í:/petermoulding/web/root/phpblackbook/files/php.ini
 [PHP]
; Language Options ;
engine = On
short_open_tag = Off
asp tags = Off
precision = 14
y2k_compliance = Off
output_buffering = Off
output handler =
```

A következő kód egyetlen nagy különbségtől eltekintve ugyanaz, mint az előző: a **file()** helyett az fgets() olvassa be az adatokat, mivel a **file()** a fájl méretétől függetlenül minden egyes alkalommal az egészet olvassa be, az fgets() ezzel szemben csak azt a rekordot, amelyiket kiválasztod, utána pedig megáll. Az fgetsQ háromlépcsős folyamatot igényel: az fopen() megnyitja a bemeneti adatnak szánt fájlt, és a mutatót a fájlra állítja, az fgetsQ a fájl egy rekordját olvassa be, a fájl végén pedig hamis értéket ad eredményül, ha pedig kész

vagy a fájl beolvasásával, az fclose() bezárja azt. A hátralévő példák a kódrövidségért a file()-t használják, de bármelyiket megváltoztathatod az fgets() vagy hozzá hasonló függvények használatával, amelyei csökkentheted a nagy fájlok miatt feleslegesen elvégzett műveleteket. Mivel az ilyen típusú fájlműveletek nagy fájlok esetén használatosak, egy pótlólagos, a filesizeQ-függvényt tartalmazó sor a fejrészben megjeleníti a fájl méretét:

```
$file = "php.ini";
$line_limit = 70;
$print_limit =10; $size =
filesize($path                "/"     .     $file);
print("<br><em>File: "       $path   .     "/"    .
                                                    size: $size

$pointer = fopen($path . "/" . $file, "r"); while($line =
fgets($pointer, $size) and $print_limit)
    {
    $line = substr($line, 0, $line_limit);
    print("<br>" . $line);
    $print    limit—;
    }
fclose(\5pointer);
```

Ezen kód eredménye - a fejrészt leszámítva - megegyezik az előzőével, így a következő csak az első sor:

```
File:    i:/petermoulding/web/root/phpblackbook/files/php.ini,      size:
6301
```

HTML-fájl megjelenítése

A következő kód egy HTML-fájllal birkózik meg. A HTML-fájlok megjelenítésével az a probléma, hogy a fájlból megjelenített HTML összeakad a megjelenítő oldalad HTML-jével. A megoldása a htmlspecialchars()- vagy htmlentities()-függvények használata, melyek a HTML-t és egyéb karaktereket sztring megjelenítésre konvertálják, amelyek a képernyőn anélkül jelennek meg, hogy a böngésződ HTML-nek értelmezné azokat. Ha a tesztfájlt anélkül jeleníted meg, hogy a HTML tag-eket speciális karakterekkel helyettesíted, a 8.10 ábrán látható eredményre jutsz. A következő kód használatával viszont a 8.11 ábra lesz az eredmény:

File: i :/petermouiding/web/root/phpblackbooh/files/test, html

Heading One

Somé text.

8.10 ábra HTML-fájl a tag-ek felcserélése nélkül

```
$file = "test.html";
$line_limit = 70;
$print_limit = 10;
$text = file($path . "/" . $file);
print("<br><em>File: " . $path . "/" . $file
while (üst ($k, Sv) = each($text) and $print_limit)
    t
    $v = substr($v, 0, $line_limit);
    print("<br>" . htmlentities($v));
    $print
```

Fűe: i:,'pstermoulding/Yueb/root/phpbtackbook/files/test.html **<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">**

<head>
<title>Test page</title>
</head> <body>
<hl>Headmg One</hl>
text.</p>

/htmí

8.11 ábra HTML-fájl a tag-ek felcserélésével

Miért van két függvény ugyanarra a feladatra? Nem tudom, ez biztos egyike azoknak a titkos PHP-fejlesztői dolgoknak. A **htmlspecialchars()** a karakterek közül csak a legszükségesebbeket változtatja meg, hogy a HTML-t értelmezés nélkül jeleníthesd meg. A **htmlentitiesQ** is ugyanezt a karakterkészletet változtatja meg, no meg számtalan speciális ékezetes karaktert, amelyeket főleg az európai nyelvekben használnak. Amennyiben az európai latin betűktől eltérő karakterkészletet kell megjelenítened, a **htmlspecialchars()** használatával végrehajthatod az alapvető karakterek cseréjét, a többi karaktert pedig a saját célod alapján egyedileg fordítsd. Az általános fordításra a következő hexadecimális megjelenítésben találsz példát.

Bármilyen típusú fájl megjelenítése

A következő kód az előzőnek egy jelentős kibővítése, hiszen ezzel bármilyen létező fájlt megjeleníthetsz. A **$print_limit** a megjelenítendő rekordok számát állítja be, a **$line_limit** az egy rekordból megjelenő karakterek számát, a **$segment_length** az outputban soronként megjelenő karakterek számát határozza meg, a **$current_record** a hosszú fájlokban való tájékozódást segítő rekord, a **$current_byte** pedig egy a hosszú rekordokban való tájékozódást segítő mező. A **get_html_translation_table()** a **htmlentities**()-hez használt táblázatot hívja be, így további elemeket adhatsz hozzá és **strtr()-rel** pedig manuálisan fordíthatsz szöveget, ezáltal nagyobb kontrollhoz jutsz, és a szóközöket -re fordíthatod. Ez a karakter akkor is szóközt jelentet meg a böngészőkkel, ha azok maguktól nem jelení-

tenék azt meg. A fordító táblázat a $safe-be kerül, a while()-ciklus pedig a decimális 0-tól a decimális 32-ig (szóköz) az összes karakterhez további elemet rendel, így az összes furcsa karakter, mint például a sortörés, szóközként jelenik meg ahelyett, hogy a HTML-t megbolygatná a honlapodon. Minden rekord egy tömbbe, a tömb összes eleme pedig a $v-be kerül, és az elemek a Sline_limit-nek megfelelő'en csonkulnak, és részekre oszlanak. Minden rész külön sorban jelenik meg. Az **ord()** az egyéni karaktereket a decimális értékükre, a dechex() pedig a decimális értékeket a hexadecimális megjelenítésre konvertálja. Mivel bizonyos hexadecimális megjelenítések elvesztik a elöl álló nullájukat, a kód egy sora visszaadja ezeket, egy kis ciklus pedig szóközöket szór a részek karaktermegjelenítéséhez, így a részek igazítottan jelennek meg az oszlopban. A betűtípus meghatározásával a karakterköz rögzítetté válik, így a képernyőn könnyedén megszámlálhatod a karaktereket:

```
$file    =    ". /images/me.jpg";
$line_limit    =    256;
$print_limit    =    10;
$current_record    =    0;
$segment_length    =    16;
$safe    =    get_html_translation_table(HTML_ENTITIES);
for{$i    =    0;    $i    <=    32;    $i++)

    $safe[chr         =
                    " ";

$text = file($path . "/" . $file);                                  ;
print("<br><em>File: " . $path . "/" . $file . "</em>" );
while (üst ($k, $v) = each($text) and $current_record <= $print_limit)

    $current__record++;
    $current_byte = 1;                                                          (
    print("<br><font face=\"'Courier New',Courier,monospace\">"                 s
        . " Current record: " . \$current_record                                ■
        . ", current byte: " . $current_byte . "</font>" );
    $v = substr($v, 0, $üne_limit) ;                                            !
    $current_byte += strlen($v);
    while (strlen($v) )                                                         ,a

        $segment = substr($v, 0, $segment_length);
        $v = substr($v, $segment_length);
        $hs = "";
        for($i = 0; $i < strlen($segment); $i++)

            $hex = dechex(ord(Ssegment[$i]));
            if(strlen($hex) < 2)
                {                                                               !

                $hex = "0" . $hex; $hs

            .= $hex . " ";

        while(strlen(Ssegment) < $segment length)

            $segment .= " ";

        print ("<brxf ont face=\" ' Courier New', Courier, monospace\">"
            . strtr(Ssegment, $safe)  . "  " . $hs . "</font>" );
```

A kódot könnyűszerrel módosíthatod, hogy a számodra szükséges feladatnak jobban megfeleljen, így például a nagy fájlokat a **file()** helyett az fgets() használatával olvashatod be. Ha a 300 és 305 közötti rekordokat akarod megjeleníteni, állítsd a **$print_limit-et** 305-re, és rejtsd el az első háromszáz megjelenített rekordot. Jóllehet a PHP a fájlfüggvények nagyon széles választékával rendelkezik, neked csak néhányra van szükséged a megfelelő adat megtalálására.

Üres fájlok létrehozása

Furcsának tűnhet egy üres fájl létrehozása, de ez egyenértékű azzal a trükkel, amikor a szkriptben üres változót hozol létre. Ha hosszú szkriptet írsz, amely véletlenszerűen tárol és olvas be értékeket, általában a szkript elején üres változót hozol létre, és megjegyzéseket teszel a változó használatára vonatkozóan, a kódod többi része pedig feltételezi, hogy ez a változó végig létezik. Ugyanezt megteheted a fájlokkal is - például létrehozod az összes fájlt, amelyre a weboldalad új felhasználójának szüksége lehet. Lehet, hogy az új felhasználónak nincsen listája a kedvencekró'l, de a felhasználói név létrehozásakor kialakíthatsz egy üres kedvencek fájlt, így amikor a felhasználó bejelentkezik nem áll fenn a fájlallokációs hiba veszélye. Ennek a megoldásnak a lehetséges alternatívája egy olyan szkript létrehozása, amely a felhasználó első bejelentkezésekor létrehozza az alapértelmezett fájlokat, így az összes lehetséges hiba egy helyen történik, és egy telefonhívással megoldható.

A touchQ-függvény egy fájlnevet és egy opcionális dátum/idő paramétert igényel, üres fájlt hoz létre, vagy ha a fájl már létezik, csak az utolsó módosítás idejét változtatja meg. Sikeres végrehajtás esetén igaz, hiba esetén hamis értéket eredményez. Az opcionális dátum/idő Unix időjelzés, így úgy állíthatod be az utolsó módosítás idejét, ahogy azt a **filemtimeQ** adja. Az alapértelmezett dátum/idő az aktuális idő, és akkor használhatod az alapértelmezést, ha új felhasználó számára másolsz fájlokat, és úgy akarod a fájl létrehozásának idejét megváltoztatni, hogy azt mutassa, amikor a felhasználó először lépett be.

A következő kód a **touchQ** használatával változtatja meg egy fájlban a dátum/időt:

```
$file = "./touchme.txt"; touch($file,
strtotime("+1 week")); print("<br>File
" . $file . " mtime: "
. date("Y-m-d H:m:i", filemtime(\$file)); touch($file);
print("<br>New mtime: " . date("Y-m-d H:m:i",
filemtime(\$file));
```

A következő két sor mutatja a Windows NT alatt használt PHP 4.0.5 által adott eredményt:

```
File ./touchme.txt mtime:
0 New mtime: 0
```

A következő sor a Linux alatti eredményt mutatja abban az esetben, ha nem megfelelő jo-

gosultsággal próbálták használni a függvényt. Ha nincsen jogosultságod a fájlhoz való hozzáférésre, két hibát kapsz eredményül, minden **touch()-ra** egyet. Ha hozzáférhetsz a fájlhoz, de nincsen jogosultságod megváltoztatni, egyetlen hibát kapsz az első touch()-ra, amikor a dátum/időt próbálod megváltoztatni:

```
Warning: utime failed: Operation not
permitted (Figyelem: Hozzáférés megtagadva)
```

Fájlok feltöltése

Ez a példa a 9. fejezet „Fájlok feltöltése" című részből vett kód kivonatával kezdődik. Először ellenőrizd, hogy a php.in-ben a **file_uploads** be legyen kapcsolva. Windows és Windows NT alatt meg kell határoznod a **upload_tmp_dir** könyvtárat, amelyhez persze hozzáférésed kell, hogy legyen. Győződj meg arról, hogy az **upload_max_size** nagyobb, mint a legnagyobb fájlod, a **memory_limit** pedig legalább kétszer akkora, mint az upload_max_filesize. Ez után tedd a HTML-t egy olyan oldalba, amely megkérdezi a felhasználót, hogy melyik fájlt akarja feltölteni. A HTML-ben a MAX_FILE_SIZE-nak legalább akkorának kell lennie, mint a feltöltendő fájl:

```
file_uploads = On upload tmp dir = T:\upload upload_max_filesize
= 8M memory_limit = 30M
```

Szúrd be tesztoldaladra a következő, itt uploadingfiles.html-nek nevezett kódot, állítsd az action= űrlap paramétert az oldalad nevére, majd teszteld az oldalt. Az if() a feltöltés által létrehozott $uploadfile-mező észlelésére szolgál. Ha a mező nem létezik, az űrlapot mutatja, ha pedig létezik, akkor annak a tartalmát jeleníti meg. A **$uploadfile** az űrlapban <**input type="file"**> tag-jében beállított név, és a bemeneti fájl nevét tartalmazza teljes elérési úttal együtt. A **$uploadfile_name** a **$uploadfile-ból** kivont fájlnevet tartalmazza. A **$uploadfile_size** a fájlméretet tartalmazza, sikertelen feltöltés esetén értéke 0. A **$uploadfile_type** a fájl MIME-típusát tartalmazza arra az esetre, ha a fájltípus alapján egyedi feldolgozást akarsz végrehajtani:

```
if(isset(Suploadfile))

   print("<br>All set to upload a file." );
   print("<br>uploadfile: " . $uploadfile );
   print("<br>uploadfile_name: " . $uploadfile_name
   print("<br>uploadfile_size: " . $uploadfile_size
   print("<br>uploadfile_type: " . $uploadfile_type

else

   print("<br>Please teli me which file I should upload." )
   print("<form enctype=\"muitipart/form-data\""
       . " action=\"uploadingfiles.html\" method=\"post\">" .
       "<input type=\"hidden\" name=\"MAX_FILE_SIZE\"" . "
       value=\"200000\">"
```

```
"<input type=\"file\" name=\"uploadfile\" size=\"60\"
"<br><input type=\ " submit\" value=\ "Upload\"x/f orm>"
```

Amikor az űrlap megjelenik a böngészőben, a 8.12 ábrán látható, Netscape 4.76-ból vett űrlaphoz hasonló dolgot fogsz látni. A legújabb böngésző'k - a Mac-en lévő böngészőket leszámítva - hasonlóképpen kezelik a fájlfeltöltést. Valaki megpróbálta Mac-ről is engedélyezni a fájlfeltöltést, de olyan sok probléma és hiba jelentkezett, hogy végül megszüntette az oldalán a fájlfeltöltés lehetőségét. Nos, megoldható a dolog, de azt javaslom, személyesen ülj le egy Mac-hez (vagy bármilyen hibákat eredményező böngészőhöz), és inkább saját magad tesztelj le mindent, mint hogy más emberek értelmezésére támaszkodj. Minden általad feltöltött fájlt töltsd le, vagy küldd vissza e-mail-ben ugyanarra a gépre, ahonnan feltöltötted, és az esetleges módosulások észrevételéhez hasonlítsd össze a feltöltött fájlt az eredetivel.

Please teli me which file I should upload.

I:\PeterMoulding\web\root\phpblackbookAfiles\test.html Browse...

Upload |

8.12 ábra Fájlnév űrlapfájlfeltöltéshez

Ötlet: Macintosh-on számtalan tömörítési rutin van, és nincsen fájlkiterjesztés a fájl formátumánakjelzésére. Ha azt gyanítod, hogy egy fájl hibát tartalmaz, próbáld kicsomagolni, és hasonlítsd összes a kicsomagolt fájlt az eredetivel. A Mac-fájlok zippelhetők, de gyakrabban vannak az Aladdin Stufflt-tal tömörítve (www.aladdinsys.com).

Ha a fájl feltöltődik, a következő négy sorhoz hasonlót látsz megjelenni. A php22.tmp a feltöltési célkönyvtár ideiglenes neve, a test.html pedig a forrásgépen levő eredeti fájlnév. A meglehetősen megbízhatatlan JavaScript használata nélkül lehetetlen felfedni az eredeti könyvtár kilétét. Ha trükközni akarsz, például emberek után kémkedni, olyan fájlokat feltöltve, amelyeket ők nem is választottak ki, a JavaScriptet kell használnod, ami megbízhatatlansága miatt csupa haszontalan állományt fog feltölteni:

```
Ali set to upload a file.
uploadfile: T:\upload\php22.tmp
uploadfile_name: test.html
uploadfile_size: 183
uploadfile_type: text/html
```

Ha olyan fájlt próbálsz meg feltölteni, amely nagyobb, mint a HTML MAX_FILE_SIZE-ban (jól látod, bizonyos böngészők vagy szerverek miatt nagybetűvel kell írni) meghatározott méret, a következő üzenetet kapod:

```
Warning: Max file size exceeded - file [uploadfile] not
saved (Figyelem: Fájlméret túllépés - az [uploadfile] fájl nincs
elmentve)
```

Mit tegyél a feltöltött fájllal? A következő kód beszúrásával a későbbi használathoz a megfelelő könyvtárba mentheted a fájlt, vagy adatbázisba olvasáshoz megnyithatod az ide-

iglenes fájlt. A legújabb PHP-változatokban elérhető **is_uploaded_file()** segítségével ellenőrizheted, hogy a feltöltött fájl rendben van-e, a **move_uploaded_file()** pedig az ideiglenes könyvtárból az állandó könyvtárba menti a fájlt:

```
if (isset($uploadfile) and is_uploaded_file($uploadfile)) {
    if(move_uploaded_file($uploadfile,
        $path . "/test/" . Suploadfile_name))
        {
        print("<br>Uploaded file saved."
        ); }
    else { print("<br>Uploaded file savé
        failed." );
```

Megjegyzés: A move_uploaded_file() a PHP 4.0.3-tól érhető el.

Ha nehezen kibogozható hibákat követsz el, a következő két hibaüzenetet kapod, vagy mert a célkönyvtár nem létezik, vagy mert bár létezik, rosszul vannak beállítva a hozzáférési jogok:

```
Warning: Unable to create (Figyelem: Nem sikerült létrehozni)
   [1]i:/petermoulding/web/root/phpblackbook/files/
    upload/test.html': No such file or
    directory
   (Figyelem: Nem sikerült létrehozni) Warning:
Unable to move 'T:\upload\php21.tmp[1] to
   'i:/petermoulding/web/root/phpblackbook/files/upload/test.html'
   (Figyelem: Nem sikerült áthelyezni)
```


A másik elkerülendő hiba az inkompatíbilis fájnevek hibája, amely akkor történik, ha egy Windows alatti nevet, például a „test file"-t, Unix alatt próbálod használni, ahol a fájlnevekben a szóköz nem engedélyezett. Fájlt veszíthetsz el, ha a TestFile és testfile-fájlokat Unixből töltőd fel, ahol két különböző fájlnak számítanak (hiszen a Unix különbséget tesz a kis- és nagybetű között), majd Windows, Windows NT vagy hozzájuk hasonló operációs rendszer alatt mented el őket, hiszen ezek az operációs rendszerek nem különböztetik **meg** a kis- és nagybetűket, hanem azt feltételezik, hogy a két fájl ugyanaz, így a második felül fogja írni az elsőt.

Ha olyan fájlt próbálsz meg feltöltetni, amely meghaladja HTML MAX_FILE_SIZE méretet, hálózati hibaüzenetet kapsz a böngészőben, és a weboldal szkriptjének végrehajtása elmarad.

CRC-számolás fájlokra

Egy megfelelő CRC-vel (Ciklikus Redundancia Ellenőrzéssel, Cyclical Redundancy Check) meggyőződhetsz arról, hogy a fájl a hálózaton keresztül történt továbbítás után is változatlan, de a CRC-t használhatod olyan fájlok összehasonlítására is, amelyeket máskülönben nehéz összevetni - ilyen például két kép. A hagyományos CRC32 32 bites identi-

tást **állít elő,** amely 4 milliárd esetből egyszer téved. Ez a példa azt mutatja meg, hogyan hozhatsz létre hagyományos **PHP** CRC32-t, és hogyan konvertálhatod át azt gyakrabban használt egész vagy hexadecimális formára, a más rendszereken generált CRC-kkel való összehasonlítás végett. A **PHP** a legtöbb 32 bites rendszeren 32 bites előjeles egészeket használ a mínusz kétmilliárd és plusz kétmilliárd között tartományból. Az általános CRC32 egy előjel nélküli 32 bites egész, 0 és 4 milliárd közti értékkel.

A következő kód két fájlt, a $a-t és a $b-t nézi meg, megkeresi és megjeleníti a fájlméreteket és a CRC-ket. Mivel a fájlok különböző méretűek, nem lenne szükség a CRC-k összehasonlítására, de vedd úgy, mintha ugyanolyan **méretűek lennének. A** CRC alapján megkülönböztethető a kettő, és a CRC kiszámítási módja miatt kiemeli a kis különbségeket, így a hasonló fájlok teljesen különböző CRC-ket produkálnak:

```
$a = "/images/me.jpg"; $b =
"/images/kingparrot.jpg" ; $asize =
filesize($path . $a); $bsize =
filesize($path . $b);
print("<br>Size a: " . r b: .
Sasize $acrc = (double) $a)
crc32($path , $bcrc = $b)
(double) crc32($path , , n $bcrc);
print("<br>CRC a: " . $acrc
.
```

A kód eredménye:

```
Size a: 6530, b: 60810
CRC a: 199407017, b: -40759560
```

Az itt kapott CRC-k kiválóan alkalmasak az összehasonlításra, hiszen ugyanolyan fájlrendszerből származnak. Ha viszont különböző fájlrendszerből származó CRC-ket figyelsz meg, 32 bites előjel nélküli egészeket is találni fogsz, ezért a következő **kód** a CRC-t 32 bites előjel nélküli egészre konvertálja. Ehhez először a 4 milliárdos értékre lesz szükséged, amit a legegyszerűbben a 8 bitet négyszer összeszorozva kaphatsz meg. Az alábbi kód a negatív CRC-ket kivonja a 4 milliárdból, így kapja meg a végeredményt:

```
$thirtytwo = (double) 256 * 256 * 256;
print("<br>Thirtytwo: " . 256
if(\$acrc < 0) $thirtytwo);

    $acrc = (double) $thirtytwo + $acrc;

if($bcrc < 0)

    $bcrc = (double) $thirtytwo + $bcrc;

print("<br>CRC (unsigned) a: " . $acrc . $bcrc)

", b: "
```

Az eredmény:

```
Thirtytwo: 4294967296
CRC (unsigned) a: 199407017, b: 4254207736
```

A következő kód az előjel nélküli számot hexadecimálissá alakítja. Ha a szám elveszíti az első nullát, az if()-es kódrészlet visszaadja azt:

```
$acrc = dechex($acrc);
$bcrc = dechex($bcrc);
if (strlen($acrc) < 8)
    {
    $acrc = "0" . $acrc;
    } if (strlen($bcrc) <
8)
    {
$bcrc = "0" . $bcrc; } print("<br>CRC (hex) a: " . $acrc .
", b: " . $bcrc );
```

Az eredmény:

CRC (hex) a: 0be2b5a9, b: fd920ef8

9. *fejezet*
Űrlapok

Áttekintés

A PHP-ban könnyű űrlapot létrehozni, a PHP-tömbökkel pedig hosszú listákat adhatsz hozzá ezekhez az űrlapokhoz. A cookie-kból, URL-lekérdezésekből, GET- és POST-alapú űrlapokból származó változók egyszerű PHP-beli megjelenítésével gyorsan eljuthatsz a többoldalas űrlapokig és az olyan egyedi űrlapfunkciókig, mint a fájlfeltöltés. Az űrlap-vezérlők a hagyományos űrlapoktól távol eső dolgokra, így a következő részben említett alternatív navigációra is használhatók. Az űrlapok elküld gombjai a navigációs célú képek és lmkek alternatívájaként is felfogható.

Jelen fejezetben végig az űrlapok kérdéseinek létrehozásának módszereit taglalom, hogy a kód újraírása nélkül gyorsan és egyszerűen építhess fel űrlapokat. A „Gyors megoldások" részben épített újrafelhasználható űrlappal és kérdező függvényekkel anélkül rakhatsz be további kérdéseket és anélkül módosíthatod az űrlapot, hogy a kódhoz hozzányúlnál. Ha már végeztél az 5. és a 20. fejezettel, esetleg adatbázisba vagy XML-fájlba akarod fordítani az űrlapfüggvények bemenő adatait. Az adatbázis felőli megközelítés a gyorsan változó honlapok esetén ideális.

Alternatív navigáció

Az oldalak linkelésének szokásos módja a rögzítő tag, az <a> használata. A rögzítő tag a változókat egy URL-en keresztül továbbítja, de az URL-ek gyorsan zavarossá válhatnak, amikor változókkal töltőd fel őket. Az űrlapok az értékeket az átláthatóbb POST-módszerrel továbbítják, és a POST mind nagy értékekkel, mind az értékek hosszú listájával működik. (Van egy olyan POST-űrlaporn, amely 49,000 fájlnevet továbbít az egyik oldalról a másikra.)

Az űrlapokban a <form>-tag helyettesíti az <a> rögzítő tag-et, az <input> tag-ek helyettesítik az URL-lekérdezési sztring egyedi elemeit, és az űrlapelküldő gomb helyettesíti azt a szöveget, amelyet a rögzítési kezdő- és végtag közé teszel. A következő példa egy előformázott lekérdezésre mutató normál linket mutat, amelyet ugyanez a link egy űrlapon keresztüli hozzáféréssel követ. A lekérdezésnek megvan az a képessége, hogy a már megtekintett dokumentumokat kizárja. Az első példában minden kizárt dokumentumot elneveznek egy mezőn keresztül, például excludeOl, míg az űrlapos példa egy example[]-nek nevezett mezőt használ, ami a PHP-ban egy egyszerű tömböt hoz létre. Mikorra a keresgélő dokumentumok tucatját végignézi, a link első formája a legtöbb böngészőben hibákat okoz, a második forma ezzel szemben jóval több dokumentumnevet kezel, mint amennyit egy session alatt át lehet böngészni:

```
$link = "<a href=\"query-html?"
        "sortby=date&sortorder=descending&country=Surinamé" .
    "&type=document&author=cia .
    "$exclude01=dl991122&exclude02=d3010304" .
    "&find=tropical+rain+forest+conservation">Find</a>
```

```
$link = "<form action=\"query.html\" method=\"post\">"
    . "<ínput type=\"hidden\"
    name=\"sortby\"           name=\"sortorder\" value=\"descending\"
    value=\"date\">"          name=\"country\" value=\"Suriname\">"
      '<input   type=\"hidden\"name=\"type\" value=\"document\">"
      '<input   type=\"hidden\"name=\"author\" value=\"cia\">"
      '<input   type=\"hidden\"name=\"exclude[]\" value=\"dl991122\">"
      '<input   type=\"hidden\"name=\"exclude[]\" value=\"d3010304\">"
      '<input   type=\"hidden\"name=\"find\" value=\""
      '<input  type=\"hidden\"
      '<input type=\"hidden\"
         "tropical rain forest conservation\">
      '<input  type=\"submit\"  value=\"Find\">"
      '</form>";
```

HTML vagy tisztán PHP?

A PHP jól keverhető' a HTML-lel, de a kódot egyre nehezebb olvasni, és minden alkalommal, amikor visszatérsz HTML-be, vesztesz a PHP erejéből. Erőt és rugalmasságot szerzel, ha az egész űrlapot, az összes tag-et és paramétert tisztán PHP-ben írod. Jelen fejezet szkriptje létrehozza minden HTML-tag minden elemét, így teljes körű ellenőrzéssel bírsz a tartalom és a formátum felett.

Ahogy a böngészők egyre inkább támogatják a Cascading Style Sheets-t (CSS), úgy csökkentheted az oldalak tartalmának közvetlen formázását, de légy óvatos, mielőtt a jó öreg <font> tag-eket valami újabbra cseréled. Egyszer segítettem egy reklámügynökségnek bemutatót készíteni egy nagy cég nagyon nagy emberének. A főnök megnézte a prezentációt, felírta az ügynökség referenciaoldalainak listáját, majd hazament a gyerekeihez vacsorára. Késő este a gyerekek öreg PC-jén a lassú modemmel megpróbálta a referenciaoldalakat megtekinteni, és azt találta, hogy bizonyos oldalak nem voltak olvashatóak. Persze, hogy nem az ügynökség nyert.

A sebesség és kompatibilitás problémájának megoldása különlegesen fontos az oldalad regisztrációs űrlapja esetében, hiszen a potenciális vásárlók ezt látják és használják legelőször. Vannak oldalak, amelyek vásárlók ezreit veszítik el, mert a regisztrációs űrlapjuk nem minden böngészővel működik. Sok cég az oldalaikon lévő gyatra hibanaplózás, a gyenge kimutatások és elemzések miatt soha nem fogja megtudni, hogy hány vásárlót veszít el és miért. Amikor regisztrációs űrlapokat, lekérdezési űrlapokat, visszacsatolási űrlapokat és problémajelentő űrlapokat hozol létre, feltétlenül győződj meg arról, hogy azok könnyen használhatóak és minden böngészővel bombabiztosán működnek.

A legkeményebb csatát a marketingosztállyal kell majd megvívnod, hogy a zavaró és tolakodó kérdéseket, figyelemelterelő képeket és a nyakatekert szövegeket távol tartsd az űrlaptól. Nem tudom, miért, az emberek szívesen tesznek fel olyan kérdéseket online, amire nyomtatott űrlapon egyébként ritkán válaszolnak a vásárlók. Sőt az űrlap előállítói adott kérdéseket kötelezővé tesznek, és utána csodálkoznak, miért nem töltik ki az emberek az űrlapot.

Az online űrlapokon keresztül történő navigációt nagymértékben befolyásolja, hogy milyen gyorsan tudod az oldalakat megváltoztatni. Ha nem tudsz gyors változtatásokat végre-

hajtani, az oldalak használhatatlanok és félrevezetők lesznek. Lesznek olyan vezetők és marketingesek, akik 10 perccel azután kérnek azonnali változtatásokat, hogy a tesztcsapat hazament. A kérések lehetnek olyan egyszerűek, mint egy rendkívüli ajánlat törlése, mert a cég kifogyott a termékből, de olyan fontosak is, mint egy oldal azonnali eltávolítása, mert annak tartalma törvénybe ütköző, és naponta több ezer dolláros bírság jár érte. Néhány évvel ezelőtt a liftnél vártam, hogy hazamenjek, amikor üvöltést hallottam, így elindultam kinyomozni, mi lehet a baj. Egy gyorsan növekvő online-értékesítő oldal fiatal és briliáns marketingmenedzsere úgy nézett ki, mint aki azonnal szívrohamot fog kapni, mert a vadonatúj termékük, a cigaretta, a megfelelő jogi figyelmeztetés nélkül volt az oldalakon megjelenítve. Szerencsére úgy terveztem meg a weboldalt, hogy gyorsan meg lehessen változtatni, így a problémát hamarabb megoldottam, mint ahogy a lift megérkezett.

Az egyik dolog, amit a fejezetben található kód tanít neked, az a következetesség. Minél következetesebben alakítod ki a kérdések és űrlapok formátumát és elrendezését, annál egyszerűbb lesz ügyfeleidnek a honlapodon az űrlapokat kitölteniük. A következetesség hosszú távon is meghozza az eredményét, és jó ok arra, hogy mindent PHP-kóddal hozzál létre, így elkerülheted a kósza HTML-különcségeket.

A fejezetben található kódok egy része többválaszos kérdéseket kezel. Bármikor, amikor több válaszlehető'ség van, mérlegeld a szabad formátumú válaszadás lehetőségét. Például amikor az első amerikai weboldalak próbáltak meg tengerentúli országokban értékesíteni, a rendelési űrlapon meg kellett adni a vásárló címét, de a kiválasztható államok között csak amerikai államok voltak. így a világon élő 5,9 milliárd ember közül 5,6 milliárd nem tudta rendesen beírni a címét. Napjainkban a legtöbb honlap már jobban működik, így olyan opció is választható, mint másik állam, egyik sem, a irányítószámok nem csak öt jegyűek lehetnek, vagy egyáltalán nem kell irányítószámot megadni, és a telefonszámoknál hosszabb körzetszámok is megadhatóak. Szerencsére ebben az e-mail-es világban faxszámot már egyáltalán nem kell megadni.

A minimális HTML

A HTML-űrlapoknak szükségük van:

- Kezdő tag-re: <form>
- Valamilyen adattag-re, mint az **<input>**
- Műveleti tag-re, mint az < **input type** = **"submit">**
- Végtag-re: **</form>**

Ezeket a tag-eket helyezheted a lap tartalmának elrendezésére szolgáló szerkezetek, így táblázatok celláin belülre, vagy a **<form>** tag-gel az egész táblázatot tedd keretbe és az egyedi **<input>** tag-eket helyezd az egyedi cellákba, mint azt a következő példa mutatja:

```
<form action="page.html"
method="post"> <input type="text"
name="anything"> <input type="submit"
name="doit"> </form>
```

A <form>-tag több böngészőben is megzavarhatja a szóközöket. Egy űrlap kezdete bizonyos böngészőkben hasonló sortörést szúr be, mint a táblázatok. Az űrlap vége is szúrhat be sortörést vagy szóközt.

Az űrlapok tartalmazhatnak további formázási elemeket, például az űrlap egyes elemeinek az elrendezését szabályozó táblázatot, bekezdést és a HTML-formázók széles választékát! Vannak szabályok arra vonatkozóan, hogy mi kerülhet az űrlapba, és mit lehet az űrlap köré helyezni. A legtöbb böngésző megszegi ezeket a szabályokat - van, amely sortörést szúr oda, ahova nem kellene, van, amely figyelmen kívül hagyja az összetett formázási lehetőségeket, és van, amely a rosszul formázott HTML-t is elfogadja. Azt javaslom, tar-tózkodóan állj a kérdéshez, és csak az űrlap logikus és olvasható elrendezéséhez minimálisan szükséges HTML-t használd, mivel a kedvenc böngésződ elfogadhat olyan hibákat, amelyek egy másik böngésző használatakor tönkreteszik az oldalaidat.

Problémáid adódnak, ha az űrlapot másmilyen HTML-tag szinten fejezed be, mint amelyik az űrlapot elindítja. Például ha egy űrlapot táblázaton kívül kezdesz el, és a táblázat egyik cellájában fejezed be, némely böngésző össze fog zavarodni. A szerkezetet egységessé és szimmetrikussá kell tenned. Én általában létrehozom az <input>-ot, majd először körbeveszem <td>- és </td> tag-ekkel, aztán <tr>- és </tr> tag-ekkel, majd mindezt táblázattag-ekkel, és végül az egészet űrlaptag-ekkel. Ha az űrlapot az oldal egy részére kell elhelyeznem, az űrlap köré egy külön táblázatot teszek.

lli

Könnyebbé teheted az űrlapok és táblázatok szimmetrikus elrendezését és egymásba ágyazását, ha minden egyes layert függvényként hozol létre, ezt követően pedig ezeket a függvényeket ágyazod egymásba. Böngészők és böngészőverziók tucatjaival tesztelem az oldalakat, és a top 30-as listámon levő összes böngésző megbízhatón kezeli az űrlapokat, ha az űrlapok elemei a többi HTML tag-hez képest megfelelő szinteken vannak létrehozva. Az egymásba ágyazott PHP-függvényekkel sokkal megbízhatóbban lehet űrlapokat létrehozni, mint a Dreamweaverrel vagy más drága honlapkészítő programmal.

Bell és Whistle

A HTML-ben nincsen <bell>-tag, a PHP-ban pedig nincsen whistle()-függvény. Gondosan mérlegelned kell, hogy az űrlapot milyen bonyolulttá teszed csupán annak érdekében, hogy az megfeleljen a marketingrészlegnek is. A PHP-ban egyszerű függvények, tömbök, és összetett objektumok segítségével építhetsz fel egy űrlapot. Amit te akarsz, az nem más, mint hogy a legegyszerűbben megváltoztathasd az űrlap formáját, amikor a tesztelők jelzik az űrlappal kapcsolatos problémákat.

A megfelelő megközelítés a formázás, a sorrend és az adatok elkülönítése, így a kérdések sorrendjét a formázás megváltoztatása nélkül tudod módosítani, és a kérdésekben is bárki ki tudja a szavakat cserélni anélkül, hogy az egész oldalt át kellene szerkeszteni. A kérdések sorrendjét tárolhatod egy tömbben, és végigfuthatsz a tömbön, ahogy azt a következő példa mutatja. (A 3. fejezetben teljes részletességgel esett szó a tömbökben használt listákról.) A kérdések sorrendje egyszerűen megváltoztatható a tömbelemek felfelé vagy lefelé való mozgatásával:

```
$questio [] = "name";
$questio [] = ■ "address";
$questio [] = ■■ "city";
$questio [] = ; "country";
   whileflist , $v) = each($question))
  print ( "<br>The question is: " . $
  }
```

ötlet: A fenti kódban használhatsz for()-ciklust is, de a while(each()) ciklusnak számtalan előnye van. Például akkor is helyesen adja vissza a következő tömbelemet, ha a cikluson belül új elemet szúrsz be vagy törölsz a tömbből.

Ha összetettebb neveid vannak, egyszerűbb lehet az elemeket a számuk alapján azonosítani. Amikor egy listát dolgozol fel, használhatod az elem indexszámát arra, hogy az elemet újra megtaláld a tömbben. Gondold végig a következő, űrlapba kerülő elemek listáját:

```
$fruit[] = "Akee";
$fruit[] = "Black Sapote";
$fruit[] = "Durian";
$fruit[] = "Guaran*";
$fruit[] = "Otaheite Gooseberry";
```

A guaran^: a koffein egy lehetséges forrása, és a benne levő ékezetes karakter alkalmas arra, hogy összezavarja az embereket, amikor egy szövegdobozba vagy kódba próbálják a nevét begépelni. Az ilyen nevet listából kiválaszthatóként kell megjeleníteni, hogy ne kelljen a karakterek feldolgozásával foglalkozni. Amikor ezt a listát kiválasztható elemekké konvertálod, azonosítóként a listaelem számát használd.

A PHP tömbfeldolgozását használva futtass ciklust a tömblistán. Először állítsd a tömböt az elejére, majd az elemek egyenkénti feldolgozására használd a **while()-, HstQ-** és **each()-**függvényeket. A „Gyors megoldások" részben számtalan példát mutatok erre. Az elsődleges cél az egyéni névazonosítóként való használatának elkerülése úgy, hogy az egyes listákat egy egészként dolgozzuk fel, illetve elkerüljük az olyan kódokat, amelyekben az egyéni elemeket azonosítani kell.

Ne felejtsd el, hogy a PHP-tömbök megkülönböztetik a kis- és nagybetűket, így a **$fruit["Apple"]** nem azonos a **$fruit["apple"]-vel.** Légy óvatos a megjelenítéshez és azonosításhoz használt nevek keverésével. Szeretem a következő trükkös kódot, mert eltávolítja a megjelenítési nevekből a szóközt, és mindent kisbetűre állít:

```
$display_name = "Macadamia nuts";
$id = strtolower(str^replace(" ", "", $display_name));
```

A szöveg különböző nyelvű oldalakon történő megváltoztatásához nézd meg a 12. és 20. fejezetben, hogyan lehet a szöveget strukturált formában bevinni. Mindkét fejezetben több módszert is találsz arra, hogyan használhatod a más emberek által szerkesztett kiterjedt szöveglistákat, illetve technikákat, amelyek jól jöhetnek az űrlapkérdéseidnél. Miután a sorrendet és a formázást elkülöníted a kérdések tartalmától, koncentrálhatsz az oldal felépítésére és az olyan témákra, mint a feltett kérdések típusa.

Űrlapkérdések

A HTML-lel számtalan módon lehet kérdéseket feltenni, és válaszokat kapni az egyszerű bejelölődoboztól és rádiógombtól kezdve a kiterjedt kijelölési listákig. Olyan kérdést akarsz feltenni, amelyet a felhasználó könnyen megért, és amely a felhasználó válaszadási lehetőségét adott válaszlehető'ségekre korlátozza. A kérdéstípusokat keverheted is, például rádiógombokkal megadsz három lehetséges választ, egy üres szövegmezőbe pedig megjegyzést írhatnak a felhasználók - ha kívánnak.

Itt van példának néhány az űrlapokkal kapcsolatos gyakori hiba, amelyeket az ázsiai és csendes-óceáni piacra betörni szándékozó amerikai cégek weboldalain találtam:

- rögzített kijelölési lista az államok neveivel, az ausztrál államok bejelölésének lehetősége nélkül;
- az állam kötelező megjelölése, figyelem kívül hagyva, hogy sok országban nincsen az amerikai államoknak megfelelő fogalom;
- nem minden országban van irányítószám;
- nem minden országban ötjegyű az irányítószám;
- minden országnak megvan a maga sajátossága, ami a nemzetközi oldalak űrlapjainak tervezését érdekessé teszi.

Eldöntendő kérdések

A következő kérdésekhez jól alkalmazhatók a HTML-rádiógombok:

- Szokott halat enni? Igen/Nem
- Szokott tejterméket enni? Igen/Nem
- Ablak melletti ülést szeretne? Igen/Nem

A rádiógombok akkor használhatók a legjobban, ha a több lehetséges válasz közül csak egyet lehet megjelölni. A böngészőben a kérdés úgy jelenik meg, ahogy a 9.1. ábra mutatja.

Do you eat fish? Yes'[1] No^

9.1 ábra Igen/Nem kérdés rádiógombok használatával

A válasz pedig úgy, ahogy a 9.2. ábra.

The value of $fish is yes

9.2 ábra Válasz egy Igen/Nem kérdésre

Mi történik a HTML- és PHP-kódban? A HTML a következő:

```
<form action="radiobutton.html" method="post">
Do you eat fish?    Yes
<input type="radio" name="fish" value="yes" checked>
```

```
Snbsp;Snbsp; No<input type="radio" name="fish" value="no">
Snbsp;Snbsp; <input type="submit" value="Test">
</form>
```

Vedd észre, hogy mindkét rádiógomb neve fish, és mindkettőhöz van érték rendelve. Az alapértelmezett érték, amely ebben az esetben a yes, be van jelölve. A PHP a kiválasztott értéket a fish-nek nevezett mezőben adja vissza, és a példában az érték a következő kóddal van megjelenítve. Figyeld meg a fish létezésének ellenőrzését - if(isset($fish)). Ez olyankor fontos, amikor az űrlapoldalak saját magukra mutatnak az eredmények vagy a hibák megjelenítésére. Tegyél megjegyzést az oldal tetejére, mely elmagyarázza, mi jelenik meg az első belépésnél, mi jelenik meg hiba esetén, és hogy hova jut a felhasználó, ha az elküld gombra kattint:

```
if(isset($fish))        {
   print("The value of \$fish is " . $fish);
```

Egy kérdésre annyi válaszlehetőséget adhatsz, amennyit csak akarsz, feltéve, hogy minden válasznak ugyanaz a neve. Ha kihagyod a name-mezőt, némely böngésző az előző rádiógomb nevét fogja használni, de nem hiszem, hogy az összes böngésző ezt tenné.

Ha kihagyod a value-mezőt, a PHP az értéket on-ra állítva adja vissza. Adhatsz az egyes válaszoknak más és más nevet, és rákereshetsz az on értékű mezőre, de ez többválaszos kérdéseknél bonyolult lehet. Ezen nehézség illusztrálására a következő kódrészlet két sort tartalmaz a halas-kérdéses példából, az első < input> tag-nak a fishyes-t, a másodiknak a fishno-t adva, hogy az egyes mezők jelenlétét tesztelhesd. Ez a fajta megközelítés egyszerűbbnek tűnik, amikor a kérdésekre kevés válaszlehetőség van, de nehezen kezelhetővé válik, ha a válaszok száma több:

```
<input type="radio" name="fishyes" value="yes" checked>
   No<input type="radio" name="fishno" value="no">
```

A name-mezőt minden egyes kérdésnél különbözőre állítsd, és győződj meg arról, hogy a név nincs keveredésben egyetlen másik PHP-változóval sem. A legjobb technika talán a kérdés sorszámának, a $k-nak a használata, ahogy a következő fish-példában mutatom. A nevet egy közös előtag használatával hozod létre, ami a példában a question. Az űrlapból visszajövő mezők questionO, question 1, question2 stb. nevet kapnak:

```
$question = "<input type=\"radioV"
   . " name=\"question" . $k .
   "\"" . " value=\"yes\"
   checked>";
```

Egy vagy több válasz lehetséges?

Sok olyan kérdés van, amelyre egy, több vagy egyetlen válasz sem adható, olyan kérdések, melyekre használható a HTML <select>-tag. A <select> tag-gel egy választ lehet megjelölni, de beállítható úgy is, hogy többet lehessen megjelölni, jóllehet nem biztos, hogy a többszörös választás minden böngészővel és operációs rendszerben működik. íme példaként néhány kérdés, és a kód, amelynek eredménye a 9.3 ábrán látható lista:

- Mik a kedvenc gyümölcseid?
- Mit csomagolsz, amikor Sydney-be utazol?
- Válaszd ki a kedvenc Copper sörödet!

```
print("<form action=\"query.html\" method=\"post\">"
    . "<select name=\"favorite\"Xoption>Sparkling Ale</option>"
      "<option>Pale Ale</optionXoption>Dark Ale</option>" .
    "<option>Stout</option><option>Premium Ale</option>" .
    "<option>Vintage Ale</option><option>Old Stout</option>" .
    "<option>Special Old Stout</option>" . "<option>Genuine
    Draught</option>" .
    "<option>Light</optionXoption>DB</optionX/selectx/form>")
```

9.3 ábra Válaszd ki a kedvenc sörödet a <select> használatával

Gondolj végig egy olyan helyzetet, amikor a vásárló nem tud egy kérdésre válaszolni. Szeretem az olyan szoftvercégeket, amelyek megkövetelik, hogy használat előtt regisztráld a terméket, majd a regisztrációs oldal közepén megkérdezik a termékről a véleményed. Az egyetlen őszinte válasz a következő lehet:

Eddig a termékük az időmet rabolta és semmi hasznom nem származott belőle!

Ugyanez a probléma merül fel, amikor egy bolt gyerekkocsit ad el neked, és megkérdezi, hogy hány gyereked van. Lehet, hogy még a tervezési fázisban vagytok, vagy csak a kutyádat akarod kényeztetni.

A semmi, az érvényes válasz. Engedd, hogy az emberek kihagyják az olyan kérdéseket, amire nem tudnak válaszolni, mentsd el a többi választ, és gondolkodj el azon, hogyan tudnád a problémás kérdést feltenni legközelebb, amikor az emberek a honlapodon járnak.

Amikor az űrlapból megjönnek az eredmények, egy csomó változót kapsz, amelyek egy igen/nem válasszal vannak beállítva, vagy pedig egy tömböt, amely a kiválasztott elemek listáját tartalmazza. Te választhatod ki, hogy pontosan mit akarsz, mi felel meg a programozási stílusodnak. En általában a listamegközelítést alkalmazom, mert így könnyen adhatok hozzá új elemeket.

Gondold végig a következő űrlapot, amely a következőket kérdezi:

- Szereted a banánt?

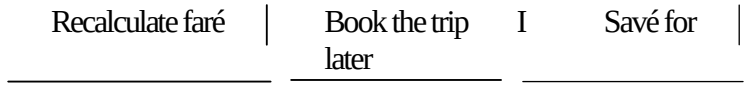
- Szereted a narancsot?
- Szereted az almát?

A kérdések listája hamar unalmassá válik, ezért változtasd meg a formátumot, hogy a „Szereted" kérdés fejrészként jelenjen meg, ne a gyümölcsök egyszerű listájaként. A PHP-kódodban az űrlap visszaadhatja a $bananas-t igaz vagy hamis értéket tartalmazva, a Soranges-t szintén igazként vagy hamisként és így tovább. Jobban szeretem azonban a listát tömbben megkapni, amely tartalmazza a kérdéslistába kerülő gyümölcsöket, mintha egy másik tömbben a kiválasztott gyümölcsök egyszerű listáját kapnám. Bárki újabb gyümölcsöt adhat hozzá anélkül, hogy a kód egyetlen sorát meg kellene változtatnom.

A fejezet „Gyors megoldások" részében találsz arra is példát, amikor a vásárlók egyetlen elemet választhatnak a listából, és arra is, amikor többet. Nézd végig a listát, hogy pontosan milyen kérdésre van szükséged, majd próbáld ki a kódját.

Összetett műveletek

Az űrlap elküld, visszaállít vagy töröl gombbal végződik, így a vásárlód mondhatja, hogy „Gyerünk, számlázz nekem", vagy „Töröld a rendelést". Nem vagy erre a két opcióra korlátozva, összetett műveleteket is végrehajthatsz, ahogy a 9.4 ábra mutatja.



9.4 ábra Többszörös elküld-gomb

A PHP-kódban egyszerű az összetett műveletek kezelése, mert minden egyes elküld gombtól különböző' mezőt kapsz. Ha az Ár újraszámolása gombot recalc-nak nevezed és a többit értelemszerűen book-nak, illetve save-nek, a szkripted a $recalc-, $book- vagy $save-változók valamelyikét kapja vissza, és az if(isset($recalc))-hez hasonló kód használatával a megfelelő műveletet választhatod ki.

Az első HTML-tag egy űrlapban a <form>, amelyik action="_page_name_"-et tartalmaz, ahol a _page_name_ annak az oldalnak a neve, amelyet a böngésző akkor kér, ha a felhasználó az elküld vagy visszaállít gombra kattint. A böngésző kiválasztja, hogy melyik oldalt kérje, majd a kérést elküldi a szervernek. Azt, hogy a böngésző melyik oldalt, tehát melyik szkriptet kéri, te döntöd el az oldal nevének a műveleti paraméterbe írásával.

Ugyanarra az oldalra akarod visszaküldeni a felhasználót vagy egy új oldalra? Ha ugyanarra az oldalra küldöd őt vissza, és a lap tetején elvégzed a feldolgozást, könnyedén megjelenítheted ugyanazt az oldalt és űrlapot, hibaüzeneteket beszúrva és további információkat kérve. Ha a felhasználót új oldalra irányítod, választhatod azt, hogy az űrlap adatainak feldolgozását egy rejtett oldalon végzed el, amely semmit nem jelenít meg, és átirányító fejlécet küldesz a felhasználó böngészőjének, hogy a felhasználót a megfelelő oldalra irányítsa.

Néha egynél több oldalra akarod az embereket irányítani, amit több űrlap egy oldalra való helyezésével oldhatsz meg. Gondold végig, mi történik, ha egy oldal csak a 9.4 ábrán látható három gombot tartalmazza. Mindegyik gomb egy teljesen különböző űrlapnak lehet az

elküldés gombja, egyedi kérdésekkel és műveletekkel. A Book The Trip (Az utazás lefoglalása) gomb egy a bankodnál lévő biztonságos oldalra mutathat, a Savé for Later (Ments el a későbbiekre) egy olyan oldalra mutat, amely felfrissíti a felhasználó profilját, a Recalculate Faré (Ár újraszámolása) pedig egyszerűen visszaugrik az előző oldalra, hogy új számítást végezhessen.

Az egyetlen probléma az összetett műveletekkel az adatkoordináció hiánya. Egy oldalon nem gyűjtheted össze az első űrlap adatait, ha a felhasználó a második űrlap elküldés gombjára kattint. Az összetett űrlapok használatához először össze kell gyűjtened az adatokat, majd egy egyedi oldalon kell a szétválasztott kiválasztó űrlapokat megjelentetned.

JavaScript

Sokszor a tapasztalatlan, a JavaScript-et először használó fejlesztők fejest ugranak a legördülő kiválasztási listába és a bonyolult adatellenőrzésbe. A honlap tulajdonosai később rengeteg pénzt fizetnek nekem, hogy ugyanezt a JavaScript-et eltávolítsam. Miért? Azért fizetnek engem, hogy megjavítsam az űrlapokat, amelyek nem minden böngészővel működnek, és ha úgy tűnik ís, hogy működnek, az oldal adatbázisa végül tele lesz érvénytelen adatokkal. Az én feladatom az, hogy az űrlap minél több böngészővel működjön, és a haszontalan adatokat távol tartsam az adatbázistól. Ehhez bizonyos lépéseket követek.

A csinos gördülőmenük eltávolítása

Az első lépésben a csinos JavaScript-gördülőmenüket kell eltávolítani. A normál HTML <select>-tag olyan gördülőmenüt biztosít, amely nem csak egyszerűen felépíthető a PHP-ban, de mindenki számára érthető is. Ami ennél bonyolultabb, az egyes embereket össze fog zavarni, és vagy rossz mezőbe bevitt vagy teljesen hiányzó adat lesz az eredmény. Gyakori tünet egy félig üres oldal. Ugyanis az oldal közepén egy JavaScript által szabályozott mezőre kattintva az össze nem-függő mezők tömkelege tűnik el.

Mezők érvényesítése a szerveren

A második lépésben minden mezőt érvényesítened kell a szerveren. Mindegy, mennyire jól csinálod meg a JavaScript-et, az emberek kijátsszák azt, és hibás adatokat küldenek. Az egyik trükk úgy működik, hogy lemásolják az oldalt, a másolatban megváltoztatják a JavaScript-et, és azután a másolat válaszol a böngészőből. Sok honlap használ befejezetlen és rosszul formázott JavaScript-et, amely csak Microsoft Internet Explorer alatt működik, az olyan böngészőkben pedig, mint az Opera, hibaüzenetet küld, és megengedi a böngészőt használó embernek, hogy eldöntse, a szkript folytatódjon vagy leálljon. Olyan adatot kaphatsz eredményül, amelyet a szkripted csak részben érvényesített, vagy amelyet teljesen egészében más valaki szkriptje hozott létre, így a saját PHP-kódodban mindent ellenőrizned kell, különösen az árakat és az végösszegeket. Ha a szerveren egy mezőt a PHP-vel ellenőrzői, nem sok időt takaríthatsz meg a böngészőben a mező JavaScript-tel való ellenőrzésével.

Ötlet: Ha egy JavaScript- vagy cookie-alapú bevásárló kosarat próbálsz hackeléssel módosítani, a végösszeg $5,000-ról $1,000-ra való csökkentése működhet, de a $10 nem -azt valaki mindig észre fogja venni.

Biztosan lesz olyan projekted, amikor valaki ragaszkodni fog a JavaScript használatához, hogy üresen hagyott kötelezően kitöltendő mezőkkel ne fogadja el az oldalt. Ha a potenciális vásárlókat a visszaküldött oldallal megállítod, lehet, hogy elhagyják az oldalt, és soha nem térnek vissza. Ha az űrlapot a szerverre küldöd és ott érvényesíted, megállapíthatod, hogy melyik mezőre nem válaszolnak a látogatók, és ezt az információt továbbíthatod az űrlap tervezői felé.

Hosszú űrlapok

A többoldalas űrlapok zavaróak, ha nincsen igazi ösztönző, amiért azt kitöltenénk. Egy nyomtatott kérdőíven a „Cukorbetegek részére" című szakaszt vagy kitöltőd, vagy átug-rod. Ennek az online megfelelője a „Cukorbeteg Ön?" kérdés az első oldalon, és a választól függően behívod vagy átugrod a cukorbetegek számára szánt kérdéseket tartalmazó oldalt.

A 2000. évi amerikai elnökválasztás floridai problémája azt mutatja meg, milyen könnyű az embereket az űrlapokkal megkavarni. Ez a probléma online még bonyolultabb az emberek számára, mert nem tudnak előre- vagy hátraugrani, és nem tudják az űrlap különböző részeit összehasonlítani. Ezért a különböző oldalakon ugyanazt az elrendezést kell követned, amely vizuális segítséget nyújt a kérdéshez és ahhoz, hogy milyen típusú választ várs. A következő rész arra vonatkozólag tartalmaz útmutatást, hogyan kezeld a többoldalas űrlapokat.

Hosszú űrlapoknál nagyon hasznos, ha a honlapod összes űrlapján az összes kérdést ugyanazzal a formázófüggvénnyel alakítod ki. így ha egy új látogató az egyik oldalon kitölti az egyik űrlapot, utána bármelyikkel képes lesz megbirkózni. A formázás és a kérdések különválasztásával, sokkal egyszerűbb a kérdések új sorrendbe rendezése és az oldalak ré-szekre bontása.

A hosszú űrlapok feldarabolása

Rengeteg online-bolt honlapján találhatunk példát a túlságosan hosszú űrlapokra, regisztrá ciós oldalakra, felmérésékre és játékra csalogató űrlapokra, amelyek elriasztják a potenciális vásárlókat. Az oldalnak egymáshoz kapcsolódó részekre kellene a hosszú űrlapot bontania, minden részben egy fontos kérdéssel, majd a részeket különböző oldalakon helyezni. H

Gondolj egy olyan élelmiszert árusító honlapra, amely szeretne a vásárlóknak hetente e-mailt küldeni, és ehhez tudni akarja a vásárló korát, jövedelmét és nevét. Sokan kitöltik az ilyen űrlapot, sokan egyszerűen otthagyják az oldalt, sokan pedig azt teszik, amit én, vagyis hamis, összevissza válaszokat adnak a rámenős és a tárgyhoz egyáltalán nem tartozó kérdésekre. Én soha nem kérek e-maileket, mert oly sok marketinges küldözget időpocsékoló, haszontalan leveleket; a korra a választható legkisebb vagy legnagyobb értéket jelölöm be, a jövedelmi szintre pedig egy olyat, ami teljesen irreális egy koplaló szerző esetében. A nemre vonatkozó kérdés néha megengedi, hogy az Egyébre kattintsak.

Mi történik, ha az e-mailre vonatkozó kérdést egy külön oldalon teszed fel, és mellékelsz egy levélmintát? A levél lehet érdekes és hasznos, ha például egy ételreceptet tartalmaz. így nagyobb a valószínűsége, hogy az olvasó pozitív választ ad a „Szeretne ilyen jellegű e-maileket kapni?" kérdésre.

Ahelyett, hogy közvetlenül kérdezed meg valakitől a korát, küldj neki egy oldalt, amely a honlap életkorhoz kötött funkcióját mutatja be, majd kérdezd meg a korát és egyéb személyes kérdéseket. Például az oldaladon, amely a bébiételeid hihetetlen széles választékát mutatja be, mosolygó csecsemők fényképét beszúrva tedd fel a „Segítsen nekünk, hogy segíthessünk a gyermekeit táplálni, kérem, mondja meg..." kérdést, és kérdezd meg az összes családtagjának az életkorát.

Próbáld ki, hogy egy oldalon egy témával foglalkozol, egy kis csábítással felteszed a kérdést, majd jöhetnek a kísérő anyagok. (Nem számít, ha ezek kifutnak a képernyőről.) Mindig jobb pontos választ kapni egy kérdésre, mint az összes kérdést feltenni, de pontatlan válaszokat kapni.

Az információ oldalról oldalra való továbbítása

Ha többoldalas űrlapjaid vannak, az egyik oldalról a másikra az információt rejtett mezőkkel, cookie-kal vagy session-rekordokkal továbbíthatod. Tipikus stratégia az első oldalon a felhasználó keresztnevének megkérdezése, hogy a nevét a következő oldalak tetején megjelenítve sokkal személyesebbé tedd az oldalt. A nevet legegyszerűbben egy rejtett mezővel továbbíthatod. A továbbításra a legjobb hely egy session-rekord - ha van session -, hiszen a session-rekordból a honlap összes oldala elérheti a nevet.

Rejtett mezők használata

A rejtett mezőket, **<input type="hidden"**>> könnyen lehet a hosszú listákhoz hasonló dolgokkal használni. Tudni kell azonban, hogy ha az űrlapot kitöltője félig kész állapotban otthagyja, később visszatér és megpróbálja folytatni, a rejtett mezők nem működnek. A cookie-k segíthetnek ezen, de azoknak is megvannak a korlátjaik.

Cookie-k állítása az összes oldalra

Ha arra kérsz egy felhasználót, hogy töltsön ki egy többoldalas űrlapot, az első oldalon bevitt adatok egyenesen az adatbázisba kerülhetnek, majd rejtett mezőkön a második oldalra továbbíthatók. Amikor egy vásárló profilját frissíted fel, az adatokat azonnal a profilba mentheted, és semmit sem kell egyik oldalról a másikra továbbítanod. Amikor azonban egy új vásárló profiljához gyűjtesz adatokat, semmit nem célszerű addig elmenteni, amíg az utolsó oldalon a vásárló a beleegyezéseként az elküld gombra nem kattint.

Az új profil részére az adatokat rejtett mezőkkel továbbíthatod oldalról oldalra, de ha egy többoldalas űrlap kellős közepén új oldalakat kell beszúrnod, akkor az összes utána következő oldalra új rejtett mezőket kell beszúrnod, ami nagyon unalmas lehet és hibákat okozhat. A cookie-k egyszerűbbek, mert egy cookie-ba bármit bedobálhatsz, és minden egyes oldalhoz új cookie-t állíthatsz be. Csak a sorrendben utolsó szkriptnek kell az egyes cookie-kat végignéznie. A cookie-knak megvan az az előnyük, hogy éjszakára a vásárló gépén maradhatnak, aki másnap folytathatja ugyanannak az űrlapnak a kitöltését.

A cookie-k problémája akkor jelentkezik, amikor egy hosszú űrlapnak már több oldalát felépítetted. Belefutsz a cookie-k korlátaiba, és az egész designt újra kell gondolnod. Minden böngésző más szabályokat követ a cookie-kra, tehát ez csak útmutató. A cookie-specifi-káció azt mondja ki, hogy domainenként csak 20 cookie-t használhatsz, ami eleve kizárja,

hogy egy 21 oldalas űrlapot hozz létre oldalanként egy cookie-val. A cookie-k maximális mérete 4KB, így egy hosszú űrlap összes információját nem gyűjtheted ki egy cookie-ba.

Szintén problémát okoz, ha a vásárlóid kikapcsolják a cookie-kat, vagy úgy állítják be a böngészó'jüket, hogy az esemény végén elfelejtse őket, esetleg olyan szoftvert használnak, amely naponta törli a cookie-kat. A cookie-k természetesen akkor sem működnek, ha a vásárló a munkahelyi számítógépén kezd el kitölteni egy űrlapot, és az otthoni PC-jén próbálja befejezni. Belátható, hogy a cookie-kat azok korlátai miatt inkább csak session-azo-nosítók tárolására érdemes használni, segítségükkel aktívan tarthatók a megkezdett session-ök.

A session-öket a 19. fejezetben mutatom be, de íme egy gyors lecke, hogy mit kell tenned (a PHP 4.0.5 alapján). A cookie-k használatához be kell kapcsolnod a use_cookies-t a php.ini-ben (ha lehetséges), a use_trans_sid automatikusan az URL-be kell tegye a session-azonosítót, ha a cookie-k nem elérhető'k, a rewriter.tags pedig beállítja a HTML tag-paraméterek listáját, amely az URL-eket továbbítják. Ezzel már elindulhatsz egy tesztoldalon:

```
session.use_cookies = 1
session.use_trans_sid = 1
url_rewriter.tags =
    "a=href,area=href,frame=src,input=src,form=fakeentry"
```

Session-rekordok használata

A session-rekordok bármilyen méretűek lehetnek és akárhány bejegyzést tartalmazhatnak. Egy modern adatbázisban szöveges mezőben vannak eltárolva, és maximum négy milliárd karaktert tartalmazhatnak, ami azt jelenti, hogy egy űrlap nagyobb lehet a figyelmem érték-tartományánál és a gépelési képességeimnél. Ha egy session-rekord egy modern operációs rendszer külön fájljában van, a rekord több terabájtos is lehet. Egyetlen session-rekordban tárolni lehet az összes ember által a világ összes kitöltött űrlapjából származó szöveget. A honlapok oldalai közti kommunikáció ideális megközelítése a session-azonosító cookie-ban, az összes többi információ pedig session-rekordban való tárolása.

A session-rekordok problémája ideiglenes természetükből adódik. Amikor a böngészőből kilépsz, a session-rekordok eltűnnek, ha elmész ebédelni, túllépik a rendelkezésre álló időt, és a session-azonosító oldalról oldalra való továbbítása miatt a cookie-któl vagy URL-ktől függenek. Nem egyszerű egy session-höz újra csatlakozni, hiszen egy egyszerű módszer biztonsági kockázatokat okozna, lehetővé téve más embereknek, hogy a session-öddel ka-lózkodjanak. Éppen emiatt a felhasználók nem tudják könyvjelzővel megjelölni egy session- egy oldalát, hogy másnap ugyanonnan folytathassák.

Ha meg akarod engedni, hogy az emberek másnap ugyanahhoz a session-höz csatlakozzanak, be- és kijelentkezési rendszer kell létrehoznod felhasználói névvel és jelszóval. A session-ük összes információját el kell mentened egy felhasználói profilban, és másnap a profil tartalmát felhasználva kell új session-t létrehoznod. Elméletben használhatod újra ugyanazt a session-azonosítót, de ezzel lehetővé teszed, hogy a session-ödbe belenyúljak.

ötlet: Hogyan kalőzkodj a gyengén programozott oldalak session-jein: Amint a kollégád hazamegy, vizsgáld meg a cookie-fájljait (a Netscape-ben cookies.txt), hogy kikeresd az oldal nevét és a session-azonosítót. Kapcsold ki a cookie-kat a böngészőjében, így kényszerítve az oldalakat, hogy URL-t használjanak. Jelentkezz be ugyanarra az oldalra bármi-

lyen más névvel, majd vágd ki és másold a kollégád session-azonosítóját a te URL-edbe. A session-ök akár több óráig a szerverfájlban maradnak, így általában rengeteg időd van mindenre.

Használható hosszú űrlapok tervezése

A emlékezőképességem két oldalnyi. Másként megfogalmazva, amikor a második oldalt írom, vissza tudok emlékezni, mit írtam az első oldalon, de amikor a harmadik oldalt írom, már nem. Ezt a tartományt tekintem mérvadónak, és amikor űrlapokat tervezek, figyelembe veszem a mások hasonló jellegű képességét. Ha a felhasználói profilhoz hasonló űrlapot kell létrehoznom, az a cél vezet, hogy a felhasználó az űrlap elején azonosítsa magát, és ezt használja a következő oldalak alapjaként.

Sok kicsi sokra megy

Ha valaki addig nem lehet új vásárlód, amíg egy hatalmas űrlapot ki nem tölt, potenciális vásárlókat fogsz elveszíteni. Azt javaslom, hogy az első oldalon csak a vásárló nevét és e-mail-címét kérd, adj neki felhasználói nevet a belépéshez, a marketing célú kérdéseket pedig hagyd későbbre.

Úgy tervezd meg az űrlapot és az azt támogató kódot, hogy a döntő fontosságú e-mailcímet azelőtt elmentsd, mielőtt a látogatót az agresszív vagy unalmas kérdéseiddel felizgatnád. Alakítsd ki úgy az oldalt, hogy félúton abbahagyhassák a regisztrációt, és később a felhasználói nevükkel visszatérhessenek.

Fizesse meg a marketing a zaklatást

Ha a marketingmenedzser tudni akarja az ügyfél életkorát, akkor ajánljon fel ösztönzőt vagy valamilyen nyereményt. A regisztrációs oldalon a legszükségesebb kontaktadatokat szerezd be, a vásárló ismerje meg a belépési információkat, és csak akkor tegyél fel neki további kérdéseket, ha már belépett. Halvány fogalmam sincsen, miért kell egy irodaszert értékesítő oldalon a nememre rákérdezni - ha férfi vagyok, kék tollat küldenek, ha nő, akkor rózsaszínt?

Ha az oldaladon nem alkoholt, cigarettát vagy pornográf dolgokat árulsz (illetve semmi olyat, amihez a törvények minimális életkort ír elő), nincs szükséged a vásárlók életkorára ahhoz, hogy vásárolhassanak. Az életkorra való kérdés lehet egy különleges bónuszkérdés, melynek megválaszolásáért valamilyen ösztönző jár, például egy ingyenes ajándék a következő szállításban. Te megkapod a korát, ő pedig ösztönzést kap, hogy vásároljon valamit ahhoz, hogy megkapja az ingyenes ajándékot - ösztönzőket venni mindenkinek jó üzlet.

Ugyanez igaz a jövedelemre és nemre vonatkozó kérdésekre. Engedd a felhasználót akkor is regisztrálni, ha nem válaszol ezekre a kérdésekre, majd adj neki ösztönzést, hogy mégis megtegye. Ahhoz, hogy őszinte választ kapj, győződj meg róla, hogy az ösztönző nem indukál torzítást. Ha a fiataloknak szánt ösztönző az X-akták DVD-n, az idősebbeknek pedig a Moulin Rouge DVD-n, akkor azt mondanám, hogy 95 éves vagyok!

Adj visszajelzést

Ha már van valamilyen információd, amellyel azonosíthatod az űrlapot kitöltő felhasználót, helyezd ezt valahová a lap tetejére, ezzel is emlékeztetve a felhasználót arra, hogy ő már be-

lépett. Amint kitölti az űrlap újabb és újabb szakaszait, adj neki visszajelzést, valami olyat, mint az Amazon nyomon követő listája, ahol a felhasználó látja a már kitöltött és a még hátralévő részeket. A kevésbé jól tervezett űrlapok gyakori hibája, hogy vannak olyan felhasználók, akik többször is regisztrálnak, vagy mert elfelejtették, hogy már regisztráltak, vagy mert azt hiszik, hogy a regisztráció nem sikerült.

Egy tömbbel könnyen kilistázhatod a felhasználó nevét és a kitöltött űrlapszakaszokat, majd tedd ezt az információt az űrlap egyik sarkába. Gyakran a következőhöz hasonló tömb alapján tervezem az egész űrlapot:

```
Selements[] = array("name" => "firstname", "length" => 60,
    "question" => "First name", "complete" => falsé);
$elements[] = array("name" => "lastname", "length" => 60,
    "question" => "Family name", "complete" => falsé);
$elements[] = array("name" => "address", "length" => 60,
    "question" => "Address", "complete" => falsé);
```

Az űrlap felépítéséhez ciklussal végigfutok a tömb elemein, oldalanként 10 kérdést állítva be, és az egyes kérdéseket akkor tekintem befejezettnek, ha megvan az információ. Ha hiba van egy mezőben, a tömbben a mezőnek megfelelő elemhez egy hibaelemet szúrok be. Az űrlapot kezelő oldalon a tömböt egyszer session-változóként kezelem, majd a feldolgozás végén egy ciklussal végigfutva a tömbön, az adatbázisba mentem a tartalmát. Tovább finomíthatod az űrlapot egy „mindig mutat" elem hozzáadásával, amely jelzi az összes oldalon ismétlődő fájlokat, például a vásárló nevét.

A meg nem kérdezett és meg nem válaszolt kérdések közötti különbség

Ha az adat már az adatbázisban van, az adatbázis eszközeivel a mezőket jelölheted üresnek, ismeretlennek vagy nullának. MySQL-ben a sztringmező a meződefiníciótól függően adhat üres vagy nulla hosszúságú sztringet (a részleteket az 5. fejezetben találod). Ezt olyan kérdéseknél használhatod, mint például a „milyen hobbijai vannak", és a MySQL nullját fordítsd a PHP false-ra, jelezve azt, hogy a hobbimezőt még nem kérdezted meg, és akkor kell a vásárlótól megkérdezni, amikor legközelebb olvassa a profilját. Hasonlóképpen a MySQL üres sztringet a PHP-be fordíthatod ""-re, jelezve azt, hogy a hobbira vonatkozó kérdést már feltetted, de még nem válaszoltak rá semmit sem. Ha túl bonyolult a null/false és "" megkülönböztetése, ahogy néhány adatbázisban és a PHP3-ban az volt, hozz létre egy külön mezőt, amely jelzi, hogy feltetted-e már a kérdést. Akár egy dátummal is jelölheted, hogy mikor tetted fel a kérdést, és évente újra felteheted.

Fájlok feltöltése

Az űrlapokkal fájlokat is feltölthetsz gépedről a webszerverre. Van néhány apró különbség a feltöltő és a hagyományos űrlapok között, és ezen túlmenően a két eltérő űrlaptípus máshogy működik és mást jelenít meg a különböző böngészőkben és operációs rendszerekben. A PHP leegyszerűsíti a szerver feldolgozási feladatát, de nem segít abban, hogy a felhasználóknak elmagyarázd, hogyan töltsenek fel a böngészőjükkel és operációs rendszereikkel fájlokat.

Az első dolog, ami a feltöltéshez kell, egy űrlaptag, amely beállítja az enctype-paramétert. Másold be ezt a példát, vagy ragadd meg a *HTML Black Book-ot* (The Coriolis Group, Inc., 2000) és nézz utána az enctype-nak. Az űrlaptag így néz ki:

```
<form enctype="multipart/form-data"
action="apage.html" method=\"post\" >"

<input type="hidden" name="MAX_FILE_SIZE" value="200000">
```

A fájlnév beírására egy szövegdobozra van szükséged, és a HTML a file-típust használva egyedi fájlnév szövegdobozt kínál a kiválasztási eszközzel. Ebben a tag-ben a size a fájlnév szövegdobozmérete, nem pedig a fájlé. így néz ki:

```
<input type="file" name="uploadfile" size="60">
```

A szokásos elküld tag-gel és az űrlap végtag-jével kell lezárnod, ahogy itt látod:

```
<input type="submit" value="Upload"></form>
```

A PHP beállító fájlja, a php.ini számtalan olyan paramétert tartalmaz, amely korlátozhatja, hogy mit tölthet fel az űrlap. A file_uploads-t on-ra kell állítanod, az upload_tmp_dir-nek a megfelelő könyvtárra kell mutatnia, az upload_max_filesize-nak pedig legalább akkorának kell lennie, mint a legnagyobb feltölteni kívánt fájl mérete. Bizonyos fájlműveleteket a PHP-ra allokált memória mérete korlátoz, így állítsd a PHP memory_limit-jét az upload_max_filesize-nál nagyobbra. Ezt a következőképpen teheted meg:

```
file_uploads = On
upload_tmp_dir = T:\upload
upload_max_filesize = 8M
memory_limit = 30M
```

A file_uploads lehetővé teszi, hogy a PHP fájlfeltöltéseket végezzen. Sokak szerint ez biztonsági kockázatot hordoz magában, és van, aki az erőforrás-használat elfogadhatatlan módszerének tekinti. Bárhogyan is döntesz, a fájlfeltöltéseket korlátozd azáltal, hogy csak olyan felhasználóknak engedélyezed, akik beléptek, így tudod tájékoztatni őket a feltöltés megfelelő módjáról. Képzeld el azt problémát, ha egy webszervert szexképekkel vagy lopott Nirvána-dalokkal árasztanak el.

Amikor valaki feltölt egy fájlt, az az upload_tmp_dir-ben megadott ideiglenes könyvtárba kerül, így mielőtt a fájlt áthelyeznék, ellenőrizni tudod. Győződj meg róla, hogy az ideiglenes könyvtárban még arra is elegendő hely van, hogy több feltöltés történhessen egyszerre, és ne felejtsd a könyvtárat éjszakánként megtisztítani, hogy a szerver ne duguljon el a sok feltöltéstől.

Az upload_max_filesize-zal azt próbálod a fájl méretének korlátozásával befolyásolni, hogy mi tölthető fel. Ezt azonban ki lehet játszani, például a több gigabájtos lopott filmeket sok-sok fájlban töltik fel, majd a megfelelő szoftverrel ezeket összerakják. A fájl méretének korlátozása önmagában kevés arra, hogy megakadályozd a szervered rossz célra való használatát.

Gyors megoldások

Űrlap létrehozása

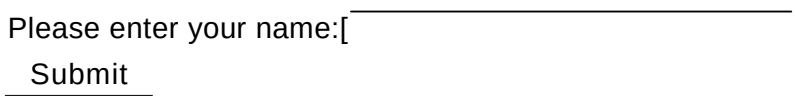
íme HTML-ben egy alapűrlap, amely megkérdi a neved, majd visszatér a createaform.html nevű oldalra:

```
<form action="createaform.html" method="post">
Please enter your name:
<input type="text" name="name">
<input type="submit" name="submit" value="Submit">
</form>
```

Itt van egy alap PHP-kód űrlap létrehozására:

```
print("<form action=\"createaform.html\" method=\"post\">" .
    "Please enter your name:" . "<input type=\"text\"
    name=\"name\">"
    . "<input type=\"submit\" name=\"submit\" value=\"Submit\">" .
    "</form>");
```

A 9.5 ábra böngészőben mutatja az eredményt.



9.5 ábra Egy alapűrlap egyetlen kérdéssel és egy elküld-gombbal

A createaform.html-oldalon a felhasználó által megadott adatok a **Sname** nevű változóba kerülnek. A PHP az űrlapok összes mezejét megragadja **POST-on** vagy **GET-en keresztül,** és az értékeket a HTML <input>-tag name-paraméterével elnevezett mezőkbe helyezi, íme egy kód a createaform.html-oldalon levő adatok felhasználására:

```
if (isset($name) ) [print("Your name is: " . $name);}
```

Űrlapok létrehozása függvényekkel

Az „Űrlapok létrehozása" című részben mutatott kódot nagy űrlapok esetén nehéz megváltoztatni. A megfelelő megoldás az űrlap kis számú függvényre és különálló kérdésekre való felbontása. íme egy alapvető PHP-kód az űrlaptag-ek létrehozására, amely a kérdéseket a **form()** nevű függvénnyel bevitt szöveges sztringként tartalmazza:

```
function form($page, Stext)
    {
    return("<form action=\"" . $page . "\"
        method=\"post\">" . $text
```

```
. "<input type=\"submit\" name=\"submit\"" . "
value=\"Submit\">" . "</form>");
```

A form()-függvény első paramétere a céloldal neve, utána jön az űrlapba kerülő összes kérdést tartalmazó sztring. Az űrlapmód alapértelmezésben post, mert ez az űrlapok létrehozásának legjobb módja. Minden űrlaphoz kell elküld gomb és <input>-tag az elküldés típusával, így beraktam egyet a form()-ba. Egy töröl gombot szintén be lehet tenni.

A következő a szöveges inputot használó űrlap alapvető kódja. A függvénynek szüksége van a kérdés nevére, a kérdés szövegére és egy nem kötelező alapértelmezett válaszra:

```
function form_text($name, $question, $default="")

    if(strlen($default))

        $default = " value=\"" . $default . "\"";

    return($question . "  "
        . "<input type=\"text\" name=\"" $name "V"
        . $default . ">");
```

Az <input> tag-ben lehet egy alapértelmezett érték, hogy az embert az űrlap kitöltésére ösztönözze. A form_text()-ben az if(strlen($default))-sor az alapértelmezett mező szövegét ellenőrzi, mielőtt a value-paraméter szövegét az alapértelmezett szöveg köré helyezné.

A form_text()-ben a return() először a kérdést, néhány szóközt, majd az <input> tag-et a megfelelő paraméterekkel adja vissza. A $question azért szükséges, mert csak így tudod a vásárlóval közölni, hogy mit is szeretnél megtudni. A $name szintén kötelező, mert különben nem tudod a szkriptedbe visszaérkező választ azonosítani.

Most már egy könnyen bővíthető kóddal hozhatsz létre űrlapot, amely soronként egy kérdéssel bővül. A következő kód egy kérdést helyez a form_text()-függvénybe az <input>-tag létrehozására, majd a form_text()-függvényt a form()-függvénnyel veszi körül, hogy a kérdésből űrlapot csináljon. A böngésző részére pedig az egészből outputot hoz létre a print()-tel:

```
print(form ("form.html",
    form text("name", "Please enter your name:")
```

A kód a 9.6 ábrán látható űrlapot adja eredményül.

Please enter vour name: 1p3ubmíT]

9.6 ábra Egykérdéses űrlap

Hosszú lista létrehozása űrlapon belül

Az „Űrlapok létrehozása" című részben mutatott kód nehezen használható a kérdések hosszú listája esetén, így tömb használatával fogjuk a példát kibővíteni. Először hozd létre a tömböt, amely a kérdéseket olyan sorrendben tartalmazza, ahogy azt az űrlapon szeretnéd:

```
$question["name"]    =    arrayC'type"    =>    "text",
    "question"    =>    "Please    enter    your
name:"); $question [ "address" ]    =    arrayC'type"
=>    "text",
    "question"    =>    "Address:");
$question["city"]    =    arrayC'type"    =>
"text",
    "question"    =>    "City:"); $question["state"]
=    arrayC'type"    =>    "text",
    "question"    =>    "State:"); $question [ "country"
]    =    arrayC'type"    =>    "text",
    "question"    =>    "Country:");
```

A tömb öt elemet tartalmaz, és minden elem egy a name-, type- és question-elemeket tartalmazó tömb, ahol a name a tömb kulcsa. A name-mező azonosítja a questíon-re adott választ tartalmazó mezőt, így összhangban kell, hogy legyen a válaszokat feldolgozó szkript mező'elnevezési szabályaival. A type az <input>-tag form_text() nevű formázó függvényének használatára utasítja a form()-ot. A question az <input>-tag elé kerülő olvasható szöveget tartalmazza, ami nem más, mint amit a felhasználó lát. A question tartalmazhat HTML-beli kiemeléseket, hiszen outputként változatlanul jelenik meg.

A következő alapvető kód a szöveg input használatával épít kérdéseket. Ez az „Űrlapok létrehozása" című részben mutatott függvény, néhány változtatással. Egy nevet használ a kérdésre adott válasz a kérdés szövegének és a nem kötelező alapértelmezett válasz azonosítására:

```
function    form^text($name,    $parameters)

    if (!isset ($parameters["default"]))
        {
        $parameters["default"] = "";
        } if
    (strlen($parameters["default"]))
        {
        $parainеters["default"]    =    "    value=\""
                                                .    $parameters["$default"]


    return($parameters["question"]    .    "Snbsp;snbsp;"
        .    "<input    type=\"text\"    name=\""    .    $name
        .    " .    Sparameters    ["$default"]    .    ">");
```

Ez a függvény a Sname-et diszkrét paraméterként, a többi paramétert pedig tömbben fogadja el. Ha a $parameters["default"] nem létezik, akkor létrehozza azt. A kérdésre adott alapértelmezett válasz meglétére ellenőrzi a $parameters["default"] hosszát, és ha nincsen alapértelmezett szöveg, átugorja a value= létrehozását. A return()-utasítás változatlan for-

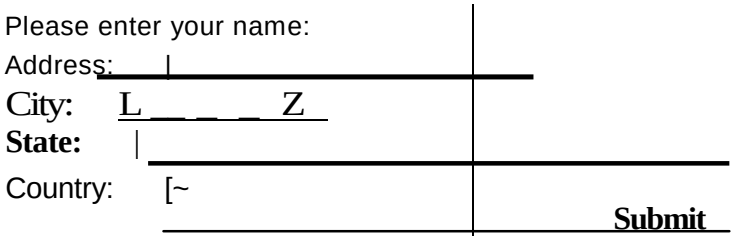
```
                $question) )

        if (strlen($text)) {$text .= "
        $function = "form " .
        $v["type"];
        $text .= $function($name,
        $v); }
    return("<form action=\"" . $page . "\"
        method=\"post\">" . $text
        . "<input type=\"submit\"
        name=\"submit\"" . " value=\"Submit\">" .
        "</form>");
```


A form()-ban a while-ciklus végigfut a kérdések listáján (a $question tömb
mázott kérdést a Stext-be helyezi. Mivel a tömb lehet üres, a formQ kezdés
sztringként létrehozza a $text-et.

A while()-cikluson belül a kód ellenőrzi, hogy van-e szöveg a $text-en belül
kor egy <br> sortörési tag-et szúr be, hogy minden kérdés külön sorban jel
után a kód a form__ konstansból és a tömb típuseleméből függvénynevet ho
egészet együtt a $text-be rakja.

Ha még nem olvastad a 10. fejezetet, talán csodálkozol a $function() miatt.
tozót talál ott, ahol függvénynévnek kellene megjelennie, a függvénynévben
függvény neveként fogja használni. Ez a tulajdonság lehetővé teszi a függvé
mikus létrehozását.

A return() Elküld gombot <input type = "submit"> szúr be a kérdések vég
űrlaptag-et tesz az egész köré. Az igazítás elég összevisszának tűnik, de a kö
oldás ezzel a problémával birkózik meg. A kód a 9.7 ábrán látható űrlapot h



9.7 ábra Többkérdéses űrlap

Oszlopok igazítása

Ez a megoldás az űrlapok oszlopait igazítja egy további kód *hozzáadásával, és* megadja neked a kezdő lökést az űrlapok megjelenítésének ellenőrzéséhez. A példa a kérdéseket lefele futtatja az oldalon, mert a függőleges görgetés mindig egyszerűbb, mint a vízszintes. A szöveget a Squestion-tömbhöz hozzáadva és a form()-ban egy újabb oszlopot létrehozva egyszerűen beszúrhatsz a jobb oldalra egy - a kérdésekkel kapcsolatos megjegyzéseket tartalmazó - további oszlopot.

Kezded a kérdéseket tartalmazó tömbbel. Ez a tömb csak két kérdést tartalmaz példaként:

```
$question["name"]    =    arrayC'type"    =>    "text",
    "question"    =>    "Please    enter    your
name:"); $question["address" ]    =    arrayC'type"
=>    "text",
    "question"    =>    "Address:");
```

A **$question-tömb** összes eleme - kulcsként - tartalmazza a kérdésre adott válasz azonosítására a name-et, illetve egy kételemű tömböt. Ebben a tömbben a **type** jelzi a kérdéshez megfelelő formázófüggvényt, a **question** pedig a kérdést tartalmazza (a szövegformázáshoz szükséges **HTML**-formázó tag-ekkel együtt).

A következő kód ugyanaz, mint a „Hosszú lista létrehozása űrlapon belül" részben bemutatott megoldás **(form_text()),** csak a kérdés szövegét eltávolítottam, így azt a **formQ**-függvényen belül lehet formázni:

```
function    form_text (Sname,    $parameters)

if(!isset($parameters["default"]))
      {
      $parameters["default"]    =    "";
      }
   if(strlen(Sparameters["default"])) {
      $parameters["default"]    =    "
      value=\""
                                          $parameters["default"]

   return("<input type=\"text\" name=\""    $name    "\""
      . Sparameters["default"] . " >");
```

A form_text()-függvény a $parameters["default"]-ben levő szöveget ellenőrzi, mielőtt a value= paraméter a szövegbe helyezi. Ha a $parameters["default"] üres, akkor nincs szükség a paraméter létrehozására.

A return() az összes szöveget az <input> tag-be rakja.

Másold le a form()-függvényt a „Hosszú lista létrehozása űrlapon belül" részből és helyezd a $v["question"]-t a kérdést formázó függvény elé. A form() a kérdések listáját a Squestion-tömbbó'l kapja, a while()-ciklussal végigfut a tömbön, és az egyes kérdések tartalmát a "form_" . $v["type"] nevű függvénynek továbbítja, ahol a type az egyes kérdéstömbök type-eleme:

```
function form($page,
    $question) {
    Stext = ""; while (üst ($name, $v) =
    each ($question) )


        if(strlen($text)) (ítext .= "<br>";}
        $function = "form " . $v["type"];
        $text .= $v["question"] .
        " Snbsp;" . $function($name, $v);
        }
    return("<form action=\"" . $page . "\"
        method=\"post\">" . $text
        . "<input type=\"submit\"
        name=\"submit\"" . "
        value=\"Submit\"X/form>") ;
```

A form() részleteit a „Hosszú lista létrehozása űrlapon belül" részben magyaráztam el, röviden kitérve a változók függvénynévként való szereplésére. A változók függvénynévként való használatát teljes részletességgel a 10. fejezetben mutatom meg.

A form()-nak ebben a változatában a kérdés szövegét egyszerűen kivettem a formázófüggvényből és a form()-on belül formáztam azt, így az összes kérdés szövege ugyanazt a formázást kapja.

Most már bármilyen formázási technikát hozzáadhatsz, így bekezdéseket, sortöréseket és táblázatokat is. Egy egyszerű háromoszlopos táblázatot mutatok be, ahol a középső oszlop egy kis távolságot szúr a kérdések és a válaszok oszlopai közé:

```
function form($page,
    $question) {
    $text = ""; while (üst ($name, $v) =
    each ($question) )


        if (strlen(Stext) ) {Stext .= "<br>";}
        $function = "form " . $v["type"]; $text
        .= "<trxtd>" . $v ["question"] . </td>
        "<td>  </td>"
                . $function($name, $v) .
```

```
return("<form action=\"" . $page . "\" method=\"post\">" .
    "<table>" . $text . "</table>"
    . "<input type=\"submit\" name=\"submit\"" .
    " value=\"Submit\">" . "</form>");
```

A **form()** ezen változatában a táblázat sor <tr> és a cella <td> tag-jei veszik körül a whileQ-ciklusban generált szövegelemeket. így minden kérdés új sorban jelenik meg. A táblázat első oszlopában vannak a kérdések, a második oszlop egy kis területet szúr be, az <input>-tag pedig, ahova a látogató gépelheti a szöveget, a harmadik oszlopban van.

A return()-utasítás a táblázat kezdő- és végtagjait a kérdések köré helyezi, így az összes sor és cella az űrlapon belül jön létre. Ezzel a szimmetrikus elrendezéssel megelőzhetők az egyes böngészőknél olyankor jelentkező problémák, amikor az űrlaptag-ek a táblázaton kívül kezdődnek és belül végződnek (vagy fordítva).

Az Elküld gomb a táblázaton kívül található, így azt a táblázat nem fogja az oldalon igazítani. Nézd meg a böngésző által megjelenített képet, utána egy apró változtatást javaslok, ami érdekelhet téged. A böngésző képe a 9.8 ábrán látható.

Please **enter** yoiir name:

Address:

Submit

9.8 ábra Űrlap oszlopba igazított kérdésekkel

Nem szeretem, ha az Elküld gomb ott van, ahol most. Úgy gondolom, a gombnak közvetlenül az alatt kell elhelyezkednie, ahol a felhasználó begépeli az adatokat, ezért végrehajtok egy apróbb változtatást kézzel. Az Elküld gomb igazításához helyettesítsd a **<table>-, $text-, </table>-** és **Submit** tag-eket tartalmazó sort a következő kóddal:

```
 "<table>"    .    $text
"<tdxinput
type=\"submit\"   "              name=\"submit\""
value=\"Submit\"
```

Ez a kód az Elküld gombot a táblázat egy új sorának harmadik cellájába helyezi, így a gomb közvetlenül az utolsó <input>-tag alá kerül. Amikor a szemeidet és az egeredet lefuttatod a képernyőn, mind a három (feltéve, hogy két szemed és egy egered van) egyből az Elküld gombra mered. Most a böngésző a 9.9 ábrán látható űrlapot jeleníti meg.

Please enter your name:

Address:

Submit

9.9 ábra Mezőket és elküld-gombot tartalmazó űrlap formás oszlopban

Egy válasz a sok közül

Ha éppen a saját utazási irodádat nyitnád meg, és az ügyfeleidtől online szeretnéd megkérdezni, hogy „Melyik országba akar utazni?", egy olyan űrlapra lenne szükséged, amelyen *íz* emberek egy hosszú kiválasztási listán jelölhetnék be a kívánt célországokat. Ez a megokk-egy az országnevek listáját tartalmazó tömböt, a HTML-űrlap a választható országokat megjelenítő **<select>** tag-jét, és egy, a választást tároló változót használ.

Először az országok listáját tartalmazó tömbre van szükség:

```
$country[]    =    "Australia";
$country[]    =    "Austria";
$country[]    =    "Azerbaijan";
```

Kell egy függvény, amely a kiválasztási listákkal együtt megjeleníti a kérdéseket, és a következőkben bemutatott form_select()-függvény pontosan ezt teszi. A függvény a tömb nevét, a kiválasztási listában megjelenő lehetőségek listáját és egy opcionális alapérteimezer mezőt kér, amely utóbbival beállíthatod, hogy melyik lehetőség legyen alapértelmezésben előre kiválasztva. Nézd végig a következőket:

```
function    form_select($name,    Sparameters)

    $output    =    "";
    while (list ($k,    $v)    =    each ($parameters ["üst"] ) )

        if (isset($parameters["default"])    and    $v    ==
        $parameters["default"]\*

            $output    .=    "<option    selected>"    .    $v    .
            "</option>"; }
        else    {    $output    .=    "<option>"    .    $v    .
            "</option>";


    return ("<select    name=\""    .
        $name .    $output    .
        "</select>");
```

A **form_select()** annyiban hasonló a jelen fejezet egyéb Gyors megoldásaiban használt kérdésformázó függvényekhez, hogy egy, a válaszra vonatkozó névre és egy alapértelmezett értékre van szüksége, mely utóbbi a legvalószínűbb válasszal ösztönzi a felhasználót a válaszadásra. A többi rész új, mert egy **<select>** tag-et hoz létre.

A <select> tag-gel kiválasztási listát nyújthatsz a látogatódnak, aki gépelés nélkül jelölheti ki a neki megfelelő válasz. Bizonyos böngészőkben és operációs rendszerekben több lehetőség is kiválasztható, de csak akkor, ha a <select> tag-ben a multiple-paraméter hozzáadásával ezt megengeded.

A form_select()-ben a lehetséges válaszok listáját a **$list** tartalmazza. A while()-ciklus végifut a **$list-en,** egyedi **<option>** tag-eket hozva létre, amelyek mindegyike egy választ

tartalmaz. Az alapértelmezett választ a megfelelő <option> tag-be írt selected kulcsszóval lehet beállítani. Az <option> tag-ek a $output-sztringbe kerülnek, ezért a ciklus első lefutása előtt létrehozom a $output-ot, hátha egyetlen elem sincsen a $list-ben.

A return()-függvény adja hozzá a neveket a <select> tag-hez, és a $output-ban a <select> tag-gel körülveszi az <option> tag-ek listáját.

Hogyan jeleníted meg az új kérdésed? Változtasd meg az előző példában használt űrlap kérdéstömbjét, így az országokat tartalmazó tömb lesz a Üst-paraméter a select típusú kérdésekhez. Ezután az előző megoldásban használt form()-függvényen elvégzett kis módosítással könnyedén kezelhetők az ilyen típusú kérdések:

```
$question["destination"]    =    array("type"    =>    "select",
    "question"    =>    "Which    country    do    you    want    to
    visit?", "üst"    =>    $country) ;
```

A $question minden egyes eleme egy az elem nevét, típusát és a kérdést tartalmazó tömb. Ez az elem, a select típusú, az elemlistát is tartalmazza. Az elemlista egy olyan tömböt fogad be, amely a select típusú kérdések által generált <select>-tagben megjelenítendő válaszok listáját tartalmazza.

A $question kulcsértéke elnevezi a választ az azt fogadó szkriptben, a type meghatározza a kérdést formázó függvényt, a questionelem pedig a kérdés szövegét tartalmazza.

Egy default nevű opcionális elemet is beszúrhatsz, amely a kérdésre adott alapértelmezett választ állítja be. A <sdect>-listákban az alapértelmezett választnak egyezni kell a listatömb egy elemével, beleértve a kisbetűs-nagybetűs írásmódot is. Az alapértelmezett elem hozzáadásának biztonságos módját a következőkben mutatom meg. Amikor az "Anguilla" elemet hozzáadod a Scountry-listához, először a $country_default-hoz add, és utána a $country_default-ot add a $country-hoz, mint itt:

```
$country_default    =    "Anguilla";
$country[]    =    $country_default;
```

Amikor az elemet hozzáadod a $question-höz, a default-ot közvetlenül a $country_default-ból add meg. így a default hozzáadódik. A következő példa az előző kérdést tartalmazza, csak a default hozzáadásával:

```
$question["destination"]    =    arrayC'type"    =>    "select",
    "default"    =>    $country_default,
    "question"    =>    "Which    country    do    you    want    to
    visit?", "üst"    =>    Scountry) ;
```

Ezután az előző megoldásban kialakított form()-függvényt változatlanul hagyva használd azt az új kérdéstípushoz. A form()-ban a kérdéslista egy tömb, melynek minden elemét az adott elem type-értékének megfelelő függvény dolgoz fel. Az előző példák a name-et és default-ot továbbították a függvénynek, a form_select()-nek pedig te akarsz egy további paramétert továbbítani. A form() továbbra is működni fog, mert az összes, a name-től különböző paraméter egy egyszerű tömbben továbbítódik.

A paraméterlistához alkalmazott tömb használatának egyik lehetséges alternatívája, ha egy rögzítetten három paraméteres listát használsz, és a jelenlegi form()-függvényt úgy változtatod meg, hogy az elfogadjon dummy-változót, így minden függvény pontosan három paramétert fogad el.

Hivatkozás:	**oldal**
Alapértelmezett függvényparaméterek használata	347

A következő kód egy külön sorral bővíti ki a táblázatot, az Elküld gombot pedig a külön sorba helyezi. így az Elküld gomb a válaszbeviteli területtel lesz sorba állítva. Futtasd le a kódot az oldaladon, a böngésződnek a 9.10 ábrán látható űrlapot kell megjelenítenie:

```
function    form($page,
    $question) < $text    =    "";
    while (üst <$name,     $v)     =    each ( $question) )
        {
        if ( !isset($v["default"])) {$v["default"] = "";}
        if(strlen($text)) {$text .= "<br>";} $function =
        "form_" . $v["type"]; $text .= "<trxtd>" . $v [
        "question" ] . "</td>"
            . "<td>Snbsp; </td><td>";
        if (isset($v["list"]) )
            {
            $text .= $function($name, $v["list"]r $v["default"]);
            }
        else
            {
            $text .= 5function($name, $v["default"]);
            }
        $text .= n</td>"; }
    return("<form    action=\""    .    Spage    .    "\"
        method=\"post\">" .    "<table>"    .    $text    .
        "<tr><tdx/td><td></td>" .    "<tdxinput    type=\"submit\"
        name=\"submit\"" .    "
        value=\"Submit\"x/td></tr></table></form>") ;
```

Which country do you want to visit? | Anguilla

Submit

9.10 ábra Űrlap legördülő listával

Ezután a kódnak meg kell jelenítenie a választást. A következő kód egyszerűen ellenőrzi, hogy a **$destination** nevű változó létezik-e (mivel a **destination** az, amit a **Squestion**-tömbben a kérdéshez beírtál), majd megjeleníti a változót:

```
if(!isset($destination))
```

```
print("<br>You selected " . $destination); }
```

Amikor egy űrlap egy URL-en vagy a GET- vagy POST-mechanizmusokon keresztül elküldi az értéket a céloldalnak, a céloldal PHP-szkriptje a változót az értéket létrehozó HTML-tagben használt name-paraméterben adott névvel látja. Ha az <**input name="pet"**> és valaki a beviteli mezőbe a "dog" szót írja, akkor a szkript a $pet-et a dog értéket tartalmazva látja.

Egy válasz a sok közül rádiógombokkal

Ha a kocsidban utazol, az előre beállított rádiógombokat megnyomva azonnal választhatsz a rádióállomások között. Az egyik ingázó embertársad ezt a lehetőséget megcsinálta HTML-ben is a **radio** típusú <input>-tag hozzáadásával. A rádiógombok akkor célszerűek a <select>-tagben, amikor rövid listákat jelenítesz meg, mert ekkor az összes alternatívát egyszerre láthatod. Ezzel szemben a kiválasztási lista a hosszabb listák esetén célravezető, mert az összes lehetséges válasz rádiógombokkal való megjelenítése nagyon sok oldalnyi helyet foglalna el.

Ha az előző, utazási irodás példában szeretnél rádiógombokat használni, egy az országok listáját tartalmazó tömbbel kezdenéd:

```
$country[]  =  "Australia";
$country[]  =  "Austria";
$country[]  =  "Azerbaijan";
```

Szükséged van egy függvényre, amely rádiógombokkal jeleníti meg a kérdéseket, és a következőkben bemutatott form_radio()-függvény pont ezt teszi. A tömb nevét, a megjelenítendő lehetőségek listáját és egy opcionális default-mezőt kell megadni neki, ez utóbbiban az alapértelmezettként kiválasztott lehetőséget tudod beállítani. íme a form_radio():

```
function  form_radio($name,
  $parameters) {
  $output  =  "";
  while (Üst <$k, $v) = each ( $parameters [ "list" ] ) ) {
     if(strlen($output)) {
        $output .= "

     $output .= "<input name=\"" . $name . "\"" . " type=\"radio\""
        . " value=V" . $v . "\'-"; if (isset($parameters["default"]) and $v
     == $parameters["default"])
        {
        $output .= " checked";
        } $output .= ">" . $v .
     "\n";
     }
  return($output);
```

A form_radio() annyiban hasonlít a form_select()-hez, hogy egy a válaszmezőhöz tartozó nevet és egy alapértelmezett értéket fogad el, amely utóbbi a legvalószínűbb válasz előre történő megjelölésével válaszadásra sarkallja a felhasználót. A többi része új, hiszen egy <input>-taget hoz létre. A Gyors megoldások „Több válasz a sok közül" részében hasonló kódot használok egy < input type = "checkbox">-tag előállítására.

Az új kérdésed megjelenítéséhez másold le az előző rész kérdéstömbjét, ahogy itt is látod, és tápláld be az előző megoldás formQ-függvényébe:

```
$question["destination" ]    =    arrayC'type"    =>    "radio",
    "question"    =>    "Which    country    do    you    want    to
    visit?", "list"    =>    Scountry);
```

A Squestion kulcsértéke elnevezi a választ az azt fogadó szkriptben, a type meghatározza a kérdést formázó függvényt, a question-elem pedig a kérdés szövegét tartalmazza. Egy opcionális, default nevű elemet is hozzáadhatsz, amely egy alapértelmezett választ ad a kérdésre.

Több válasz a sok közül

Az *előző* Gyors megoldások példában, amikor utazási irodát nyitottál, és arról kérdezted az ügyfeleket, milyen országokba akarnak utazni, megállapítottad, hogy jó lenne egy űrlap egy hosszú kiválasztási listával, amelyen az emberek több országot is bejelölhetnek, ahova szívesen utaznának. Ez a megoldás az egyes választások elfogadására HTML-űrlap bejelölő-dobozokat, egy az országok listáját tartalmazó tömböt, illetve egy a választások tárolására használt tömböt alkalmaz. A kód az „Egy válasz a sok közül" részben használt kód némileg módosított változata, amelyben a <select>-et <input type = \"checkbox\">-ra cseréltem. Ezt a kódot használva az egyetlen választást tartalmazó változó helyett a választások listáját tartalmazó tömböt kapsz eredményül, amikor a felhasználó visszajuttatja hozzád az általa kitöltött űrlapot.

Megjegyzés: Talán már kitaláltad, hogy kedvelem a PHP-tömböket, mivel űrlapok használatakor értékek listáját képesek továbbítani. A többdimenziós tömbök nem működnek: csak egy lehetőséged van, és ez nem más, mint a hagyományos name="fieldname>" HTML-tagnek a PHP-barát name="fieldnameO"-re cserélése (habár ez a megkötés a későbbi böngészőkkel és a PHP későbbi verzióival változhat).

Kezdj egy az országok listáját tartalmazó tömbbel:

```
$country[]    =    "Australia";
$country[]    =    "Austria";
$country[]    =    "Azerbaijan";
```

Szükséged van egy függvényre, amely minden egyes tömbelem köré helyezett <input type = "checkbox">-szál megjeleníti a tömbből a lehetséges válaszokat. A következőkben bemutatott form_checkbox()-függvény a tömb nevét, a kiválasztási listában megjelenő lehetőségek listáját és egy opcionális alapértelmezett mezőt fogad el. Az alapértelmezést az összes lehetséges válaszra beállíthatja, így olyan listát is létre tudsz hozni, amelyből minden ki van választva, és a vásárlónak azt kell megjelölnie, amire nincsen szüksége.

Figyeld meg, hogy a <input>-tagban a name=""-paraméter []-t tartalmaz a mezőnév után, ezzel jelezve a PHP-nak, hogy az értéket a tömb következő elemébe rakja bele. A [] a name-paramétert a PHP-kóddal teszi ekvivalenssé, mint a példában:

```
$destination [ ]  =  "_the  name  of  the  country__";
```

Az <input>-tag value=""-paramétert is tartalmaz. HTML-ben a value-paraméter kihagyható, a böngésző pedig az <input>-tag és az </input>-tag közötti sztringet fogja használni. A PHP-ben és néhány böngészőben a value-paraméter elhagyása azt fogja eredményezni, hogy a tömbelem az on értéket tartalmazza majd. Kerüld el ezeket a problémákat azáltal, hogy használod a value-paramétert.

Bejelölődobozok esetén a kiválasztott elemeknek selected helyett checked-nek kell, hogy lenniük, ne felejtsd hát a kódnak ezt a sorát megváltoztatni. A fejezet korábbi példáival szemben ez a függvény lineáris módszert használ a kimeneti sztring felépítésére, ami jobban használható a sok lehetséges opcióból épített nagy sztringek esetében. A tagösszetevőket részről részre adom a kódhoz, így a jövőben is könnyen szúrhatok be kódrészeket anélkül, hogy a már meglévő kódot, illetve annak logikáját megzavarnám. A sztring végére egy \n-t is beszúrtam, így könnyebb lesz a HTML-t a böngésző „forrás megtekintése" pontjában olvasni. Amikor egyéb sztringépítő függvényeket írsz PHP-ban, talán össze szeretnéd hasonlítani ezt a megközelítést a form_select()-évei. Nézd meg ezt itt:

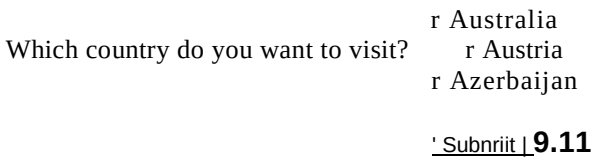
```
function  form_checkbox($name,  $parameters)
    {
    $output  =  "";
    while (list ($k, $v) = each ( $parameters [ "üst" ] ) ) {
        if(strlen($output)) {
            $output .= "

        $output  .=  "<input  name=\""  .  $name  .  M[ ]
            .  "  type=\"<:heckbox\""
            .  "  value=\""  .  $v  .  "\"";
        if(is_array($parameters["default"]) and
        in_array($v,  $parameters["default"] ) )
            {
            $output  .=  "  checked";
            } $output  .=  ">"  .  $v  .
        "</input>\n";
        }
    return{$output);
```

A form_checkbox() a kérdés nevét, a válaszok listáját, illetve az alapértelmezett válaszok opcionális listáját fogadja el.

A while()-ciklus a $output-változóban állítja össze a listát, így az első dolgod a $output létrehozása legyen. A while()-ciklus végigfut a listán, és elemenként egy <input>-tagét hoz létre. Annak érdekében, hogy az elemek egy függőleges listában elkülönüljenek egymástól, egy sortörést <br> szúr be minden egyes <input>-tag elé. Az első elem előtt

nincsen üres sor, mert csak akkor csinál sortörést, ha van szöveg a $output-ban: erre van az if(strlen($output)).

Az <input>-tag egy nevet kap a válasz céloldalon való azonosítására, illetve a checkbox ti-pusát. A checkbox a 9.11 ábrán látható opciólistát hozza létre. Minden bejelölődoboz egy külön elem, így a látogató annyi dobozt jelöl meg, amennyit akar.



' Subnriit | **9.11**

ábra Űrlap többszörös kijelölődobozzal

Hogyan jelenítsd meg az új kérdésed? Megváltoztatod az ezekben az űrlappéldákban használt kérdéstömböt, így az országokat tartalmazó tömb Üst-paraméter lesz egy checkbox típusú kérdéshez, mint itt:

```
$question["destination"] = arrayC'type" =>
    "checkbox", "question" => "Which country do you
    want to visit?", "üst" => $country) ;
```

Ha ez megvan, az előző példában használt form()-függvényt változtatás nélkül használva megjeleníted az országkiválasztási listát. Ehhez a következőt kell beírnod:

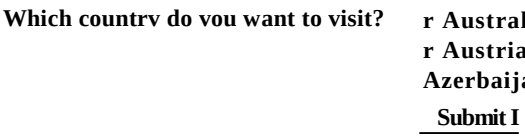
```
function form($page, $question) {
    .
    Stext = ""; while (üst ($name, $v) = each
    ($question) )
        {
        if(strlen($text)) {$text .= "<br>";}
        $function = "form_" . $v["type"];
        $text .= "<trxtd>" . $v [ "question" 3 . "</td>"
            . "<td>  </td>"
            . "<td>" . $ function ($name, $v) . "</tdx/tr>";


    return("<form action=\"" . $page . "\" method=\"post\">"
        . "<table>" . Stext . "<tr><tdx/tdxtdx/td>"

        #
        . "<tdxinput type=\"submit\" name=\"submit\""
        . " value=\"Submit\"x/tdx/trx/table>"
        . "</form>");
    }
```

A form() a válaszokat megkapó oldal nevét és a kérdések listáját kéri. A while()-ciklus a megfelelő HTML-t felépítve végigfut a kérdéseken. A return() az Elküld gombot szúrja be, majd a HTML-t az űrlap- és táblázattagek közé helyezi.

Jóllehet, a legtöbb ember már bolondulni fog az űrlapodért úgy is, ahogy az a 9.11 ábrán megjelenik, néhányan hiányolni fogják a felülre igazítást, amit a 9.12 ábrán láthatsz. Ezt a kis változtatást, amely az összes űrlapra érvényes és könnyen módosítható a kód megvál-

toztatása nélkül, a formázási opcióban, $option állíthatod be, mint itt:

```
$option["form"]["valign"] = "top";
```



9.12 ábra Űrlap többszörös kijelölődobozhoz gondosan hozzáigazított kérdéssel

Ha ezt megteszed, egy külön sort kell beszúrnod az oszlopok közötti távolság kezelésére, amelyhez így könnyen adhatsz és vehetsz el belőle, ha a weboldalad fejlődése ezt megkívánja. A Soption-tömböt egy szokványos include-ból, egy adminisztrációs adatbázisból vagy egy site adminisztrációs oldal által generált XML-fájlból töltheted be. A változtatáshoz a következőt kell beírnod:

```
$option["form"l["separator"]    =    "Snbsp; ";
```

Ezután meg kell változtatnod a form()-ot(hogy az a formázást az űrlap táblázatának összes cellájára alkalmazza, és a nehezen módosítható távolságigazító oszlopot a $option-ből betöltött értékkel helyettesítse. Más nyelvekben a változókat elég a szkript elején egyszer globálisnak nyilvánítani. PHP-ban egy változó addig nem globális, amíg a függvényeden belül egy global utasítással annak nem nevezed. Ez azt jelenti, hogy egyszerűbb egy tömböt egy global utasításban globálissá nyilvánítani és mindent a tömb elemeként eltárolni, mint minden értéket külön változóban tárolni, és az összes változóhoz külön global utasításokat rendelni.

A $option-ből származó értékek használata előtt ellenőrizd az isset() használatával, hogy az érték be van-e állítva. Ezen felül állíts be egy használható alapértelmezést arra az esetre, ha valaki elfelejtett egy értéket. Az elválasztó sztring esetében megváltoztattam a kódot, hogy teljesen kihagyjam a táblázatelválasztó oszlopot, ha az elválasztási érték nincsen definiálva. Végül, de nem utolsósorban a formázott <td>-sztringet, a $td-t használtam az elválasztó oszlopban, hátha valaki úgy dönt, hogy további <td> formázó opciót, például bgcolor-t ad hozzá. Hogy lásd, mindez hogyan működik, nézd meg ezt:

```
function form($page, Squestion)

    global $option;      ;     . ,
    $td = "<td";       "*'■' '•"'■*-'"■-■
    if (isset (Soption["form"] ["valign"]))

        $td .= " valign=\"" . $option["form"]["valign";     "V
        }                                                  ' , ' ,
    $td .= ">"; $sep = "";
    if(isset($option["form"]["separator"]))
```

```
    $sep = $td . $option ["forrn"] [ "separator" ] . "</td>";
    }
Stext = ""; while (üst ( $name, $v) = each
($question) )
    {
    if (strlen($text)) {$text .= "<br>";}
    $function = "form " . $v["type"];
    $text .= "<tr>" . $td . $v["question"] . "</td>" . $sep     $td;
    $text .= $function($name, $v["default"])


return("<form action=\""    .    $page    .    "\"
    method=\"post\">" .     "<table>"     .     Stext     .
    "<trxtdx/td><tdx/td>" .     "<tdxinput    type=\ "submitV
    name=\"submit\"" .     "    value=\"Submit\" .     "</form>");
```

A form() a $option-t globális változókét definiálja, így hozzáférhet a form()-on kívül definiált $option-höz.

A $td=-vel kezdődő kód a táblázat celláinak definícióját állítja be. A <td>-tag a $option-ben lévő beállításokból épül fel, így a $optíon-t használhatod a HTML-ed formázásának teljes ellenőrzésére. A példában csak annyit állítottam be, hogy a <td> valign-paraméter függőlegesen igazítsa a *táblázni* celláiban a szöveget. Ezt kiterjesztheted a színekre és további <td>-opciókra is.

Megjegyzés: Miért így alakítsd ki a formázást, és miért nem a *CSS-t használva? Azért, mert nem minden böngésző képes a CSS feldolgozására. Vannak olyan böngészők, amelyek rossz dolgokat jelenítenek meg CSS-sel. Még a legújabb böngészők sem egységesek a CSS értelmezésének finom részleteiben.*

A Ssep = -vel kezdődő kód egyszerűen egy cellát hoz létre a szöveget tartalmazó oszlopok elkülönítésére. Azért használok elválasztó oszlopokat, mert a képernyő széles, és az embereknek vízszintesen nagyobb térre van szükségük az adatok elválasztására, mint függőlegesen.

A kód Stext="<tr>"-rel kezdődő sora a régi kód továbbfejlesztett változata, amelyben a rögzített <td>-taget $td-re cseréltem, a "<td> </td>"-t pedig $sep-re.

A form()-függvényen végzett további munka, miként azt a kódot lefuttatva megláthatod, jobb és rugalmasabb űrlapot eredményezett. Ezen űrlap eredményeinek feldolgozását a következő kóddal kell kezdeni. Az adat a $destination-nek nevezett változóban van, így először azt kell ellenőrizned, hogy a változó létezik-e, ha létezik, akkor tömb-e, és ha igen, akkor egy ciklussal jelenítsd meg az értékeit. Az űrlap összes kiválasztott eleme a value=""-ból kap értéket, ami tömbben tárolódik értékként. Bármilyen adatfeldolgozást elvégezhetsz, ha az adatok szerencsésen megérkeztek a szkriptedbe. A kód a következő:

```
$d   =   "";
i f ( isset($destination))
    {
    $d    =    $destination;
```

```
if(is_array($destination))
    {
    $d = "";
    while (üst ($k, Sv) = each ($destination) )

        $d .= "<tr><td>" . $v . "
        } $d = "<table>" . $d .
    "</table>";
```

A kód egy alapértelmezett értéket állít be a $d-nek, amely az a változó, ami a HTML-t és a szöveget kapja. Az if(IssetQ) ellenőrzi, hogy a $destination létezik-e (az oldalra való első belépéskor lehet, hogy nem létezik), majd a $d-hez a $destination-t rendeli (hiszen a $destination lehet formázott sztring). Ha a $destination tömb, akkor a $d-ből kitörli az értéket, a whileQ-ciklus végigfut a $destination-ön táblázatsorokat szúrva be, majd a $d-t <table>-tagek közé helyezi.

Válaszok megőrzése és hibák kiemelése

Az *előző* megoldás azt feltételezte, hogy a vásárló kitölti az űrlapot, és csak egyszer kattint az Elküldre. De mi történik, ha egy többoldalas űrlap első oldaláról a másodikra akarod az adatokat továbbítani, vagy az első oldalt hibaüzenettel együtt újra meg akarod jeleníteni? Ebben az esetben csak külön mezőket kell a Squestion-tömbbe és a feldolgozó függvénybe tenned. Ekkor egy újabb oszlopot is beszúrhatsz a hibaüzeneteknek, és piros szöveget szúrhatsz az egyes kérdések eié, így a vízszintes elrendezés sem fog oldalról oldalra ugrálni. Végeredményben ugyanazt a $question-tömböt fogod használni, mint az előző példában, csak további értékeket adsz neki a feldolgozás alatt. A hibaüzenetek egyszerű teszteléséhez változtasd meg a kérdést úgy, hogy legfeljebb két országot lehessen kiválasztani, mint itt:

```
Squestion["destination"] = arrayC'type" =>
    "checkbox", "question" => "Select up to two
    countries:", "list" => $country);
```

így összepárosíthatod a válaszokat a kérdésekkel, minden kérdést névvel indexelve. Majd amikor egy válaszmezőt dolgozol fel, amilyen például a $destination, a kérdésre hivatkozhatsz $question["destination"]-ként, és további elemként könnyedén beszúrhatsz válaszokat és hibaüzeneteket is.

Amikor hiba van egy válaszban, és üzenetet akarsz a felhasználónak visszaküldeni, hogy felhívd a figyelmét a hibára, csak egy sorra van szükség a kódban, hogy az a Squestion-tömbbe szúrja be a hibát. A következő példában megláthatod, hogyan működik ez, ha egy vásárló több, mint két országot választ ki. Természetesen kezelheted magad a saját hibáidat, amint az oldalad sikeresen megjeleníti a hibaüzeneteket. A $question-tömbbe így teheted be a hibákat:

```
if (count($destination) > 2)
    {
    $question["destination"]["error"] =
```

```
        "You selected more than two countries."
         . "<br><em>(Hint: Two is the number after one.)</em>";
    }
```

Arról is meg kell győződnöd, hogy a korábbi válaszokat továbbítottad, nehogy az embereknek újra be kelljen azokat gépelniük. Ehhez nem kell mást tenned, mint a korábbi válaszokat új default-értékként berakni a kérdéstömbbe, mint itt:

```
if (isset($destination)  and  is_array($destination) )

    $question["destination"]["default"]  =  $destination;
```

Mind az alapértelmezett list, mind a **$list** betölthető adatbázisból. Ha van egy adatbázisbeli táblázatod, amely a **$list** és az alapértelmezett lista adatait is tartalmazza, a két listát egy tömbben egyesítheted, és a tömböt SQL-ből feltöltheted. Ez azt jelentené, hogy a **form_checkbox()-ban** a **$parameter["default"]** eltűnne és a $v egy tömb lenne, mely tartalmazná egyrészt a megjelenítendő értékeket, másrészt valamit, ami azt jelzi, hogy az elemnek a **selected** alapértelmezett beállítását kell adni. Végül egy kicsit megváltoztatod a form()-ot, hogy a hibaüzenetet a kérdés elé tegye, mint itt:

```
function form($page, $question)

    global $option;
    $td = "<td";
    if(isset($option["form"]["valign"]))

        $td .= " valign=\"" . $option["form"]["valign"] . "\"";

    $td .= ">";     _^
    $sep = "";
    if(isset($option["form"]["separator"]))

        $sep = $td . $option["form"]["separator"] . "</td>";

    $text = "";
    while(list($name, $v) = each($question))

        if(isset($v["error"]))

            $v["question"] = "<font color=\"Red\">" . $v["error"] .
                "</font><br>" . $v["question"] ;

        $function = "form_" . $v["type"];
        Stext .= "<tr>" . $td . $v["question"] . "</td>" . $sep . $td;
        if (isset($v["list"] ) )

            $text .= $function($name, $v["list"], $v["default"]); 1
        else í $text .= $function($name, $v["default"]);

        Stext .=
```

```
return("<form action=\""  .  $page  .  "\"
    method=\"post\">" .  "<table>"  .  $text  .
    "<tr><tdx/td><tdx/td>" .  "<tdxinput  type=\"submit\"
    name=\"submit\"" .  "
    value=\"Submit\"X/td></tr></table></form>") ;
```

Az első változtatás a form()-on a kód, amely a $v["error"]-t dolgozza fel. Az **error** elem csak akkor létezik, ha egy kérdésre hibásan válaszoltak, így először ellenőrizni kell, hogy az elem létezik-e. Ha az **error** létezik, a kód kiveszi belőle a szöveget, pirosra állítja (a <font>-tagben), és a vörös szöveget beszúrja a kérdés szövege elé.

í "i

10. fejezet

Függvények

Áttekintés

Képzeld el, hogy kocsit tervezel. A motort és a karosszériát együtt kell megtervezned, mert ki kell próbálnod, hogy jól illeszkednek-e. Ha egy tervező csapattal dolgoznál együtt, akkor ez a csapat mind a motor, mind a karosszéria kialakításán egyszerre dolgozna. Most gondolj egy CD-lejátszóra, amely majd ebbe a kocsiba kerül. Mivel egy ilyen eszköz teljesen másfajta tervezési technikát igényel és teljesen másféle tervezési problémák merülhetnek fel, egy külön tervező csapat fog foglalkozni vele. A kocsit tervező csapat összeállítja az autót, egy szabványos méretű lyukat és néhány vezetéket hagyva a műszerfalon, ahova majd a CD-lejátszó bekerül. A CD-lejátszót tervező csapat úgy alakítja ki a lejátszót, hogy az a lyukba és a vezetékekhez pontosan illeszkedjen, amikor elkészül.

Ugyanezt teheted PHP-ban a függvényekkel. Válaszd ki a programod azon részeit, amelyek egyedül is megállnak, és tedd azokat függvényekbe. A függvényfejlesztői munkát kiadhatod másoknak, és a programfejlesztési időt csökkentheted a függvények más programokban való használatával. Amikor Jacques Nassar, az ausztrál szuperhős a Ford Motor Company világszintű műveletekért felelős vezetője lett, a 22 öngyújtó designt egy darab, három méretben gyártott designra cserélte. Jacques Nassar ma a Ford első számú embere. Gondolj bele, mivé válhatsz a kód megfelelő újrahasznosításával.

Ez a fejezet a függvényeket és azok használatát mutatja be, illetve megtanít arra, hogyan alkalmazd őket saját magad. A következő függvényekkel fogok a fejezetben hangsúlyosan foglalkozni:

- **call_user_func()**
- **create_function()**
- **func_get_arg()**
- func_get_args()
- **func_num_arg()**
- **register_shutdown_function()**

Amikor a kód gyorsan változik, egyszerűbbnek találhatod a fejlesztést, ha először egyben hagyod a kódot, majd a kódrészeket függvényekre darabolod. Később fel fogod ismerni, hogy mi való leginkább függvénybe, és a kód megírása előtt létrehozod azokat a függvényeket, amelyeket használni fogsz.

Az *objektumokkal* elkülönülő részekre bonthatod az adatok és kódok összetett elegyét. Ha hasznos függvények és adatok csoportjait alakítottad ki, gondold meg, nem lenne-e érdemes egy objektumba összerakni őket. (Az objektumokról a 17. fejezetben olvashatsz részletesen.)

A függvények nagymértékű rugalmasságot biztosítanak. Annak érdekében, hogy a függvényírás minden apró részletét elsajátítsd, a következő részekben a legegyszerűbb függvényektől haladok a PHP4 összes opcióját kihasználó bonyolultabbak felé.

A világ legrövidebb függvénye

Egy mintakód a legmegfelelőbb kiindulási pont a függvények tárgyalásához. A függvényeket egyszer kell a function utasítással definiálni, utána az egész kódban használhatóak. Az első lépés tehát - ahogy itt is láthatod - a függvény definiálása. A következő a világ legrövidebb PHP-függvénye:

```
function a() {}
```

A példában a function kulcsszóval kezdődik a függvény definiálása. Ezt követi a függvény neve (esetünkben a), a zárójelek (), amelyek az opcionális beviteli mezőknek adnak helyet, majd a kapcsos zárójelek {}, amelyek körülveszik a függvény kódját. Ne felejts el szóközt rakni a function szó és a függvény neve közé, viszont ne tegyél a függvény neve és az első zárójel, illetve a második zárójel és az első kapcsos zárójel közé. Ahol használhatsz szóközt, ott tabulátort, illetve új sorra ugrást is használhatsz. Minden függvénynek ez a felépítése, és ehhez beviteli mezők, feldolgozó kód és az eredmény megjelenítésére használt eszköz megfelelő keveréke adható.

A függvényt a függvénynév és a zárójelek beírásával használhatod. Példánkban ez:

Az a() példafüggvény nem csinál semmit, és nincsen eredménye sem. Ezért hamisat ad eredményül, ha a következőkben if()-fel teszteled:

A függvénynevek nem különböztetik meg a kis- és nagybetűt. így az a()-t a következőképpen is használhatod:

Megjegyzés: A beviteli mező, paraméter és argumentum mind ugyanazt jelentik: azt a mezőt, amellyel információt viszel a függvénybe. A beviteli mező technikailag nem helyes, hiszen a mezők kétirányúak is lehetnek. Vannak iskolák, amelyek a paraméter, és vannak, amelyek az argumentum kifejezést használják.

Ez a függvény ebben az állapotában nem tesz semmit, de a következő szakaszokban tárgyalt függvények már igen.

Értékek visszaadása

Az *előző* példában az a()-függvény nem csinált semmit, így végeredményben semmilyen haszna nincsen. Normális esetben a függvényt arra használod, hogy valamilyen feladatot elvégezzen, és valamilyen értéket adjon eredményül, amit a függvények a return()-utasításon keresztül csinálnak meg. A PHP nem kényszerít a változókra vagy függvényekre adattipizálást, így a függvényedből bármilyen típusú értéket kaphatsz eredményül.

Ha nem ismernéd a *tipizálás* kifejezést, akkor elmagyarázom. Arra a folyamatra használják bizonyos nyelvekben, amikor valami az adatot meghatározott adattípusúvá változtatja. A

PHP az adatokat általában nem tipizálja, amikor változóba helyezed őket, de azért ez is előfordulhat, mint a következő példa mutatja. A $b értéke a $a-ban lebegőpontos számként van eltárolva. Hogy esetleges zavarodottságod fokozzam, elmondom, hogy a *lebegőpontos számokat* hívják *valósnak,* illetve *doubles-nek.* Ez utóbbi a PHP-ban a legelterjedtebb:

```
$a = (double) $b;
```

Előfordulhat, hogy eredményként egy tömböt szeretnél, ha minden jól működik, és egy logikai hamis értéket, ha rossz valami. Ebben a részben arról lesz szó, hogyan ad a függvény eredményül értéket, hogyan jelenítsük meg a kapott értéket annak igazolására, hogy az megfelelő, majd hogyan teszteljük a kapott értéket a egyedi eredmények észlelésére és arra az esetre, ha a függvény hibát jelez. A következő példa olyan függvényt mutat be, amely egy <font> tag-et ad eredményül, amellyel egy hibaüzenet elkezdhető. Számtalan egyszerű formázófüggvényt találsz ebben a könyvben, mert azokkal megbízható HTML-oldalakat lehet létrehozni, amelyek minden böngészőben működnek:

```
function red() {return("<font color=\"red\">");}
```

Ez a függvény mindössze egy a piros színt magában foglaló HTML <font> tag-et tartalmazó sztringet hoz létre, és a sztnnget adja eredményül. Használd a red() példafüggvényt print- vagy echo-utasítással (vedd észre a pontösszefűző karaktert), mint itt:

```
print(red() . "my error message</font>");
```

A print-utasítás egy sort hoz létre a weboldaladon, amely piros betűkkel jeleníti meg a következőt:

```
my error message
```

Ha a print()-utasítás használatával az értéket egyenesen a függvényből küldöd a böngészőbe, akkor nagyobb oldalformázó függvényekben nem használhatod a függvényt az egész honlap felépítéséhez. A következő példa azt mutatja meg, hogy némely programozók hogyan írnak üzenetfüggvényt beépített printQ-utasításokkal. Ekkor nincsen ellenőrzésed az üzenet megjelenése felett:

```
function warning_message() {print("<font
    color=\"red\">Warning!</font>");
```

Ehelyett kérdezd le az értéket a függvényből, majd tedd a függvényt olyan műveletekbe, mint a printQ. A következő példa az előzőnek egy változata, amelyben a printQ kikerült a függvényből, így a figyelmeztető hibaüzenetet sokkal rugalmasabban használhatod. Ha a printQ-függvény az oldal létrehozásának utolsó lépése, bármennyi szövegformázó függvényt egymásba ágyazhatsz:

```
function warning_message()

    return("<font color=\"red\">Warning!</font>");

print(warning_message());
```

A függvényből kapott eredményekkel kiterjesztheted a függvény használhatóságát. A warning_message()-függvényt bármilyen másik függvénybe bele lehet ágyazni, amelyik üzeneteket küld a weboldalra. így van egy központi hely, ahol megváltoztathatod a kiemelést és a figyelmeztetések szövegét. Ha a warning_message()-függvényt több másik függvény is meghívja, és „Figyelmeztetés"-ről „Vigyázz"-ra akarod az üzenetet változtatni, elegendő a változtatást egyszer végrehajtani.

A következő sorban a hibaüzenet további kiemelést kap megjelenítés előtt:

prir.t ("" . warning_message () .

A PHP-függvények kódolásának másik szokványos módszere, ha a függvénnyel valamilyen műveletet végzünk, és a függvény siker esetén igaz, hiba esetén hamis eredményt ad. A fejezet későbbi részében használt fájlfüggvények vagy hamisat vagy fájlazonosítót eredményeznek, mely utóbbi nem más, mint egy nullánál nagyobb egész szám a megnyitott fájlok esetén. Ez a nulla értékű egészeket hamisnak értelmezi, míg minden más egészt, így a negatív értékeket is igaznak. Ez azt jelenti, hogy a fájlazonosító értékével úgy tesztelheted a sikert vagy hibát, mintha az igaz/hamis lenne.

A következő példában láthatod, hogy hogyan használd a fopenQ PHP-függvényt és hogyan teszteld az eredményt. Vedd észre, hogy a fopen() előtti @ - például nem létező fájlok esetén - elrejti a PHP beépített hibaüzenetét, így megfogalmazhatod a saját üzenetedet:

```
$file_handle = @fopen("atestfile.txt",
"r"); if($file_handle)
    {print("<br>File open");} eise
{print("<br>File did not open"};}
```

A PHP tudja if()-tesztben használni az érték-hozzárendelés eredményét. így a következőképpen egyszerűsítheted az előző kódot:

```
if($file_handle = @fopen("atestfile.txt", "r"))
    {print("<br>File open");} else {print
("<br>File did not open");}
```

Figyeld meg, hogy a pontosvessző (;) eltűnik a fopen()-utasítás mögül. Az egyik leggyakoribb kódolási hiba az if()-be helyezett utasítás végén hagyott pontosvessző. PHP4-ben az ilyen hiba azonnal szintaktikai hibát okoz, így az első tesztelésen kijavíthatod.

Ötlet: A beágyazott fájlokban lévő kód mindaddig nincs szintaktikailag ellenőrizve, amíg valahova be nem ágyazzák a fájlt írj egy tesztoldalt, amely beágyazza, és így le is ellenőrzi az összes fájlt.

A PHP a nulla hosszúságú sztringeket hamisnak, minden más sztringet igaznak értelmez. Ha olyan függvényt készítesz, amely sztringet ad eredményül, és szeretnéd hibajelzésre használni, gondold meg, hogy a nulla hosszúságú sztringet használod-e hibajelzőként. Hogy folytassuk a bemutatást, a következőkkel egy olyan függvény eredményét tudod ellenőrizni, amely szöveget hoz létre, és nulla hosszúságú sztringgel jelzi a hibát:

```
if(generate_text())
```

Az a tény, hogy a függvény sztringet ad eredményül, problémát okoz. Mivel a PHP nem erőlteti a változók tipizálását (a változók típusáról a 2. fe]ezetben esett szó), a nullát tartalmazó sztrmget zéró egésznek értelmezi, azt viszont hamisnak. Ha sztringet eredményező függvényed érvényes sztringként előállíthatja az egyszerű zérót, akkor a következő tesztet kell használnod az üres sztring, a "" és "0" megkülönböztetésére:

```
if(generate_text() === "")
```

Az == (egyenlő) összehasonlító operátor összehasonlítja, hogy a két mezőnek ugyanaz-e az értéke, de nem ellenőrzi, hogy ugyanolyan típusú változók-e, így a PHP automatikus típuskonverziója miatt különböző értékek ugyanolyannak tűnhetnek. Az = = = (azonos) összehasonlító operátor az értékek összehasonlítása előtt ellenőrzi, hogy a mezők ugyanolyan típusúak-e, és nem keveri-e össze a hamisat a 0-val, a ""-vei vagy a "0"-val.

Értékek bevitele

A függvényekbe a zárójelek között meghatározott mezőkön keresztül lehet értékeket bevinni. A következőken a **red()**-példát fejlesztjük tovább szöveg bevitelével és a szöveg **<font>** tag-ek közé helyezésével. így a **red()**-függvényt bármilyen szöveggel használhatod, és biztos lehetsz benne, hogy a szöveg a megfelelő **<font>** kezdő- és végtag között van:

```
function red($text) {return("<font color=\"red\">$text</font>");}
```

A következő kódmintában használt red()-függvényt most könnyebb használni, ennek eredménye a függvények gyakoribb használata és a hibák csökkenése.

```
print(red("my error message") ) ;
```

Az eredmény változatlanul egy piros betűkkel írt sor a böngésződben:

```
my error message
```

A PHP alapértelmezésben értékként továbbítja a paramétereket (a függvényen belül lemásolja a beviteli mezőket). A beviteli mezőkön végrehajtott változások nem jelennek meg a függvényen kívül, mert a függvény az értékek másolatán végzi a változtatásokat. A másolatot jellemezhetnénk lokális változónak vagy lokális hatáskörűnek. A hatáskörről e fejezetben később még részletesen szó lesz.

Ha az eredeti beviteli mezőt akarod megváltoztatni, tegyél &-jelet a beviteli mező elé. így a **PHP** hivatkozásként továbbítja az értéket, és a függvény a másolat helyett az eredeti változóval dolgozik. Ennek szemléltetésére próbáld ki a következő kódot:

```
function numbered_message(&$number, $text)
    {
    $number+-t-;
    return < "<brxfont color=\ " red\ " ><br>"
        . $number . " " . $text . "</font>");
    } $message_number =
0;
```

```
print("<br>The message number is " . $message_number) ;
print(numbered_message($message_number, "bright red error message"))
; print(numbered_message($message_number, "another error message"));
print("<br>The message number is now " . $message_number)■
```

A következő eredményt kell látnod. Az üzenetszám 0-val kezdődik, és a numbered_message()-függvénnyel 2-re nő. Ha a számot értékként továbbítottad volna, a numbered_message() a változó másolatát változtatta volna meg, nem magát a változót. Ekkor minden üzenet l-es számú lenne, és az utolsó sor a 0 értéket jelenítené meg:

```
The message number is 0
1 bright red error message
2 another error message
The message number is now
2
```

Tetszőleges értékek

Ha a függvényedet állandóan ugyanarra a célra használod, érdemes egy olyan alapértelmezett beviteli értéket meghatározni, amelyet bizonyos esetekben persze figyelmen kívül lehet hagyni. Adhatsz alapértelmezett értékeket a beviteli mezőknek, és a függvényt használó ember eldöntheti, hogy más értéket ad meg vagy nem. A date()-függvénynek a mai dátum és idő az alapértelmezett értéke, ezáltal az input-mező kitöltése tetszőleges. Ha a date()-et használod, választhatod az alapértelmezett mai dátumot, vagy saját adataidat is megadhatod a függvénynek. Legtöbbször az alapértelmezett beállítást fogod használni, és az éppen aktuális dátumot adod az adatbázis rekordjába vagy a weboldalba, de megeshet, hogy más dátumokat, például születésnapokat vagy a következő hónap első napját adod meg.

Például a red()-függvénynek lehet alapértelmezett figyelmeztető üzenete, de megengedheted, hogy a felhasználó határozza meg a használni kívánt színt.

Ötlet: Miért használnál a piros helyett más színt egy függvényben, amely piros szöveget kell, hogy előállítson? A hagyományos HTML-ben van egy RGB (piros/zöld/kék) szín, a UffOOOO, amely a 100 százalék pirosat, nulla zöldet és kéket jelöl. Ezt a színt bizonyos színvakság esetén nem látják az emberek. Ha ezt #ff3333 színre cseréled, amely némi zöldet és kéket is tartalmaz, akkor mindenki el tudja az üzenetet olvasni. Ez is minden böngészőben biztonságosan megjeleníthető, és még mindig piros. A részletekért látogasd meg a http://petermoulding.com/colour/colourblindness.html oldalamat.

A következő kód az alapértelmezett figyelmeztető üzenettel mutatja a red()-függvényt:

```
function red($text="Warning, unidentified error!")
```

Ha ezt írod a szkriptedbe:

```
print(red());
```

ez lesz az eredmény:

```
Warning, unidentified error!
```

A kötelező és opcionális mezőket kombinálhatod is. Helyezd az opcionális mezőket a végére, így az emberek elhagyhatják azokat a mezőket, amelyeket nem használnak, vagy amelyekre az alapértelmezett értéket akarják használni. Ha egy függvény utolsó paraméterét használod, az összes előtte lévőt meg kell adnod. Ha az utolsó paramétert nem használod, elhagyhatod. Ha az azt megelőző paramétert sem használod, azt is elhagyhatod. A kevésbé használt paraméterek a sor végére helyezésével egyszerűbbé teheted elhagyásukat.

A következő példában a $text kötelező, mert a szöveget mindig meg kell adni. A szöveg nélküli <font>-tag semmit nem csinál egy weboldalon, így a szövegmező kötelező, de a $color-paraméter opcionális, mert a legtöbb ember úgyis az alapértelmezett pirosat hasz> nálja:

```
function message($text, $color="red")

    return ( "<brxf ont color=\"" . $color . » $text [11]</font>[1]

\ » > » print(message("bright red error

message")) ;
```

A $color-mező alapértelmezett értéket kap, amelyet akkor kell használni, ha a felhasználó nem ad meg Scolor-értéket a függvénynek. Az alapértelmezett érték logikailag *ugyanaz*, mintha a függvény első sora ezt tartalmazná:

```
if ( lisset(Scolor)) {$color = "red";}
```

A következő kódot nem olyan könnyű használni, mert egyik paramétert sem hagyhatod ki, ha bármelyiket is használod. Ha a paramétereket ilyen sorrendbe helyezed, nem hagyhatod el a $color-t, mert a $text-et mindig meg kell határozni:

```
function message($color-"red", $text)

    return ("<brxfont color = $color "\">" $text '</font>'
    \""

print(message(, "bright red error message"));
```

A print() a messageQ által eredményezett sztringet jeleníti meg, de a hiányzó colorpara-méter miatt a message() szintaktikai hibát okoz.

Változó számú beviteli mezők

Jóllehet az opcionális értékek nagyon hasznosak lehetnek, előfordulhat, hogy nem tudod, hány paraméter kerülhet a függvénybe. A PHP4 lehetővé teszi, hogy megállapítsd, hány paramétere van a függvénynek, és ennek megfelelően használd. A függvényedben a paraméterek változó hosszúságú listájának kezelésére a func_num_args()-, func_get_arg()- és func_get_args()-függvényeket használhatod.

A red()-függvényt megváltoztathatod erre:

```
function red()
    {
    $red_text = "";                                    :
    $red_count = func_num_args();
```

```
for($i = 0; $i < $red_count;
    {$red_text .= func_get_arg
return ( "<brxfont color=\ " red\ " $red,,text [1]</font>[1]
>"
```

A módosított red()-függvényben nincsenek az első, **function** red()-sorban meghatározott paraméterek, és a paraméterek dekódolása manuálisan történik. A **func_num_args(),** a **red()**-függvényben használt paraméterek számát adja meg. A for()-ciklus végigfut a paraméterek listáján, a **func_get_arg()** veszi a következő paramétert, ami a szöveg egy sora, majd az összes beviteli szöveget összegyűjti a $red_text-ben és megjeleníti a **<font>** tag-ek között. Minden sor szöveg végén a <br> beszúrásával sortörést hoz létre. Gondold végig a következőt:

```
print(red("red text", "more red text"));
```

Ez a következő két szöveges sort eredményezi:

```
red text more
red text
```

A meghatározott paraméterekre a sorszámukkal is hivatkozhatsz, mintha a paraméterek egy tömb elemei lenének:

```
$red_text = func_get_arg(3);
```

Ezt a tulajdonságot kihasználtam az előző példában, a számot $i-vel adva meg. Mindig a számokat használd, ha **a** paraméterek pozícióját valami könnyen megjegyezhető és megszámolható dologhoz tudod kötni. Például használhatod ezt a tulajdonságot olyan függvényekben, amelyek sorszámozott listákat fogadnak el. Képzelj el egy függvényt, amely egy kocsi sebességfokozatait fogadja el paraméternek. A paraméter 1 az első sebesség, a paraméter 2 a második sebesség, és így tovább.

Az összes paramétert eltárolhatod egy tömbben, majd használj tömbfeldolgozást minden egyes alkalommal, amikor az a legcélszerűbb. **A func_get_args()** a paramétereket tömbbe rendezi. Úgy futhatsz ciklussal végig a tömbön, ahogy csak akarsz. A func_get_args() így működik:

```
$red_array = func_get_args() ;
```

A rögzített stílusú kötelező paramétereket kombinálhatod az opcionális paraméterek változó listájával. Ha három normál beviteli mezőt határozol meg, majd a változó lista feldolgozást használod, az első három beviteli mező a névvel ellátott mezőkben lesz, nullától kettőig számozva. A lista változó részében levő mezőkhöz csak ezt a három elemet kell a paramétereket tartalmazó tömbben átugranod.

A következő példa az *előző* red()-függvény három rögzített pozíciójú paraméterrel rendelkezik úgy mint a **$date, $time** és **$place.** A függvény a korábban is használt **for()**-ciklussal fut végig az opcionális paramétereken, azzal a különbséggel, hogy a ciklus a 0 helyett 3-mal kezdődik:

A"

```
function red($date, $time,
    $place) {
    $red_text = $date . " " . $place
    $time $red_count =
    func_num_args(); for($i = 3; $i
    < $red_count; Si+ {$red_text " <br>";}
    .= func_get_arg($i ) return ( "<brxf $red_text "</Eont>") ,-
    ont color=\ " red\ " >" .
```

Hatáskör

A hatáskör (scope) szó a görög jel vagy cél jelentésű szkopósz szóból származik. A scope megtalálható az olyan szavakban, mint a *mikroszkóp* vagy *teleszkóp*, amelyek mind egy adott területre való ráközelítés eszközei. Amikor ezt a kifejezést függvényekkel és változókkal kapcsolatban használjuk, a hatáskör azt jelzi, hogy a kód mely részében látod ezeket az elemeket, és mely részében lehet használni őket. PHP-ban a hatáskör szabályai a következők:

- A függvénybe változólistával bevitt változók lokális változók, kivéve, ha &-jel van előttük.
- A függvényen belül definiált változók lokális változók, kivéve, ha global-utasításban nevezted el őket.
- A függvényen kívül definiált változók nem érhetők el a függvény belsejében, kivéve, ha paraméterként viszed be a függvénybe, illetve ha global utasításban vannak defini álva.
- A defineQ-utasítással definiált konstansok mindig globálisak, függetlenül attól, hogy függvényen belül vagy kívül definiálták őket.
- A függvények mindig globálisak, így egy másik függvény belsejében definiált függ vény mindenhol elérhető.
- A hatáskör egyike azoknak a dolgoknak, amelyeknél eltart egy ideig, amíg az ember megjegyzi őket, mert nyelvenként eltérőek. Microsoft Visual Basic-ben a változót ak kor deklarálod globálisnak, amikor először definiálod a program elején. PHP-ban nem deklarálsz globálisnak egy változót a program fő részében. Ehelyett az adott függvé nynek megadod, hogy hogyan férjen hozzá a függvényen kívül definiált változóhoz. A függvényen belül a változó globális lesz, de a többi függvényben csak akkor, ha azok is globálisként definiálják a változót.

A PHP-féle megközelítés biztonságosabb. Ha a Visual Basic megközelítését használod, majd valaki másnak a függvényét ágyazod be a kódodba, megszakíthatod a függvényt azáltal, ha ugyanolyan nevű globális változót definiáltál, mint amilyet a függvény lokális változóként használ. A Visual Basic-ben programozók ezért gyakran összetett elnevezési szabályokat alkalmaznak az ilyen hiba megelőzésére. PHP-ban a függvényt író dönti el, hogy lokális vagy globális változót akar.

Gondold végig például a következő kódot: ;

^^r

```
$a = "A short sentence";
```

```
$b = " ";  // the character that separates words $c
= 0;    // count of separation characters $x =
strlen($a); for($y =0 ;  $y < $x; $y++)

    if(substr($a, $y, 1) == $b)  {$c++;}
```

Ez a kód a szavak közötti szóközöket számolja meg egy mondatban. A $a-változó tartalmazza a mondatot, a $b a szóközkaraktert, a $c a szóközök számát, a substrQ pedig a függvényt, amely minden egyes szóközt megtalált. A $x tárolja a $a-ban lévő karakterek számát, és a for()-ciklus végigfuttatja a substr()-t a $a karakterein. Jóllehet máshogyan is meg lehet a szóközöket számolni, ezzel a példával könnyen meg lehet magyarázni a változók PHP-függvényeken belüli hatáskörét.

A kódnak a függvénybe való különválasztásával anélkül tudod azt a függvényben bármikor megváltoztatni, hogy a kód egyes előfordulásait mind meg kellene keresned. A függvényben használt változókat is kiemelheted a függvényen kívüli kódból. Ha egy hosszú szkriptben sokszor használod a $x-változót, nehéz lehet minden egyes előfordulásnál megállapítani, hogy az adott helyen melyik kód állítja be a változót.

Próbáld a kódot függvényekbe szétválasztani. Tartsd az egyes $x-ek elkülönített használatát a függvényen belül. Az előző példában a $a-, $b- és $c-változókra a szóközök megszámolása után is szükség van, tartsd hát meg őket a kód fő részében. A $x- és $y-változókra viszont nincsen szükség a szóközök megszámolása után, így különítsd el őket a függvénybe. így a $x és $y hatásköre a függvényre korlátozott. A beágyazott függvénnyel így néz ki a kód:

```
$a = "A short sentence";
$b = " ",•  // the character that separates
words
$c = 0; // count of separation characters
function count_spaces($input,
    $separator) {
    $x = strlen($input);
    $z = 0;
    for($y = 0; $y < $x;
        $y++) {
        if(substr($input, $y, 1) == $separator)
        {$z++;} }
return($z); } $c =
count_spaces($a, $b);
```

Ebben a példában a $a, $b, $c, $x, $y, for() és substr() pontosan azt teszik, mint az előzőben. Az egyetlen különbség abban van, hogy hol láthatók a változók: a program fő részében ($a, $b, $c) vagy a függvény hatáskörében ($x, $y, $z).

Ha sok olyan beviteli mező van, amelyet módosítani kell, hagyd ki őket a függvényfejből és global-utasítással add be őket. A global utasítja a függvényt, hogy használja a függvényen kívül definiált, létező változókat. A változókat *globális változónak* nevezik, és a hatáskörük mindenhova elér. / -

A következő példa egy korábbi példában használt numbered_message()-függvény egy részét mutatja, annyiban megváltoztatva, hogy globális változót használ. A numbered_message() a függvényen kívül definiált $message_count-változót használja, amelyhez a global-utasítással férhet hozzá:

```
Smessage_count = 0;
functíon numbered_message($text)


    global $message_count;
    $message_count + +;
```

A globális változók révén a függvényed többszörös értéket is visszaadhat, egyszerűen az összes változónak egy global-utasításban történő listázásával. Az előző példa egy változót definiált a global-utasításban, de bármennyit megnevezhetne. A függvény a változók bármelyikét megváltoztathatja, igazat vagy hamisat adva a siker vagy hiba jelzésére, és lehetővé teheti, hogy a kód többi része az összes módosított változóhoz hozzáférjen.

Az előző példában a változók globálissá tételének hagyományos módját láttuk: a változót a szkript elején definiálja, majd minden olyan függvényben, amely azt használja, globálisnak deklarálja. A változónak nem kell abban a pontban léteznie, ahol a global-utasításban elnevezed. A függvényed az adatfeldolgozás után dönthet úgy, hogy új változót hoz létre, és egy global-utasításban nevezi el a változót. A változónak nem kell az előtt léteznie, hogy a global-utasításban definiálnád, illetve a global-utasítás lehet feltételes utasításon belül, hogy a változó csak a függvény bizonyos részeinek használatakor jöjjön létre. Lehet olyan függvényed, amely return()-utasítással siker esetén mindig igazat, hiba esetén mindig hamisat ad eredményül, és csak akkor hoz létre egy hibaüzenetet tartalmazó globális változót, ha az eredménye hamis.

A globális változók függvényekben való bonyolult használata nem csak azzal jár, hogy a függvényeket nehéz lesz megtanulni és megjegyezni, hanem a szkriptben lévő hibákat is problémás lesz észrevenni. A függvényedet használó emberek annyit fognak látni, hogy a változó megváltozott, de azt nem fogják látni, hogy hol. A globális változók használatát mindig körültekintően dokumentáld.

A PHP-ban az össze függvénynév globális. Habár egy függvény másik függvényen belüli definiálásával egymásba ágyazhatsz függvényeket, ahogy azt a következő példában látod, a függvények globális jellege miatt az egymásba ágyazás ésszerűtlen. A következő példában a belső bold()-függvény az error_message()-függvényen kívül is elérhető, így semmi ok nincsen a bold()-ot az error_message()-en belülre helyezni:

```
function error_message($text)
    {
    function bold($text) {return("<strong>$text</strong>");}
    return ("<brxf ont color = \ " red\ " >" . bold($text) . "</font>");
    }
print(error_message("bright red error message")); print
(bold ("bold message")),-
```

Legfeljebb nagyon hosszú szkriptek esetén van értelme így használni a függvényeket, ha a szkript a függvények egyik felét egyféle célra, másik felét pedig más célra használja.

A szkriptet kisebb, jobban kezelhető részekre bonthatod, ahogyan az a következő bekezdésekbe le van írva, de mindig figyelembe kell venni a használt változók hatáskörét. Ha számtalan gyakran használt változót továbbítasz a részek között, a kód bonyolulttá, a változók nyomon követése pedig összetetté válhat. Ha egy függvényben sok lokális változó és csak néhány globális változó vagy paraméter van, abból egy könnyen érthető kódrészlet lesz. Ha a legtöbb változó globális, lehet, hogy a kódrészletet nehéz megérteni, és jobban jársz, ha a kódot eltávolítod a függvényből és visszahelyezed az eredeti pozíciójába.

Egy külön kérésre készített kocsi rendelésére használt szkript A függvényhalmazt használja az összes kocsi, B függvényhalmazt az sportos terepjáró és C függvényhalmazt a felnyitható tetejű sportkocsik rendelésekor. A függvényhalmazokat feloszthatod úgy, hogy azokat az a.html, b.html és c.html fájlok tartalmazzák. Ezután használj a következő kódban lévő függvényhez hasonlót a megfelelő fájl behívására:

```
function
    select_includes($model) {
    include("a.html");
    if($model    ==    "Explorer")
        <                                    r
        include("b.html");
```

A példa csak a kód részletét mutatja annak érdekében, hogy legyen némi elképzelésed arról, hogy hogyan kezelheted a függvények nagy halmazait. A példa azért működik, mert a függvények bárhol elérhetők, akkor is, ha másik függvénybe vannak beágyazva.

Ha nagy szkriptet tervezel, mérlegeld annak lehetőségét, hogy a szkriptet elkülönült részekre bontod, és mindegyik részre függvényt készítesz. Ügyelj az adatok hatáskörére, és használj minél több korlátozott használatú változót. Éppen úgy, ahogy egy weboldal designját is fej- és törzsrészre bontod, a szkriptedet is felbonthatod. Lehet egy olyan függvényed, amely minden adatot összegyűjt az adatbázisból, és egy másik, amely elvégzi a teljes oldalformázást.

Statikus változók

Tegyük fel, hogy van egy függvény, amelyet többször használsz a szkriptben, és használat után értéket kell a következő használatba továbbítania. Azt is tételezzük fel, hogy az értéket csak abban a függvényben használod. Az értéket eltárolhatod egy globális változóban, amelyet mindannyiszor használsz, ahányszor a függvényt behívod. Ennél egyszerűbb és lo-gikusabb azonban a *statikus változók* használata. íme egy példa arra, mit kellene tenned, ha nem lennének statikus változók:

```
$raessage_count    = 0 ;      //    raessage_count    is    used    by
nurabered_message() function    numbered_message($text,     $color="red") {
    global    $message_count;
    $Inessage_count++; return (
    "<brxfont    color=\" .          .
    $message_count    .    "    "        $color
                                              </font>"
```

A $message_count-változót létrehozhatod a numbered__message()-függvényen kívül, így a $message_count értéke a numbered_message()-en belüli használat után megmarad. A numbered_message()-függvényen belül global-utasítással férhetsz hozzá a $message_count-hoz.

Statikus változókkal nincs szükség a $message_count külső definíciójára, hanem helyettesítsd a global-utasítást a static-változódefiniálással, ahogy azt a következő példa kiemelt sorában láthatod:

```
function numbered_message($text, $color="red")

    static $message_count;
    $message_countT+; return (
    "<brxf ont color=\"" .          $color .
    $message_count . " " . "\">" $text .
                                    "</font>"
```

Mi történik a $message_count-tal, ha a numbered_message()-t többször behívod? Ellenőrizd a statikus változót a numbered_message() két print-utasításban való használatá val, mint itt: *r*

```
print(numbered_message("bright red error message"));
print(numbered_message("another bright red error message"));
```

Ez lesz az eredmény:

```
1 bright red error message
2 another error message
```

A $message_count értéke megmarad a numbered_message() használatai között, és nem jelenik meg a numbered_message()-en kívül. Újra és újra létrehoz egy hagyományos változót, amikor használja a függvényt, de a statikus változó változatlan marad. A statikus és hagyományos változók minden másban teljesen megegyeznek. A statikus változók lokális hatáskörűek, így nem oszthatók meg a függvények között, és nem érhetők el a függvényen kívül.

Ha olyan függvényt írsz, amely minden egyes használatakor hosszadalmas műveletekkel kiszámolja ugyanazokat az értékeket, jobban jársz, ha az értékeket statikus változókban őrzöd meg, így csökkentve a műveleti időt. Ha a $message_count kiinduló értéke egy hosszú kód eredménye lenne, a kiinduló érték újraszámolását megspórolhatod, ha leteszteled, hogy az érték be van-e állítva. Az issetQ-függvény igazat ad eredményül, ha a változó be van állítva, az !isset()-függvény eredménye pedig akkor igaz, amikor még nincsen. A következő példa a hiányzó változó tesztelését mutatja, és lefuttatja a hosszú kódot (amit itt a három pont jelképez):

```
If(!isset($message_count))
    { $message_count = ...
```

Rekurzió

PHP-ban a függvények - a *rekurziónak* nevezett művelettel - behívhatják saját magukat. A rekurzió nehézzé teheti kódod megértését, így óvakodj tőle. PHP3-ban a rekurzió különösen lassú, mert a szknptet soronként dolgozza fel, és egy függvény rekurziója rengeteg CPU-időt használ a függvény újraértelmezésére. A PHP4 inkább compiler típusú nyelvként működik, így az értelmezés eredményét elmenti és újra használja, így a rekurzió itt kevésbé időigényes.

A következő példa rekurziót használ. Habár egyszerűbben is meg lehet ezt csinálni, és rekurzió nélkül gyorsabban futna, ez a legkevésbé összetett példa, amit ki tudtam találni. (Ahhoz, hogy a rekurzióra igazán hasznos példát találjak, 18 csésze kávé és 200 oldalnyi szöveg kell - és ez csak a kód első sorához elég.) A következő példa egy pontot távolít el a sztring végéről. Ha a következő karakter is pont, a függvény behívja saját magát, hogy eltávolítsa a következő pontot:

```
$string_with_lots_of_dots = "dots.....";
function remove_dots($text)
    í
    if(substr($text, -1) == ".") {Stext = substr(Stext, 0,
    -1);} if{substr{$text, -1) == ".") {Stext =
    remove_dots(Stext);} return($text};

print(remove_dots($string_with_lots_of_dots));
```

Az első substrQ megkeresi az első pontot, és ha van pont, akkor - ugyanabban a sorban -behívja a második substr()-t, hogy eltávolítsa az első pontot. A harmadik substrQ a kiemelt sorban a második pontot vizsgálja, és behívja a remove_dot()-ot, hogy megtámadja.

Változóban elnevezett függvények

A függvényneveket lehet változókban tárolni. Hasonlóan a rekurzióhoz, ennek is kevés előnye van, és megnehezíti a kódod megértését. Ebben a részben egy példán megyünk végig, utána pedig elmondom az előnyeit és a dokumentációt. Itt van egy szkript példája, amely több adatbázishoz hozzáfér. A példa olyan függvényeket tartalmaz, amelyek hibainformációt kapnak a MySQL-től és az Oracle-től:

```
function mysql_error ( ) {...}
function oracle_error ()
{...}
```

Rakhatsz a szkriptbe olyan kódot, amely ellenőrzi, hogy éppen melyik adatbázissal dolgozol, majd a megfelelő függvényt választod ki használatra. Vagy ellenőrizheted az adatbázist egyszer, és az eredményt - a következő használathoz - tárolhatod egy változóban. A példa következő részében a kód egy nem mutatott része eldönti, hogy melyik adatbázis van használatban. A példában ez a MySQL, így a változóba a MySQL hibafeldolgozó függvény neve kerül:

```
$database_error = "mysql_error" ;                    ;"    :
```

Most már kiegészítheted a kódot a $database_error()-re való hivatkozással, és ha új adatbázist adsz hozzá, a hivatkozást nem kell megváltoztatni. Például:

```
print($database_error())•
```

Figyeld meg a Sdatabase_error() részt. Mielőtt a PHP megpróbálja elvégezni a függvényt, a $database_error()-t a $database_error() tartalmára cseréli. Ezután elvégzi az mysql_error()-t, mivel a $database_error a mysql_error()-t tartalmazza.

A call_user_func() nevű különleges függvénnyel máshogyan is be lehet hívni változókban elnevezett függvényeket (erre a fejezet „Gyors megoldások" részében találsz példát). Egy új függvény, a call_user_func_array() egy lépéssel tovább megy, és lehetővé teszi, hogy egy, a behívott függvény paramétereit tartalmazó tömböt használj. Havonta bukkannak fel új megoldások, amelyekkel több-kevesebb kódot megspórolhatsz. Azt használd, amit a legegyszerűbb megértened, megvalósítanod és elmagyaráznod.

Bizonyos szkriptek esetében jobb megközelítés a hagyományos függvények szokásos használata, kerülve a változókban elnevezett vagy a call_user_func()-hoz hasonló függvények használatát, az állapotinformációkat tömbökben, adatbázisokban vagy XML-ben tárolva. A választás az információ terjedelmétől és a használat gyakoriságától függ. Példákat a 3., 5. és 20. fejezetben találsz.

A függvénynevek változóba tétele akkor működik, ha a függvénynév egy állapotot vagy műveletet ír le, könnyen megjegyezhető módon. Ez a technika skálázható, mert minden egyes függvényen dolgozhat más-más személy. Rendkívül hasznos eljárásnak bizonyul olyankor, amikor nagy számú vagy nagyméretű függvénnyel kell dolgozni, és a teljes kód kezelhetetlen lenne if()- vagy switch{}-utasításokban.

A függvénynevet tartalmazó változó használatát dokumentálnod kell, mert a kódodon dol gozó következő emberek számára a használata már nem lesz egyértelmű. Sztring típusú változóba könnyen kerülhet érvénytelen input, és a szkript nem fog működni, ha a változó érvénytelen változónevet tartalmaz, így győződj meg róla, hogy a változó csak érvényes függvényneveket kaphat. \

Sorrend

PHP4-ben bármilyen sorrendben definiálhatod a függvényeket, míg PHP3-ban a függvénydefiníció meg kell, hogy *előzze* a függvény használatát. Definiáld az összes függvényt egy helyen, és mérlegeld a gyakran használt függvények külön fájlba való elhelyezését, amelyet beágyazhatsz az egyes oldalakba. Ha egy függvény változót használ, ellenőrizd, hogy a változó létezik-e, mielőtt a függvény használná. Ha behívsz egy függvényt, gondosan ellenőrizd, hogy az összes szükséges értéket megadtad-e a paraméterekbe. De mi történik azokkal a változókkal, amelyeket global-utasítással ér el? Bizonyosod) meg arról, hogy az összes olyan változót létrehoztad, amit a függvény global-utasításon keresztül ér el, és azokat is, amelyeket a függvényed által behívott függvények érnek el.

Gyors megoldások

Függvény létrehozása

Függvény létrehozásához a **function** kulcsszó, egy függvénynév, zárójel, majd kapcsos zárójel szükséges, mint itt:

```
function    any_name()
```

A függvénynévnek egyedinek kell lennie. A függvénynév nem tesz különbséget a kis- és nagybetű között, így nem hozhatsz létre Car() néven függvényt, ha a **car()-t** már korábban definiáltad. A **function** szó és a függvénynév közé szóköz kell, de a név és az első zárójel, illetve a második zárójel és az első kapcsos zárójel között lehet nulla, egy vagy sok szóköz, tabulátor vagy új sorok.

Egy értéket a returnQ-utasítással kaphatsz vissza. Hogy lásd, hogyan működik, vess egy pillantást a következő függvényre, amely egy honlap aktuális oldalának elérési útját adja vissza:

```
function path_to_page()
    {
    global $PATH_INFO,$PHP_SELF;
    if(isset($PATH_INFO))
        {
        return($PATH_INFO) ;
        }
    elseif(isset($PHP_SELF))
        {
 return($PHP_SELF) ;
```

Ezután, ha a függvényt a honlapod php/-könyvtárának indexoldalára rakod be, a path_to_page() a következőt írja ki:

```
/php/ index. htinl
```

Megjegyzés: A $PATH_INFO és $PHP_SELF ugyanazt az információt adják, és közülük bármelyik hiányozhat. Olyat azonban még sosem láttam, hogy mindkettő hiányzott volna.

Paraméterként bármilyen értéket, még nullát is beírhatsz. Nem kell paraméter a dateQ-függvénybe, mert az a szükséges inputot a rendszerórából kapja. Ezzel ellentétben egy grafikus képet létrehozó függvény több tucat paramétert fogad a színek és koordináták meghatározására. A következő' az *előző* példát folytatja, de némi üzenetszöveget tesz *hozzá:*

```
function path_to_page($part_a, $part_b)

    global $PATH_INFO,$PHP_SELF;
    if(isset($PATH_INFO))

    return("<p>" . $part_a . $PATH_INFO . $part_b

elseif(isset($PHP_SELF)) return("<p>" . $part_a .

$PHP_SELF . $part_b . "</p>
```

Tételezzük fel, hogy van egy /php/Índex.html nevű oldalad, és a következő' kódot tesztként rakod az oldalra:

```
print(path_to_page ("This page is ", " . " ) ) ;
```

Ez lesz az eredmény:

```
This page is /php/index.html.
```

Globális változó deklarálása

Ha sok olyan beviteli mező van, amit változtatni kell, hagyd ki őket a függvényfejből és global-utasítással add be őket. Ezeket a változókat *globális változónak* nevezik, és a hatáskörük mindenre kiterjed.

A következő példában a $message_count a numbered_message()-en kívül van definiálva. Ahhoz, hogy a numbered_message() új változó létrehozása helyett a $message_count-ot használja, nevezd meg a $message_count()-ot a kiemelt global-utasításban:

```
$message_count = 0;
function numbered_message($text, $color="red")
    {
    global $message_count;
    $message_count++;
    return( "<brxfont color = \" " . $color .
        " \" > » . $message_count . " " .
        $text . " < / f ont>");
```

Teszteld a globális változót a következő két kódsorral, amitől azt várhatod, hogy egy l-es és egy 2-es számú üzenetet eredményez:

```
print(numbered_message("bright red error message"));
print(numbered_message("another bright red error message"));
```

A változónak nem kell azon a ponton létezni, ahol a global-utasításban megnevezik. A függvény is létrehozhatja a változót. A global-utasítás biztosítja, hogy a függvény által létrehozott változó mindenki számára elérhető globális területen van tárolva a függvény lokális területe helyett.

A következő példában a $new_text csak akkor jön létre, ha függvényszöveget talál a $text-változóban. A kód a global-utasítást használja, és a $new_text-változót akkor hozza létre, ha az if () igaz:

```
function    create_text($text) {
    if(isset($text)    and
    strlen($text))

        global    $new_text;
    $new_text    .=    "<font    color=\"red\" .
    $text    .    "</font>"; } return(true);
```

Ha lemásolod az előző példát, és a globális változókat csak akkor definiálod, amikor azok adattal töltődnek fel, mások számára probléma lesz, hogy kitalálják, mely változókat használsz a függvényben. A creat_text()-példát lehet fejleszteni, ha a szkriptet valami hasonlóval kezded:

```
$new_text =      // This variable is fiiled with text by create_text
```

Statikus változó deklarálása

Minden egyes alkalommal, amikor függvényt használsz, a függvényben minden ideiglenes adattárból jön létre, ami időveszteséget okoz, és azt jelenti, hogy nem tudod az értékeket a függvényen belül elmenteni. Előfordul, hogy amikor a függvény egyik használatából származó értékeket el akarod menteni a következő használatig, rengeteg időt töltesz a változók függvényen kívüli definiálásával, illetve a global-utasításokkal, hogy a változók a függvényben is elérhetők legyenek. A statikus változók ezen segítenek.

A numbered_message()-függvény úgy ír ki üzeneteket, hogy minden egyes üzenet egyedi számot tartalmaz, amelyet a numbered_message() állít elő. Ha a numbered_message() hagyományos változó lenne, a numbered_message() minden elindításakor nullát tartalmazna. A $message_count statikussá deklarálásával a $message_count megőrzi értékét a numbered_message() behívásakor:

```
function    numbered_message(Stext,     $color="red")

    static $message_count;
    $message_count:++ ; return
    ("<brxf ont color = .          . $color . »\">"
    $message_count . "          $text . "</font>")
```

A változtatást a következővel ellenőrizheted:

```
print(numbered_message("bright    red    error    message"));
print(nuinbered_message("another    bright    red    error
message"))
                                                    = ipa?'
```

EZ az eredménye:

```
1 bright red error message
2 another error message
```

Függvény tárolása változóban

Tegyük fel, hogy céged raktárkészletének változását akarod rekordokkal nyomon követni, és a rekordokat lemezen vagy adatbázisban akarod tárolni. íme a kód, amellyel ugyanazt a feladatot szekvenciális Comma Separated Variable fájllal (CSV) vagy MySQL adatbázis-táblázattal lehet elvégezni. A feldolgozófüggvény változóban való tárolásával a példa automatikusan kiválasztja a megfelelő fájlt és feldolgozófüggvényt.

A példarekord két, egy leíró ($part) és egy mennyiségi **($quantity)** elemet tartalmaz:

```
$part = "bolt";
$quantity = 3;
```

íme a függvény, amely a rekordot a movements.csv nevű fájlhoz adja:

```
function add_file(Spart, $quantity)
    {
    $result = falsé; if($file =
    fopen("movements.csv", "a"))

    $result = fwrite($file,
    $part fclose($file); }                    $quantity);
    return($result);
```

Az fopen()-függvény megnyitja az első paraméterében megnevezett fájlt, a második paraméter, az "a" pedig új rekordokat ad a létező fájl végéhez. A **$file** változó egy a megnyitott fájlra vonatkozó azonosítót kap, az fwrite() pedig az azonosító használatával sztringet ír a fájl végéhez. Az fclose() bezárja a fájlt, hogy mások hozzáférjenek. A **$result** változó igaz vagy hamis értéket kap a fájlírás sikerétől, illetve sikertelenségétől függően, és ez a változó jut vissza hozzád, így ellenőrizheted az eredményt:

Hivatkozás: **oldal:**

Szövegfájl megjelenítése **279**

A következő kód a rekordot a movements nevű MySQL-táblázathoz adja. A MySQL-kapcsolat már létezik, és az adatbázis nevét pedig a $database globális változó tartalmazza. A táblázat kulcsa egy egész szám, és a kulcs auto_increment-re van állítva, így nem kell a kulcsértéket meghatároznod:

```
function add_mysql($part, $qúantity) { $sql =
    "insert intő movements set part='" . Spart
```

```
. "', quantity='" . Squantity . "'";
return(mysql_db_query($database, $sql));
}
```

Valahol az alkalmazás-alapbeállításokban definiálsz egy változót, amely meghatározza, hogy az **add_file()-t** vagy az **add_mysql()-t** használja:

```
$add_stock = "add_mysql";
```

Ha ezt megtetted, a készletváltozásokat a megfelelő fájltípushoz ezzel adhatod meg:

```
$add_stock("bolt", 3);
```

Ezután egy kis logikával rögtön testre szabhatod a teljes készletnyilvántartó alkalmazásodat, hogy az a függvényekbe a megfelelő' fájlfeldolgozást rakja:

```
$file_type = "mysql"; $add_stock = "add_"
. $file_type; $delete_stock = "delete_" .
$file_type; $count_stock = "count_" .
$file_type; $print_stock = "print_" .
$file_type;
```

Alapértelmezett függvényparaméterek használata

Az alapértelmezett függvényparaméterek a függvény által akkor használandó értékeket tartalmazzák, amikor a felhasználó nem határozza meg a paramétereket. A következő példában a red()-függvény alapértelmezett figyelmeztető üzenete az alábbi:

```
function red($text="Warning, unidentified error!")
```

Ha a szkriptedbe ezt írod:

```
print(red());
```

akkor a következő egyszerű figyelmeztetés jelzi, hogy valami nem működik:

```
Warning: Unidentified
error! (Figyelem: ismeretlen
hiba)
```

Egy ehhez hasonló alapértelmezett beállítás jelzi, hogy a kód elért a **red()**-függvényhez, de elfelejtettél vagy nem tudsz hasznos értelmező üzenetet megadni. A szkript legalább azt azonosítja, hogy hol történt a hiba, még ha az okát nem is.

A kötelező és opcionális mezőket kombinálhatod. A kötelező mezők kerülnek előbbre, hogy ezáltal a PHP összepárosíthassa a mezőket. A következő példában a **$text** kötelező, a **Scolor** pedig opcionális:

```
function message($text, $color="red")

    return ( "<brxf ont color=\"" . $color . "\">" . $text .

"</font>"); print(message("bright red error message"));
```

Az eredmény a következő piros betűs szöveg:

bright red error message

Ez is mutatja, hogy a függvény tökéletesen működött a második paraméter nélkül, és a függvénydefiniálásnál meghatározott alapértelmezett pirosat használta.

A következő példa nem működik, mert a message()-ben nincsen megadva érték az első paraméternek. Erre a PHP úgy reagál, mint a szomszédjaid, ha hajnali háromkor felhangosí-tod a magnód. Hibaüzenettel reklamál, és leállítja a szkripted értelmezését. A PHP-ban az utolsó megadott paraméter előtti összes paraméterre szükség van; egyetlen olyan paramétert sem hagyhatsz ki, amelyet még másik követ:

```
function message($color="red", .$text)

    return ( "<brxf ont color=\"" . $color . "\">» . $text .

"</font>"); print(message(, "bright red error message"));
```

A függvény létezésének ellenőrzése

Ha fájlokban hozol létre olyan függvényeket, amelyek más fájlokba is be vannak ágyazva, előfordul, hogy egy függvényt kétszer hozol létre, ami olyan hibához vezet, ami a szkript futását leállítja. Ez akkor gyakori, ha létrehozod az a.html-t, amelyet a b.html-be és a c.html-be is beágyazol. Később valaki a b.html-t és a c.html-t beágyazza a d.html-be. Az eredmény a kétszeresen beágyazott a.html. Megelőzheted ezt, ha a létrehozás előtt ellenőrződ, hogy a függvény létezik-e.

Itt van egy egyszerű príht-függvény definiálása a biztonsági ellenőrzésbe ágyazva. A kód létrehozza a print_message()-függvényt. A function_exists() igazat ad vissza, ha a print_message() már definiálva van. A !function_exists() pedig akkor eredményez igazat, ha a print_message() nem létezik, ahogy itt:

```
if{!function_exists("print_message"))

    function print_message($message)

        print ("<p>" . $message .

    "</p>"),-}
```

Van egy nagy különbség a PHP3 és PHP4 között. Hogy megtudd, mi az, próbáld ki a következő példát:

```
if(function_exists("red"))
    {print("Function red exists.");) else
{print("Function red does not exist.");}
function red($text)

    return("<font color=\"red\">"     $text
                                        "</font>");
```

```
íf(function_exists("red"))
    {print("Function    red    exísts.");} else
{print("Function    red    does    not    exist.");}
```

A PHP3 úgy fordítja a függvényeket, ahogy olvassa a szkriptet, így a red()-függvény csak az első teszt után van lefordítva. Ezért az eredmény a következő:

```
Function    red    does    not
exist. Function    red    exists.
```

A PHP4 viszont a függvényeket a kód végrehajtása előtt lefordítja. Ennek megfelelően az eredmény:

```
Function    red
exists. Function    red
exists.
```

Ha egy függvény esetleg több helyen - például beágyazott fájlokban - kerül létrehozásra, és meg akarod előzni a hibákat, a függvény összes definíciójának tesztbe ágyazva kell lennie.

Ha egy beágyazandó fájlt hozol létre, amely több függvénydefiníciót tartalmaz, és nem akarsz az összes függvény koré function_exists()-t rakni, kérheted a felhasználókat, hogy az include() új, include_once() nevű változatát használják.

A call_user_func() használata

A call_user_func()-függvény segítségével változóban elnevezett függvényeket használhatsz. A paraméterei a következők:

```
call_user_func ($function_name,    $function_parameter_l,    ...)
```

A $function_name egy sztring, amely a használni kívánt függvény nevét tartalmazza. Mivel a $function_name sztring típusú változó, a függvény nevét alkotóelemeiből építheted fel, vagy a nevet adatbázisból is betöltheted.

A $function_parameter bármilyen paraméter, amit a $function_name-ben elnevezett függvénynek akarsz továbbítani. Lehet sok függvény paraméter is, lehet nulla is, de a számnak meg kell felelnie az alkalmazott függvénynek.

Ezzel a példával elérheted, hogy az alkalmazásod minden operációs rendszeren működjön. Segítségével bármilyen operációs rendszernek megfelelő formátumú fájlneveket építhetsz. Először definiálj egy változót, amely a tesztszerveren jelenleg használt operációs rendszer nevét tartalmazza, ahogy itt:

```
$os    =    "unix";
```

Ezután szúrj be kódot a jelenlegi operációs rendszer megállapítására. Ennek számtalan összetett módja van, de ebben az esetben elegendő egy gyors ellenőrzés, hogy a fájlneveknek xr (meghajtó és pontosvessző) formátumú meghajtónevük van-e. A pontosvessző a Windows-t jelzi, minden más Unix. A következő kód ellenőrzi a fájlnévben a pontosvesszőt, és beállítja a $os-t, hogy a Windows-t jelezze:

```
if(substr($DOCUMENT_ROOT,1,1)    ==    ":")
    $os    =    "windows";
```

Most minden operációs rendszerre írj formázófüggvényt. Vess egy pillantást a formázó-függvény Unix-os és Windows-os verziójára, hogy fűzik össze a helyes elválasztókarakter-rel az elérési utat és a fájlnevet:

```
function file_name_unix($path, $file)
   return($path . "/" . $file); function
file_name_windows($path, $file)
   return($path. "\\" . $file);
```

Megjegyzés: A Windows NT mindkét karaktert elfogadja. Ha az NT az egyetlen Windows-verzió a webszervereden, akkor használhatod a Unix-os elválasztó karaktert.

A $os változóból felépítve a függvénynevet, a call_user_func()-függvénnyel most már behívhatod a megfelelő formázófüggvényt, mint itt:

```
$full_name = call_user_func("file_name_" . $os, "help", "index.html");
```

A create_function() használata

A create_function()-nel menet közben tudsz függvényeket felépíteni. A függvénynév dinamikusan generálódik, így a név nem ütközik össze a kód egyik függvényével sem. A create_function() szintaktikája a következő:

```
create_function($parameters, $code);
```

Ez a példa a **create_function()-nel** hoz létre egy egyszerű formázófüggvényt. A **format_text()**-függvény két paramétert fogad el, a $heading-et és a $text-et, majd a Sheading-et <hl> tag-ek közé teszi, hogy első szintű HTML fejrészformázást állítson be. A formázás befejezésére ezután a függvény bekezdéstag-ek közé teszi a szöveget:

```
function format_text($heading, $text)
Sx = "<hl>"  . $heading
Sx .= "<p>"  . $text .
return(Sx);
```

A következő ugyanezt a formázófüggvényt hozza létre a **create_function()** használatával. A create_function() két sztringet fogad el. Az első sztring a függvényfej zárójelei () közé kerülő paraméterek listáját, a második pedig a kapcsos zárójelek{} közé kerülő kódot tartalmazza:

```
$format_text = create_function('$heading,
    $text', '$x = "<hl>" . $heading . "</hl>";' .
    '$x .= "<p>" . $text . "</p>"; return ($x)
    ;'),-
```

ötlet: A kétszeres idézőjelek helyett (") használj egyszerest (') a sztringek körül, így elkerülheted a speciális karakterek elé szükséges jelek használatát.

A formázófüggvény használatához a következőket írd be:

```
$formátted_text = $format_text("A poem", "The cat sat on the
mat");
```

Miután ezt megtetted, az adatbázis információját felhasználva dinamikusan építheted fel a formázófüggvényt. A látási problémákkal küzdő ügyfeleid nagyobb betűméretet választhatnak a profiljukban, majd ezt a fontméretet megadhatod a formázófüggvénynek. A következő példában a betűméret manuálisan került be a $font_size-ba. Akkor töltőd be a $font_size-ba, ha a felhasználó bejelentkezik, majd a $font_siz.e-ot session-változóként mented. A kiterjesztett formázófüggvény a következő:

```
$font_size = "+2";
$t = '$text';
                                                    . $t
$t = '"<font size=\\"'                                  "<font>
                          $font_size . '\\">"
.$format_text = create_function('$heading,
$text
    ■$x = "<hl>" . $heading $font_size
      '$x .= "<pxfont
    size=. ' . ' . $t
    .. 'return($x);'
```

Ha ezek a vissza-perjelek és idézőjelek összezavarnak, írj egy példafüggvényt először kézzel, majd próbáld a kódot idézőjelekbe rakni.

A függvények dinamikus létrehozásával feldolgozási időt takaríthatsz meg, ha a végeredményt sokszor használod a szkripten belül. Ennek ára a kód összetettsége, és változtatás esetén nagyobb a hiba bekövetkezési valószínűsége.

Afunc_get_arg() és a func_num_args() használata

A func_get_arg()-függvénnyel olyan függvényt hozhatsz létre, amely korlátlan számú paramétert elfogad anélkül, hogy a paraméterlistában definiálnod kellene őket. A func_num_args()-függvénnyel a feldolgozandó paraméterek számát kapod meg. A func_get_arg()- és a func_num_args()-függvény csak PHP4-ben használható.

íme egy egyszerű függvény, amelyet a func_get_arg()-gal majd továbbfejlesztünk:

```
function file_name ($directory, $file)

    return($directory $file);
```

Ha azt akarod, hogy a függvény könyvtárak két szintjét kezelje, két könyvtárparamétert kell a függvényben definiálnod, és mindkettőt használnod kell a returnQ-utasításban. Mit

tegyél, ha több könyvtárszint is lehet, és a szintek száma változó? A válasz a paramétereken keresztül vezető út számolása a **func_get_arg()** és a **func_num_args()** használatával.

A file_name()-függvényes példa kibővített változatában nincsen definiált paraméter. A függvény nulla, egy vagy sok paramétert is elfogad, mint azt itt megmutatom. A / elválasz-tót raktam az elérési út összetevői közé, amelyeket a **func_get_arg()** gyűjt össze:

```
function file_name()

    Spath = "";
    Sseparator = "" ■
    for($i = 0 ; $i < func_num_args
                                    func_get_argi
        Spath .= Sseparator
        Sseparator = "/";

    return(Spath);
```

Megjegyzés: Az elválasztó változóba az elválasztó karakter az első ciklus végén töltődik be, így nincsen kiindulási /. Ezt megváltoztathatod az operációs rendszertől, illetve attól függően, hogy a fájlnév az aktuális könyvtárhoz viszonyítva relatív vagy abszolút

Afunc_get_args() használata

A func_get_args()-függvény egy tömbbe teszi a függvény összes paraméterét, így egyszerű tömbműveletekkel dolgozhatsz a paraméterek hosszú listáján. A **func_get_args()**-függvény csak a PHP4-ben érhető el.

A következő példa az egészséges gyümölcsök és zöldségek listáját ábécé-sorrendben íratja ki:

```
function fruít_list() {
    $fruits =
    func_get_args(); $fruit_list =
    " " ; sort($fruits); $c =
    count($fruits) ; for($i = 0;
    $i < $c;

        $fruit__list .= "<br>" . $f ruits [ $i] ;

    return("<p>" . $fruit_list . "</p>");
```

Megjegyzés: Az elválasztó változóba az elválasztó karakter az első ciklus végén töltődik be, így nincsen kiindulási <br>. Ezt megváltoztathatod az operációs rendszertől, illetve attól függően, hogy a fájlnév az aktuális könyvtárhoz viszonyítva relatív vagy abszolút.

Shutdown-függvény beiktatása

Ha egy óriási szkriptet készítesz, amely rengeteg fájlt tölt fel és hatalmas weboldalt hoz létre, de alkalmanként nem sikerül befejeződnie, jól jöhet egy kód, amely mindig lefut a szknpt végén, függetlenül attól, hogy mi történik. Ilyen esetben a **register_shutdown_function()** a neked való megoldás.

A következő példában a függvényt **special_code()-nak** nevezem:

```
function
    speciai_code() {
    // Code to terminate transactions.
    // Code to close files and
    databases. }
```

A függvényt a következővel iktatod be:

```
register_shutdown_fuction("special_code");
```

A special_code()-ban egy dolgot kivéve bármit csinálhatsz: nem írhatsz a böngészőnek. A böngésző HTTP session-e lezárul még mielőtt a shutdown függvény elkezdődne.

Mi a teendőd? Be kell zárnod a be nem fejeződött adatbázis-tranzakciókat és azokat a fájlokat, amelyeknek új rekordjaik vannak a kimeneti pufferekben. Hárítsd el az abnormális jelenségeket, és a későbbi elemzéshez naplózd az eredményeket.

Van olyan, amit nem érdemes megcsinálni? A szkript végén a PHP automatikusan bezárja a MySQL adatbázisokat, menti a session-információt, és letisztítja a memóriát. Csak az abnormális dolgokat kell kezelned. Ellenőrizd a használt adatbázis dokumentációját.

11. fejezet

Képek

Áttekintés

Ebben a fejezetben képeket tartalmazó fájlokról lesz szó: a GIF-, JPEG-, PDF-, PNG- és SWF-formátumokról. Ezek közül néhányat biztosan használni fogsz, így mindet ismerned kell, hogy a legmegfelelőbb formátumot választhasd, vagy tudj a nem megfelelő formátum ellen érvelni, ha valaki azt szeretné, hogy olyat használj. A GIF-, JPEG- és PNG-fájlok olyan **image** előtagú PHP-függvényekkel hozhatók létre, mint például a imagecreate(). PDF-eket **cpdf_-, fdf_-** és pdf_-függvényekkel, míg SWF-eket swf_-függvényekkel hozhatsz létre.

GIF

A Graphics Interchange Format-ot (GIF) a - jelenleg az AOL tulajdonában lévő -CompuServe számára alakították ki. Ez a formátum különböző színmélységet enged 8 bitig, és tartalmazza az LZW-nek (Lempel-Ziv-Welch) nevezett tömörítési technikát, amelyet az Unisys szabadalmaztatott. Amikor a Unisys szabadalmi díjat kért az LZW-ért, fejlesztők egy baráti csoportja létrehozta helyette a fejlettebb Portable Network Graphics- (PNG-) formátumot. Ahogy a böngészőkben terjed a PNG-támogatás, úgy van egyre nagyobb sikere. A GIF 8 bites színes diagramokra, rajzokra és szöveg alapú bannerekre korlátozza a GIF-ek használatát (a fényképekre a JPEG-formátum a megfelelő).

A GIF-nek meg van az az előnye, hogy kicsi, és pontos szín- és pixelábrázolása van, de a PHP-ben csak egy könyvtárban voltak GIF létrehozására alkalmas függvények, és ezeket is törölték, amikor a Unisys pénzt kért a jogokért. A képolvasó függvények, így a getimagesize() a szabadalmi jogok megsértése nélkül használható GIF-ek beolvasására, de a szkriptedben nem hozhatsz létre új GIF-eket. Ha valamilyen ok miatt mégis GIF-et kell a szkriptedből előállítani, és tudsz COM-ot használni, kereshetsz olyan COM-os képszerkesztőt, amely képes GIF-ek előállítására, de jobb, ha áttérsz a PNG-re.

A képeidet létrehozhatod PNG-ként, majd egy külső batch-programmal átalakíthatod őket GIF-formátumra. (A külső programokkal a 7. fejezet foglalkozik.) Ha lefordítottad a PHP-dat, keress a weben egy olyan régebbi könyvtárforrást, amely tartalmazza a GIF-támogatást, és használd azt. Rengeteg munkával jár, ha GIF-et használsz, és mivel a legtöbb modern böngésző támogatja a PNG-t, miért kínlódnál? Ezek a böngészők támogatják a PNG azon tulajdonságait, amelyek a GIF-ben is megvoltak, de olyan újdonságokkal, mint az alfa-csatorna, gondjaik vannak. Emiatt a GIF-ről PNG-re való konverzió mindig működik, de a PNG összes jellemzőjét tartalmazó új PNG létrehozása nem biztos, hogy mindenhol működik.

PNG

A Portable Network Graphics- (PNG-) formátumot **(www.libpng.org/pub/png)** a GIF-ek helyettesítésére hozták létre, mert a GIF-formátum egy része szabadalom alá esik. A vele rokon Multiple-Image Network Graphics- (MNG-) formátumot **(www.libpng.org/pub/mng)** az animált GIF-ek helyettesítésére alakították ki.

A PNG-nek olyan előnyei is vannak, amelyek a GIF-fel nem elérhetők: alfa-csatorna-átlátszóság, 24 bites szín és szöveges mezők (mint a JPEG-ben). Számtalan hibaészlelőt építettek bele, így nagy képeket is a lehető legkisebb minőségromlással továbbíthatsz hírcsoportokon és e-mailen keresztül. A PNG 24 bites színei a veszteség nélküli tömörítéssel azt eredményezik, hogy a JPEG-re való konverzió előtt a PNG az ideális módja az eredeti képed tárolásának (a JPEG magasabb szintű tömörítése a képek böngészőbe való továbbítására hasznos).

A GIF-nek a JPEG-gel szembeni egyik előnye az átlátszóság, de a GIF csak az egyszínű átlátszóságot támogatja: egy színt átlátszónak foglal le, és ez minden. A GIF átlátszóságával nem hozhatsz létre félig átlátszó árnyékot. A PNG segít ezen az alfa-csatorna-átlátszósággal, amellyel fokozatos átlátszóság is elérhető. Próbálj egy GIF-et létrehozni fehér háttérrel és kék szöveggel, majd használd a anti-aliasinget az élek kisimítására. Most állítsd a fehér hátteret átlátszóra, majd helyezd a képet egy fekete lapra. Vedd észre az anti-aliasing miatt megmaradt fehér részeket. PNG-vel az élek mentén a fokozatosság az eredeti háttérkép színe helyett a háttérszínbe halványul.

JPEG

A Joint Photographic Experts Group (JPEG) az európai telekommunikációs szabványyhivatal (CCITT) és a Nemzetközi Szabványyhivatal (ISO) közös erőfeszítéseinek eredménye. A JPEG-et úgy fejlesztették ki, hogy képes legyen a fényképek online megjelenítésére, függetlenül annak nagyságától és a hálózat sebességétől. A JPEG úgy működik, hogy te állíthatod be a tömörítést 0 és 100 százalék között, így a fájl mérete az eredeti fájlméret és néhány bájt közé esik. A „tömörítés" az információ lineáris módon történő eltávolításával valósul meg, így a fokozatos leromlás végül használhatatlan foltokat eredményez.

Az információvesztést eredményező tömörítési rendszereket nem lenne szabad tömörítési rendszernek hívni! A JPEG szóhasználata, akárcsak jó néhány szoftverreklám, félrevezette az embereket, akik azt gondolhatják, hogy a fájl mérete végtelenül csökkenthető a minőség romlása nélkül. A JPEG tömörítése a színeket is megváltoztatja, az eredeti színeket közelítő színek listájából cserélve, és ez a lista a tömörítés növelésével zsugorodik. Ha egy meghatározott RGB-színt kell a vásárló lógójában használni, használd a JPEG-et nulla százalékos tömörítéssel, vagy használj más formátumot. Próbáld ki a JPEG-det több tömörítési szinttel is, mielőtt online használnád (egy jellemző tesztet találsz a petermoulding.com/jpegcompression.html címen). Jegyezd meg a legfőbb dolgokat:

- A JPEG-tömörítés alacsony kontrasztú képekhez való, nem pedig erős, fényes színekhez.
- A JPEG-tömörítés lágy képekhez és fényképekhez való, nem pedig éles vonalakhoz és élekhez.
- Minél ismertebb a kép, annál jobban észre lehet venni a tömörítést.
- A JPEG-tömörítés megváltoztatja a színeket, ezért lógókra nem használható.

PDF

A PDF az Adobe Portable Document Format-ja **(http://adobe.com).** A **pdf_-**függvények a Thomas Merz által a PDFlib GmbH-nál (www.pdflib.com) írt és a PHP-ban könyvtárként lévő PDFlib-et használják. A cpdf_-függvények a FastlO Systems (www.fastio.com) ClibPDF-könyvtárát használják. Amikor legutóbb megnéztem, nonprofit célokra mindkét könyvtár ingyenesen használható volt, és csak akkor kell licencet vásárolni, ha a könyvtárakat üzleti projektekben használják.

A PDF-formátum friss kiegészítése a Forms Data Formát (FDF). Vannak elméleti előnyei a PDF-iormátummal szemben, de újabb technológiai korlát kerül az ügyfeleid és a honlapod közé. A felhasználónak ugyanis egy plug-int kell letöltenie, mielőtt hozzáférne az oldalhoz. Mielőtt FDF-et használnál, győződj meg arról, hogy az ügyfelek a hagyományos formátumokon keresztül elérnek téged, és az egyes FDF-formátumok legyenek elérhetők hagyományos formátumokban is.

A ClibPDF dokumentációja szerint a PDF-ben a mértékegység a PostScript-pont, az inch (hüvelyk) 1/72-ed része, amely alapértelmezésben 72 egységre, azaz 1 inch-re van állítva. Kipróbáltam az alapértelmezést, és azt találtam, hogy mindig pontokat használ, így definiáltam néhány konstanst arra az esetre, ha alapértelmezéseket adok egy dokumentumnak. A PDFlib pontokat használ alapértelmezésben, és ha egy grafikát PDF-ben és más formátumokban is használhatóra akarsz megírni, tedd egy átlagos képernyőn a pontokat és pixeleket egyenlővé (a képernyők kb. 90-110 pixel per inch-esek). Az első **define()** - az amerikai, libériái és myanman emberek számára — egy inchben definiálja a pontot. A második **define()** - a világ többi részén - az egy milliméterben levő pontok számát definiálja:

```
defineCinch",
72);                    2.83464567);
define("miIlimetre",
```

Hiába állítod be pontosan a dokumentum méreteit, az Adobe Acrobat olvasója ettől teljesen függetlenül fogja a dokumentumot megjeleníteni. Azt tapasztaltam, hogy egy 10x12,5 cm-es kép tényleges méretben megjelenítve körülbelül 5x5 cm-en tűnt fel. Az alapértelmezett megjelenítési méret megváltozik, az Acrobat reader még akkor is felnagyítja a kis képeket, ha a tényleges méret van alapértelmezésként beállítva.

ClibPDF

A cpdf_-függvények a kereskedelmi PDF-könyvtárat, a ClibPDF-et használják, amelyet a dokumentáció szerint a PHP-tól külön kell letölteni, de a PHP 4.0.5 Win32 bináris állományában ez is benne van. A ClibPDF NT4 vagy Windows 2000 alatti telepítéséhez a következőket kell tenned:

1. Állítsd le az Apache-ot.
2. A php/extensions-ből másold a php_cpdf.dll-t a c:/winnt/system32-be.
3. A php.ini-ben távolítsd el a ;-t az extension = **php_cpdf.dll** elől.
4. Indítsd újra az Apache-ot.

Megjegyzés: A Windows más verzióiban a c:/winnt/system32 helyett használd a c:\windows \system-et.

A ClibPDF-fel nemrégiben vált lehetővé egyszerre több dokumentum megnyitása, de a PDF többletráfordítása miatt nem célszerű a PDF-et arra használnod, hogy menet közben dinamikusan hozza létre az egyes weboldalakat. Jelen állapotban a PDF-dokumentumok előállítása egy website-adminisztrátorra korlátozódik, aki az új dokumentumok PDF-es változatát készíti el. Ebben a fejezetben gyorsan végigszaladok a cpdf_-függvényeken, a leírtakat vonatkoztathatod a pdf_ megfelelelőjükre. Ezen függvények szerzői azt javasolják, hogy használat előtt olvasd el a ClibPDF-dokumentációt. A Gyors megoldások PDF-es példái pdf_-függvényeket használnak.

Dokumentumfüggvények

A **cpdf_open()** egy új dokumentumot nyit meg és dokumentumazonosítót ad, amelyet minden más ClibPDF-függvényben használnod kell. Választhatsz, hogy a dokumentumot memóriában építed fel, vagy a függvényben megnevezve a fájlt, egyenesen abba írod. Én a példákban a dokumentumokat a memóriában hozom létre, ezért a php.ini-ben a memory_limit-paraméterrel több memóriát juttatok a PHP-nek. Ha nagyon hosszú fájlt hozol létre, például egy adatbázist konvertálsz egy jelentésbe, akkor jobb egyenesen a fájlba írni. A jelenlegi ClibPDF nem teszi lehetővé, hogy egyszerre egynél több dokumentumot hozz létre, de a későbbi verziókban ez majd lehetővé válik.

A **cpdf_page_init()** új oldalt kezd, és bár oldalszintű függvény, mégis itt említem, mert a PHP 4.0.5 összeomlik, ha az első oldal inicializálása nélkül próbálsz írni egy dokumentumba. A **cpdf_set_font()** olyan szövegszintű függvény, amelyet legalább dokumentumonként egyszer használni kell, különben a megjelenítéskor a dokumentum a következő üzenetet eredményezi:

```
Error processing a page. Font has not been set. (Hiba
az oldal feldolgozása során. A betűtípus nincs beállítva.)
```

A cpdf_finalize()-nak egy dokumentumazonosítót adva, az befejezi a dokumentum felépítését. A **cpdf_output_buffer()** a memóriapufferből a képernyőre küldi a dokumentumot, de PHP 4.0.5-ben ebben a függvényben nincsenek benne a megfelelő fejlécek ahhoz, hogy a dokumentumot PDF-fájlként dolgozza fel, ezért a dokumentum egyszerű szövegként jelenik meg. Ha HTML-oldal helyett PDF-oldalt küldesz, a **cpdf_output_buffer()** elé a következő kódot szúrd be:

```
Header("Content-type: application/pdf");
```

A **cpdf_save_to_file()** segítségével fájlba mentheted a dokumentumot. Ezt azelőtt kell lefuttatnod, mielőtt az cpdf_close()-zal bezárnád a dokumentumot. A fejezet példáiban a fájlnév nincsen a cpdf_open()-ben megadva, és a **cpdf_save_to_file()-t** használom. A **cpdf_close()** bezárja és törli a memóriából a dokumentumot.

Oldalfüggvények

A cpdf_page_init()-be dokumentumazonosító, oldalszám, tájolásjelző, magasság, szélesség, és a PostScript-pontokban meghatározott opcionális mértékegység kerülhet.

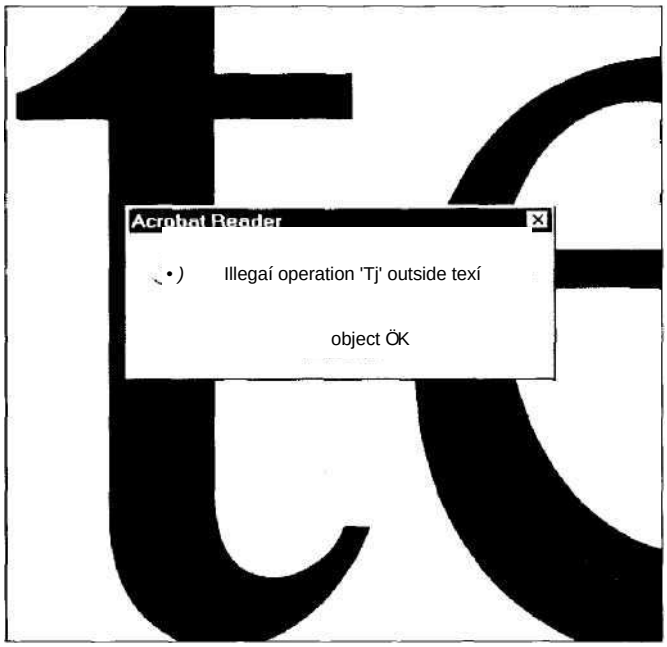
A PostScript-pont az inch 1/72-ed része, így képes kell, hogy legyél az összes mértéket inchben megadni, de bizonyos mértékek, például a betűméret, mégis pontot használnak. Az ennek a résznek az elején bemutatott define-utasítások használatával megadhatsz millimétert is. A tájolás 0 az álló és 1 a fekvő kép esetében. Használd a következő definíciókat, hogy könyebben megjegyezhetővé tedd a beállításokat:

```
define("landscape",    1);
define("portrait",     0);
```

A cpdf_save() az aktuális környezetet a memóriába menti (nem lemezre), a cpdf_restore() pedig visszatölti az elmentett környezetet. A beállításokat elmented, egy részét megváltoztatod, rajzolsz néhány ábrát, majd újra visszaállítod a beállításokat arra, ahogyan elmentetted őket. A beállításokat elmentheted egy oldal elején, az oldalt ettől eltérően alakíthatod ki, majd a következő oldalra visszatöltöd az állandó oldalbeállításokat.

Szövegfüggvények

Ha a következő függvényekkel teszel rövid dokumentumokba szöveget, bele fogsz őrülni. Ezért ha úgy döntesz, hogy a kis felbukkanó képernyőket PDF használatával hozod létre, először egy PDF-et kezelni tudó böngészőben tervezd meg őket. Találd ki a betűtípust, a méretet és minden egyebet, mielőtt PHP-val próbálnád a feladatot automatizálni. A 11.1 ábra azt a tipikus képernyőt mutatja, amelyet akkor kaptam, amikor olyan PDF-oldalakat próbáltam megjeleníteni, amelyekben egyszerű hibák vannak (például a betűméret egy kicsit nagyra van állítva).



11.1 ábra PDF-dokumentum hibás szövegbeállítással

íme néhány kifejezés azoknak, akiknek újdonság a következőkben leírt eszközökkel való képrajzolás:

- Folytonos görbe (Path) - Amikor karakterek sorozatát hozod létre, azok folytonos görbét követnek. A path-ot egy széles vonal rajzolásakor alapvonalként használod. Úgy gondolj a path-ra, mint egy út közepére felfestett középvonalra. Képzeld el, hogy először felfested a középvonalat, majd ehhez hozzáadod az utat és az azt szegélyező árkot.
- Körvonal (Outline) - Egy körvonallal körbevett karakternek vonalak vannak a külse jén, és belül vagy van kitöltés vagy nincsen. Amikor egy alakzatot, például egy vastag vonalat rajzolsz, a rajzolással outline-t hozol létre a path körül. Az út példáját folytat va, az outline kicsit olyan, mint az út mentén lévő árkok.
- Kitöltés (Fill) - A kitöltő szín az út aszfaltja, az, ami kitölti a körvonalak közötti részt. Ha kitöltő színt használsz egy egyszínű alakzat vagy karakter rajzolásakor, a kitöltő szín az egész alakzatra, így a körvonalra is vonatkozik. Ha először körvonalat használsz és utána töltőd ki az alakzatot, használhatsz más színt a kitöltéshez. így használhatsz fekete körvonalban piros belsejű karaktereket.
- Vonás (Stroke) - A körvonal egy másik neve. Ha vonalhúzással készítesz egy vonalat vagy karaktert, akkor tulajdonképpen egy körvonalat rajzolsz. Ekkor olyan dolgokat állíthatsz be, mint a vonás színe és vastagsága, így körvonalak széles választéka áll ren delkezésedre.

A ClibPDF-szövegfüggvények a következők:

- **cpdf_global_set_document_limits()** - Limitet állít be az összes dokumentumra.
- **cpdf_set_char_spacing()** - A karakterek közötti távolságot állítja be.
- **cpdf_set_word_spacing()** - A szavak közötti távolságot állítja be.
- **cpdf_set_leading()** - A szövegsorok közötti távolságot állítja be.
- cpdf_set_current_page() - Az aktuális oldalt állítja be.
- **cpdf_set_font()** - A betűtípust és a méretet állítja be.
- **cpdf_set_page_animation()** - Az egyes oldalak megjelenítése közötti időt állítja be (másodpercben).
- **cpdf_setdash()** - A szaggatott vonalak mintáját állítja be.
- **cpdf_setflat()** - A simaságot állítja be.
- **cpdf_scale()** - Egy dokumentum X- és Y-tengelyeinek léptékét állítja be.
- **cpdf_set_horiz_scaling()** - A szöveg vízszintes léptékét állítja be.
- cpdf_set_text_rise() - A következő szöveg indexelését állítja be. A pozitív szám felső, a negatív alsó indexet hoz létre.
- **cpdf_set_text_rendering()** - A szöveg renderelését állítja be, olyan lehetőségekkel, mint az egyszínű szöveget létrehozó „karakterkitöltés".
- **cpdf_set_text_matrix()** - Mátrixot állít be az aktuális szöveg transzformálására.

- **cpdf_stringwidth()** - A szöveg szélességét adja meg az aktuális betűtípusban.
- cpdf_setlinecap() - A linecap-paramétert állítja be, amely a sorok végét szögletesre vagy kerekre alakítja.
- **cpdf_setlinejoin()** - A linejoin-paramétert adja meg, amely a vonalak metszéspontjait 45°-ban illesztettre, gömbölyítettre vagy ferdén metszettre állítja.
- **cpdf_setlinewidth()** - A sorszéleséget állítja be.
- **cpdf_setmiterlimit()** - Beállítja a ferde illesztés korlátját, ami meghatározza, hogy mi lyen éles lehet egy metszéspont, amikor két vonal hegyesszögben metszi egymást.
- **cpdf_set_creator()** - A létrehozó mezőt állítja be a dokumentumban.
- cpdf_set_keywords() - Kulcsszavakat ad a dokumentumhoz.
- **cpdf_add_annotation()** - Megjegyzést ad a **dokumentumhoz.**
- **cpdf_set_subject()** - Beállítja a dokumentum tárgymezőjét.
- **cpdf_set_title()** - Beállítja a dokumentum címmezőjét.
- cpdf_begin_text() - Szövegrészt kezd el.
- **cpdf_continue_text()** - A következő sorban folytatja a szöveget.
- **cpdf_end_text()** - Lezárja a szöveget.
- **cpdf_show()** - Az aktuális pozícióban jeleníti meg a szöveget.
- **cpdf_show_xy()** - Az X- és Y-koordináták által meghatározott pozícióban jeleníti meg a szöveget. Ahhoz, hogy működjön, mindkét koordinátának a **cpdf_begin_text()-** és **cpdf_end_text()-függvények között kell lennie. Ha a cpdf_show()-t** az **cpdf_begin_text()** eló'tt használod, az Acrobat reader a következő hibaüzenetet jelzi: *Illegal operation *Tj' outside text object.*
- **cpdf_set_text_pos()** - Beállítja a szövegpozíciót a következő **cpdf_show()-ra.**
- **cpdf_text()** - A koordináták által meghatározott pozícióban jeleníti meg a szöveget.
- **cpdf_add_outline()** - Könyvjelzőt ad az aktuális oldalhoz.

Színfüggvények

Azok számára, akik jobban szeretik a színeket, itt vannak **a** színfüggvények

- **cpdf_setrgbcolor()** - A rajzolási és kitöltési színeket RGB-színekben álltja be.
- **cpdf_setrgbcolor_fill()** - A kitöltési színt RGB-színben állítja be.
- **cpdf_setrgbcolor_stroke()** - A rajzolási színt RGB-színben állítja be.

Vonalfüggvények

A vonalfüggvények a következők:

- **cpdf_translate()** - Beállítja a koordinátarendszer origóját.
- **cpdf_lineto()** - Vonalat rajzol egy abszolút ponthoz.
- **cpdf_rlineto()** - Vonalat rajzol egy relatív ponthoz.
- **cpdf_arc()** - ívet rajzol.

- **cpdf_curveto()** - Görbét rajzol.
- **cpdf_circle()** - Kört rajzol.
- **cpdf_rect()** - Négyszöget rajzol.
- **cpdf_setgray_fill()** — Beállítja vonalak vagy alakzatok rajzolásával létrehozott körvo nal kitöltési színét.
- **cpdf_setgray()** - Beállítja a rajzolási és kitöltési színeket.
- **cpdf_setgray_stroke()** - Beállítja a rajzolási színeket.
- **cpdf_rotate()** - Beállítja a fokokban meghatározott forgatást.
- **cpdf_newpath()** - Új görbét kezd.
- **cpdf_clip()** - Kivágja az aktuális görbét.
- **cpdf_closepath()** - Lezárja az aktuális görbét.
- **cpdf_fill()** - Kitölti az aktuális görbét.
- **cpdf_fill_stroke()** - Kitölti a görbét, és vonalat húz az aktuális görbe mentén.
- **cpdf_stroke()** - Vonalat húz a görbe mentén.
- **cpdf_closepath_stroke()** - Lezárja a görbét, és vonalat húz a görbe mentén.
- **cpdf_closepath_fill_stroke()** - Lezárja a görbét, vonalat húz a görbe mentén, és kitöl ti a görbét.
- **cpdf_moveto()** - Az aktuális pontot abszolút pozícióra állítja be.
- **cpd£_rmoveto()** - Az aktuális pontot relatív pozícióra állítja be.

Képfüggvények

A következő két függvény dolgozik a képekkel:

- **cpdf_import_jpeg()** - Segítségével JPEG-dokumentumot nyithatsz meg, hogy a do kumentumba importáld.
- **cpdf_place_inline_image()** - Képet helyez az oldalra.

FDF

Az FDF-opció Windows NT4 alatti telepítéséhez a következőket kell tenned:

1. Állítsd le az Apache-ot.
2. A php/extensions-ből másold a php_fdf.dll-t a c:/winnt/system32-be.
3. A php/dlls-ből másold a FdfTk.dll-t a c:/winnt/system32-be.
4. A php.ini-ben^távolítsd el a ;-t az **extension=php_fdf.dll** elől.
5. Indítsd újra az Apache-ot.

Az Adobe Forms Data Formát (FDF) teljes megértéséhez célszerű elolvasni a **http://partners.adobe.com/asn/developer/acrosdk/forms.html** oldalon **felsorolt** doku-

meritumokat. Az FDF jónéhány területet érint. Először egy olyan FDF-dokumentumot hozol létre, amely egy HTML-űrlaphoz hasonló dolgot csatol a PFF-dokumentumhoz, majd amikor az adat visszajut a szerveredre, az FDF-adatot további FDF-függvények használatával elolvashatod.

Az FDF létrehozása

Az első lépés az FDF-fel egy olyan FDF-fájl létrehozása, amit a böngészőnek el lehet küldeni. Az **fdf_create()** létrehoz egy FDF-et a memóriában, és egy azonosítót ad vissza, amelyet a többi FDF-függvény használ. Az **fdf_set_value()** egy mezőt és egy alapértelmezett értéket ad az FDF-nek. A következő példában két mező van, az egyik az árnak, a másik a mennyiségnek.

Az **fdf_set file()** egy PDF-re mutató URL-hivatkozás beszúrásával a PDF-hez kapcsolja az FDF-et. A hivatkozás azért URL, mert az Adobe Acrobat reader a böngészőben beolvassa **a** hivatkozást, majd megpróbálja a PDF-et beolvasni a böngészőből. A példában lévő teszt **URL-t** olyannal helyettesítsd, ami működik a honlapodon.

Az **fdf_save()** lemezre menti az FDF-et a későbbi felhasználásra. Ebben a példában az FDF azonnal megjelenítődik, de mivel az FDF egy vásárlói rendelési **űrlap,** esetleg meg akarod tartani és használni egy darabig. A lemezre való mentés több megjelenítési és tesztelési lehetőséget is ad. Az **fdf_close()** segítségével felszabadíthatod az FDF-re osztott memóriát (ami nem sok, hiszen a legtöbb információ a PDF-ben van):

```
$orders = "i : /orders/";
$fdf = fdf_create() ;
fdf_set_value($fdf, "price", "\$0.00", 0);
fdf_set_value($fdf, "quantity", "1", 0);
fdf_set_file($fdf, "http://test/order.pdf");
fdf_save($fdf, $orders . "order.fdf");
fdf_close($fdf) ;
```

Az FDF megjelenítése

Az FDF-et böngészőben kell megjeleníteni. Lehet linkkel kezelni, mint bármelyik fájlt, ha a webszerver be van állítva a fájltípus kezelésére. Ha nincsen, akkor a következő kóddal linkelhetsz egy oldalra. A header()-függvény egy MIMÉ típusú application/vnd.fdf-t küld, és az olyan böngészők, amelyeken az Adobe Acrobat telepítve van, fel **kell,** hogy ismerjék, hogy **a** fájl egy FDF. Az fopen() sztringként adja vissza a fájlt, és az **fpassthru()** elküldi a böngészőnek:

```
<?php
$orders = "i : /orders/";
Header("Content-type: application/vnd.fdf");
fpassthru(fopen($orders . "order.fdf", "r"));
?>
```

Az FDF-adat beolvasása

Ha a felhasználó kitölti az űrlapot és az eredményeket visszaküldi a honlapodnak, az adat egy speciális formátumban van bezárva, így az adat helyreállítása két lépésben történik. (Biztos vagyok benne, hogy valaki írni fog egy kódot, amely egy lépésben kikódolja az ada-

tokát.) Határozz meg egy könyvtárat, amely fogadja az adatokat, vagy használj ideiglenes könyvtárat. Nyiss meg egy új fájlt az fopen()-nel a w-opció beállításával, hogy írhass is a fájlba. Ha az adott állományt egy megrendelés rekordjaként akarod kezelni, akkor a fájlnév tartalmazza mondjuk a dátumot és az időt. Az **fwrite()-tal** írd a **$HTTP_FDF_DATA-**sztringet a fájlba, utána pedig zárd azt be:

```
$replies = "i:/orders/replies/"; $fdf = fopen($replies .
"areply.fdf" , "w"); fwrite($fdf, $HTTP_FDF_DATA,
strlen($HTTP_FDF_DATA)); fclose($fdf);
```

Most már van egy FDF-fájlod, amelyet beolvashatsz, hogy a megrendelési űrlap válaszait megkapd. Nyisd meg az fopen()-nel beolvasásra a fájlt, olvasd be a mezőértéket a fdf_get_value()-vel, majd az fdf_close()-zal zárd be a fájlt:

```
$fdf = fdf_open($replies . "areply.fdf",
"r"); $price = fdf_get_value($fdf, "price");
$quantity = fdf_get_value($fdf, "quantity");
fdf_close($fdf);
```

Elméletben FDF-dokumentumban egy ciklussal mezők hosszú során tudsz végigfutni a következő mintakód használatával. Az **fdf_next_field_name()** az első mező nevét adja, ha a fdf_next_field_name() második paramétere üres. A **fdfnextfieldnameQ** következő használata a következő mezőt adja, ha második paraméternek az előző mezőt állítod be. Hogy mi történik, ha a függvény túlmegy az utolsó mezőn, az meghatározatlan, így a jövőben ez változhat:

```
$field = fdf_next_field_name($replies) ; $values[$field]
= fdf_get_value($fdf, $field); while($field =
fdf_next_field_name($replies, $field))
    {
    $values[$field] = fdf_get_value($fdf, $field);
```

Ha újra be kell hívnod az FDF-ben fdf_set_file()-lal beállított PDF nevét, az fdf_get_file()-lal tudod azt visszakeresni. Az egyik lehetséges használat, amikor sok PDF-hez egy FDF-et csatolsz, és azonosítani akarod az FDF-fel használt PDF-et. Azon az oldalon, amely megkapja az FDF-től a választ, az FDF neve egy utaló sztringben lesz a PGF helyett, így vagy PDF-enként egy FDF-et hozz létre, vagy használd az **fdf_get_file()-t.**

A következő függvényekkel az Adobe-dokumentációban leírt opciókat lehet beállítani:

- **fdf_set_ap()** - Egy mező megjelenését állítja be.
- fdf_set_flags() - Jelzőket állít be egy mezőhöz.
- fdf_set_opt() - Opciókat állít be egy mezőhöz.
- fdf_set_status() - A /STATUS-kulcs értékét állítja be az FDF-dokumentumban (a részleteket az Adobe-dokumentációban keresd).
- fdf_get_status() - A /STATUS-kulcs értékét adja meg.
- **fdf_set_submit_form_action()** - A HTML-űrlap Elküld gombjához hasonló műveleti

mezőt állít be (az Adobe egyszerűbbé tehette volna az életet, ha csak lemásolta volna **a** HTML tag-paramétereket).

- fdf_setjavascript_action() - JavaScript-műveletet csatolhatsz egy mezőhöz, hasonlóan ahhoz, ahogy egy HTML tag-hez tennéd. A JavaScriptet például az input-adat érvényesítésére használhatod.

PDFlib

A PDFlib-opció NT4 vagy Windows 2000 alatti telepítéséhez a következőket kell tenned:

1. Állítsd le az Apache-ot.
2. A php/extensions-ből másold a php_pdf.dll-t a c:/winnt/system32-be.
3. A php.ini-ben távolítsd el a ;-t az **extension=php_pdf.dll** elől.
4. Indítsd újra az Apache-ot.

Megjegyzés: A Windows más verzióiban a c:/winnt/system32 helyett használd a c:\windows\system-et.

A PDF-függvényeket, melyek hasonlóak a cpdf_-függvényekhez, a „Szöveg létrehozása PDF-dokumentumban PDFlib-bel" részben mutatom meg.

Flash és Shockwave

A Shockwave-fájlformátum (SWF) leírása az OpenSWF oldalán **(www.openswf.org)** talál-ható. A PHP SWF-függvények Paul Haeberli libswf-moduljához **(http://reality.sgi.com/grafica/flash)** csatlakoznak. Ezekkel a függvényekkel hozhatsz létre olyan fájlokat, amelyek a Macromedia Flash- (www.macromedia.com/software/flash-) böngészőbővítőkkel olvashatóak. A Macromediának két böngészőbővítménye van, a Flash (www.macromedia.com/software/flash) és a Shockwave **(www.macrornedia.com/software/shockwaveplayer). (A** Shockwave-player a Flash-playert is tartalmazza.) A Flash-fájlokat a Macromedia Flash-évél, a Shockwave-fájlokat a Macromedia Directorral lehet létrehozni. A libswf-könyvtár a Flash-fájltípust alkalmazza, a PHP-függvények pedig ezt a nevet vették fel. Az SWF-függvényekkel azonban összetettebb Shockwave-fájlokat nem hozhatsz létre.

Mivel a Flash a legtöbb böngészőn telepítve van, és a Flash-sel nincs annyi gond, mind a PDF-fájlokkal, a Flash használhatóbb módja az egylapos brosúrák megjelenítésének, mint a gyakrabban használt PDF. A Flash jobb interaktivitással rendelkezik a hosszú dokumentumokban való navigálásra, de jelen pillanatban egyik jelentősebb keresőmotor sem indexeli a Flash-fájlokon belüli szöveget, így a PDF előnyben van ezen a területen.

Rengeteg nálam jobb Flash-művész van, így nincsen értelme annak, hogy bonyolult Flash-ekkel próbáljalak lenyűgözni. Amire szükséged van az swf_-függvények, a **libswf** és a Unix, mivel a könyvtár jelenleg még nem érhető el Windows vagy Windows NT alatt.

A fájl

Kezdjünk egy Flash-fájl swf_openfile()-lal való megnyitásával. Az swf_openfile()-t kell az első swf-függvényként használni, különben a könyvtár hibaüzenetet jelenít meg. A függvényben nincsen a fájlazonosítónak megfelelő valami, így nem nyithatsz meg egyszerre több fájlt.

A következő swf_openfile()-példa lemezen nyit meg egy fájlt:

```
$flash["file"]     =    realpath("./veryflash.swf" ) ;
$flash["width"]    =   400;
$flash["height"] = 500;
$flash["framerate"] = 20;                                                    \r
$flash["background"]["red"] = 20;
$flash["background"]["green"] = 20;
$flash["background"]["blue"] = 20;
swf^openfile($flash["file"] , Sflash["width"], $flash["height"],
   $flash["framerate"], $flash["background"]["red"],
   $flash["background"]["green"], $flash["background"]["blue"]);
```

A fájl beállítási kódja a realpathQ-függvényt használja, amely egy olyan típusú relatív fájlnevet fogad el, amilyet egy weboldalon használsz, és ezt a PHP-függvényekben használt abszolút névre konvertálja. (Egy weboldalon a fájlnevek a honlap címéhez viszonyítottan relatívak, de a PHP az operációs rendszer szerverének báziscíméhez viszonyítottan relatív fájlneveket használ.)

Ha az előző példában használt swf_openfile()-t a következő példában találhatóval cseréled ki, akkor az SWF-fájl a php://stdout speciális fájlnév használatával a képernyőn nyílik meg:

```
swf_openfile("php://stdout",     $flash["width"],
   $flash["height"], $flash["framerate"],
   $flash["background"]["red"], $flash["background"]["green"],
   $flash["background"]["blue"]);
```

Ha végeztél egy Flash-fájllal, akkor mielőtt újat szeretnél létrehozni, az swf_closefile()-lal be kell zárnod a meglévőt. A következő példa egyszerűen ezt csinálja, hogy erőforrásokat szabadítson fel:

```
swf_closefile() ;
```

Ha az *előző* példában használt swf_closefile()-t a következő példában találhatóval cseréled ki, akkor az opcionális 1 megadja a függvénynek, hogy a Flash-fájlt sztringként adja vissza, így a fájlt elmentheted egy adatbázisban, vagy létrehozhatsz e-mail csatolt állományt:

```
$flash["data"]     =    swf_closefile(1) ;
```

Képkocka

A fájlon belül 0-tól számozott és tetszőlegesen címkézett képkockákkal dolgozol. A következő példában látható swf_labelframe()-mel nevezd el a képkockát. Ha már van címke egy képkockán, az swf_actiongotolabel()-lel megoldhatod, hogy a Flash-prezentáció elmenjen arra a címkére. Az swf_actiongotolabel() megjeleníti az adott képkockát és megáll:

```
swf_labelfrarae("First    frame");
swf_actiongotolabel("First    frame")
```

Az swf_showframe() bezárja az aktuális képkockát, elküldi megjeleníteni, és ezután új képkockát nyit meg. A következő műveleteket az új képkockán végzi el. Az **swf_setframe()** egy szám által meghatározott képkockára ugrik, és az utána következő műveletek már erre a képkockára vonatkoznak. Ha nem tudod az aktuális képkocka számát, az **swf_getframe()** pontosan ezt adja eredményül:

```
swf showframeO
swf_setframe(5)
$flash["frame"]
                =    swf_getframe();
```

Szín

Ha bármilyen színes elemet raksz egy képkockára, az **swf_addcolor()** és az **swf_mulcolor()** használatával állíthatod be a színeket. A színeket három, 0 és 255 közötti számként tárolja a gép. Az első szám a piros, a második a zöld, a harmadik pedig a kék értékét adja meg. A számok együtt alkotják az RGB-színeket. A 0,0,0 a fekete. Ezt HTML-ben #000000-ként kell írni. A 255,255,255 a fehér, HTML-ben #FFFFFF (hexadecimális számként). A színeket az egyedi számokon végzett aritmetikai műveletekkel változtathatod.

Az **swf_addcolor()** segítségével hozzáadhatsz (kivonhatsz) a színek értékéhez (értékéből), így a képet világosíthatod (sötétítheted). Az **swf_mulcolor()** segítéségével szorozhatod (oszthatod) a színértékeket, így növelve (csökkentve) a kontrasztot. Mindkettő működik külön a piros, a zöld és a kék színeken is, így olyan módosításokat is megcsinálhatsz, mint a kép szépia-tónusokra való konvertálása. Mindkét függvény tartalmazza az alfa-módosítást, így beleveheted az átlátszóságot is. Mindkét függvény megváltoztatja az **swf_placeobject(), swf_modifyobject()** és **swf_addbuttonrecord()** által előállított színeket:

```
swf_addcolor(3,     -1,
14); swf  mulcolor(22,        60)
22,     11,
```

Objektumok

Készíthetsz képobjektumokat, és az objektumokat képkockába helyezheted. Az objektumokat 1-től 65535-ig menő számokkal azonosíthatod, és különböző mélységű layerekbe (layer: egymást átfedő, külön szerkeszthető képelem) helyezheted, amelyek szintén 1-től 65535-ig számozódnak. Az **swf_placeobject()** egy objektumot rak - példánkban az 5 számú - layerbe. A következő szabad objektumazonosító számot az **swf_nextid()-del** találod meg, és ezzel elkerülheted, hogy meglévő objektumot tegyél tönkre. Az swf_removeobject()-tel törölhetsz objektumokat, a példa az 5 mélységben lévőt törli:

```
$frame["object"]     =
swf_nextid() swf_placeobj              5)
ect($frame["obj ect" ;
swf_removeobject(5);
```

Az **swf_modifyobject()** a meghatározott mélységben lévő objektum pozícióját vagy színét változtatja meg (ebből adódik, hogy az egyes objektumokat célszerű külön layerbe tenni,

hogy egymástól függetlenül megváltoztathasd őket. Az első paraméter a mélység, a második paraméter pedig a meghatározott módosítás: lehet **MOD_COLOR** a szín aktuális mulcolorral és addcolorral való módosítására, lehet **MOD_MATRIX,** mely a aktuális mátrix használatával megváltoztatja az objektum pozícióját, illetve lehet mindkettő, **MOD MÁTRIX | MOD_COLOR:**

```
swf_modifyobject(5,     MOD_COLOR);
```

Az objektumok olyan elemek, mint a sorok vagy az **swf_defineline()-, swf_definepoly()**-és **swf_definerect()**-függvényekkel létrehozott egyszerű alakzatok. Az **swf_defineline()** az xl,yl (példánkban 1,10) és x2,y2 (példánkban 101,110) között hoz létre vonalat. A vonal vastagságát meghatározhatod (a példában ez 3):

```
$line    =
swf_nextid(); swf          1,    10,    101,    110,
defineline($line,          3)
```

Az **swf_definepoly()** x,y párok tömbjét fogadja el, amely a használt pontokat, illetve a vonalvastagságot tartalmazza. A következő példa a **$points** nevű tömböt használja, amelyet a Gyors megoldások „Diagram létrehozása GIF-, JPEG- vagy PNG-képekben" részében történetesen nyíl rajzolására használok. A példa 7 pontot és 2-es vonalvastagságot alkalmaz:

```
$polygon    =    swf_nextid();
$points    =    array(200,    200,    270,    140,    270,    170,    380,    170,    380,    230,
    270,    230,    270    ,260);
swf_definepoly($polygon,    $points,    7,
2);
```

Az **swf_definerect()** téglalapot rajzol az xl,yl és x2,y2 pontokra a megadott vonalvastagságban. A példa ugyanazokat a koordinátákat használja, mint a vonalas példa, de a téglalap két szemközi sarkaként tekinti a pontokat. Akár az **swf_definepoly(),** akár az **swf_definerect()** esetében a vonalvastagságot 0.0-ra állítva, a körvonal kitöltésével a függvények egy egyszínű objektumot hoznak létre:

```
$rectangle    =    swf_nextid0;
swf_definerect($rectangle,    1,    10,    101,    110,    3);
```

A shape-parancsokkal összetett alakzatokat hozhatsz létre, ahogy azt a következő kódban láthatod. Az alakzat az swf_startshape{)-pel kezdődik és az **swf_endshape()-pel** végződik. Az **swf_shapelinesolid()** a vonal színét (50,50,50), az alfa csatornát (0) é$ a vastagságot (1) állítja be. Ha a vastagságot 0.0-ra állítod, nem fog vonalakat rajzolni. Az **swf_shapefillsolid()** beállítja az alakzatkitöltést, és beállítja a kitöltő színt (75,75,75) és az alfa-csatornát (0). Az **swf_shapefilloff()** kikapcsolja az alakzatkitöltést:

```
$shape = swf_nextid() ;
swf^startshape($shape) ;
swf_shapelinesolid(50, 50,   50,    0,      1);
swf_shapefillsolid(75, 75,   75,    0)
swf_shapefilloff() ; swf
endshape();
```

Az **swfshapefillbitmapclipO** bittérkép használatára állítja a kitöltést. Az **swf_shapefillbit-maptile()** szintén bittérképpel tölti ki az alakzatot, de ezt csempeszerűen teszi:

Ir


```
swf_shapefillbitmapclip($bitmap);
swf_shapefillbitmaptile($bitmap);
```

Az swf_shapemoveto() az aktuális pozíciót az x,y-pozícióba viszi. Az **swf_shapelineto()** vonalat húz az aktuális pozícióból az x,y-pozícióba (ez példánkban 100,100), majd az aktuális pozícióból az x,y-ba teszi az aktuális pozíciót. Az **swf_shapecurveto()** egy másodfokú Bézier-görbét rajzol az aktuális pozícióból az első x,y-ponton (100,100) keresztül a második x,y-pontba (250,250), majd az aktuális pozíciót az utóbbira állítja be. Az **swf_shapecurveto3()** egy harmadfokú Bézier-görbét rajzol az aktuális pozícióból az első (300,300) és második (350,350) x,y-ponton keresztül, majd az aktuális pozíciót az utóbbira állítja be. Az **swf_shapearc()** ívet rajzol az első' két paramétert a kör középpontját mutató x- és y-koordinátaként használva, a harmadik paraméter a sugár, a negyedik a kiindulási szög, az utolsó paraméter pedig a végső szög:

```
swf shapemoveto(25, 40); swf_shapelineto(100, 100); swf_shapecurveto(100,
100, 250, 250); swf_shapecurveto3(300, 300, 350, 350, 340, 340);
swf_shapearc(200, 200, 50, 30, 40);
```

Az **swf_viewport()-tal** a teljes oldal egy téglalap alakú alhalmazába helyezheted a rajzodat. A paraméterek sorrendje különbözik a normál téglalap függvényekbeli sorrendtől: az első paraméter az x minimuma, a második az x maximuma, a harmadik az y minimuma, a negyedik pedig az y maximuma:

```
swf_viewport(100, 200, 100, 300);
```

Az **swf_ortho()** a felhasználói koordináták aktuális látványba való háromdimenziós ortografikus leképezését határozza meg. Az x minimumát, az x maximumát, az y minimumát, az y maximumát, a z minimumát és a z maximumát kéri a függvény:

```
swf_ortho(100, 200, 100, 300, 100, 200);
```

Az **swf_ortho2()** a felhasználói koordináták aktuális látványba való kétdimenziós ortografikus leképezését határozza meg. Az x minimumát, az x maximumát, az y minimumát és az y maximumát kén a függvény:

```
swf_ortho2(100, 200, 100, 300);
```

Az swf_perspective() a látvány perspektivikus kivetítését transzformálja. Az első paraméter a képmező szög az y irányban, a második a látvány oldalaránya, a harmadik a közeli kivágás síkja, a negyedik pedig a távoli kivágásé. Az **swf_polarview()** polár koordinátákkal határozza meg a látvány nézőpontját. Az **swf_lookat()** nézőpont-transzformációt hajt végre, segítségével például szöveget forgathatsz el. Az swf_scale() az aktuális transzformációt méretezi az x-, y- és z-tengelyeken, az **swf_translate()** mindhárom tengelyen végez egy másik transzformációt, az swf_rotate() pedig mindhárom tengely körül elforgatja az aktuális transzformációt.

Az **swf_posround()** az objektumok elhelyezésénél vagy mozgatásánál használt kerekítést változtatja meg. A kerekítés olykor segíti a képtisztaságot, de nem mindig. **f^™„„™„„**

Az **swf_pushmatrix()** verembe teszi az aktuális transzformáció mátrixot, így elmentheted azt későbbi felhasználásra. Az **swf_popmatrix()** visszaállítja a verembó'l a korábbi transzformációs mátrixot.

Szöveg

A szövegírást a betűtípus definiálásával kell kezdened. Ehhez bármilyen PostScript-betűtípust használhatsz. A példában használt **swf_definefont()** a Times-Roman PostScript-betű használatával egy talpas betűtípust határoz meg. Az swf_setfont() a meghatározott típusúra állítja az aktuális betűtípust, az **swf_fontsize()** az aktuális betűméretet határozza meg, az **swf_fontslant()** az aktuális betű dó'lését (ami lehet pozitív vagy negatív). Az **swf_fonttracking()** az aktuális betűtípus karakterei között távolságot állítja be úgy, hogy a pozitív szám növeli, a negatív pedig csökkenti a távolságot. Az **swf_getfontinfo()** egy asszociatív tömböt ad eredményül, amely tartalmazza az **Aheight-et,** a nagybetűs A magasságát pixelben, és a xheight-et, a kisbetűs x magasságát pixelben. Az **swf_definetext()** egy a meghatározott szöveget tartalmazó objektumot hoz létre, és opcionális harmadik paramétert fogad el a szöveg középre rendezésére (ahogy a példában használtam). A szövegelhelyezés megtervezését segíti a **swf_textwidth(),** amely az adott szöveg aktuális betűtípusban és betűméretben vett szélességét adja vissza:

```
$font_serif = 1;
swf_definefont($font_serif, "Times-Roman");
swf_setfont($font_serif);
swf_fontsize(30);
swf_fontslant(-5);
swf fonttracking(2);
$font__size = swf_getfontinfo();
$text = swf_nextid();
swf_definetext($text, "This text is Flash", 1)
$width = swf_textwidth("test string");
```

A betűtípusfüggvény 1 típusú PostScript-fontot igényel. A legtöbb embernek True Type fontjai vannak, így lehet, hogy neked is konvertálni kell a fontjaidat. Van egy nyílt forráskódú átalakító a http://ttf2ptl.sourceforge.net-oldalon, illetve számtalan nem ingyenest találhatsz mind Windows-ra, mind Macintosh-ra.

Bittérképek és szimbólumok

Bizonyos függvényekben bittérképeket használsz, amelyeket az **swf_definebitmap()-pel** hozhatsz létre. Inputként JPEG-, GIF- vagy RGB-képeket használhatsz, amelyek Flash fájlformátumra lesznek konvertálva. A **swf_getbitmapinfo()** egy tömböt ad eredményül, mely a bittérkép méretét bájtban (size), a szélességét pixelben **(width)** és a magasságát **pixelben (heíght)** tartalmazza:

```
$image = swf_nextid();
swfdefinebitmap($image, realpath ("./test.jpeg"));
$image info = swf getbitmapinfo($image);
```

A szimbólumok olyan apró Flash mozgóképek, amelyeket nagyobb Flash mozgóképben használsz objektumként. Ha a General Motors weboldalára készítesz Flash-animációt, kis animált szimbólumokat használhatsz az egyes autómárkákra, majd a szimbólumokat teljes képkockán rakod össze. Egy szimbólum definiálását a **swf_startsymbol()-lal** kezded és az swf_endsymbol()-lal fejezed be, közöttük pedig Flash-függvényeket használsz a szimbólum meghatározására:

```
$symbol = swf nextid();
swf startsymbol($symbol);
swf_endsymbol();
```

Műveletek

A műveletekkel a képkockák a böngészőben való megjelenését szabályozhatod. A műveleteket szkriptként rögzíted a Flash-fájlban, így szúrva be oda őket. Flash-terminológiában a képkocka lejátszása a böngészőbe teszi a képkockát, a mozgókép lejátszása pedig folytonos képkocka-megjelenítési módba teszi a képkockákat.

Az **swf_actionnextframe()** a fájl következő képkockájára ugrik, a **swf_actionprevframe()** pedig az előzőre, így szerkesztheted, megjelenítheted vagy lejátszhatod a képkockát. **Az swf_actiongotoframe()** a meghatározott képkockára ugrik, és megjeleníti azt (példánkban a 4. képkockát). Az **swf_actionplay()** az aktuális képkockától játssza le a Flash mozgóképet, az **swf_actionstop()** pedig az aktuális képkockánál állítja meg.

Hogyan teszel műveletet egy képkockára, és hogyan hajtasz végre egy műveletet? A példában a kód a 4. képkockára ugrik, a 4. képkockára helyezi az **swf_actionstop()-ot,** majd visszaugrik az 1. képkockára, és lejátssza a mozgóképet. Ha azt **akarod,** hogy az **swf_actionstop()** a képkockára kerüljön ahelyett, hogy azonnal végrehajtódjon, tedd a függvényt az **swf_startdoaction()** és **swf_enddoaction()** közé, ahogyan a példában is van:

```
swf_actionnextframe();
swf_actionprevframe();
swf_actiongotoframe(4);
swf_startdoaction(); swf
actionstop ();
swf_enddoaction();
swf_actiongotoframe(1);
swf_actionplay();
```

Keverheted az URL-információt Flash képkockákkal és mozgóképekkel. Ragadd meg az URL-t az **swf_actiongeturl()-lel,** ahogy azt a következő kódban látod, és jelenítsd meg a mozgókép előtt a **_Ievell**-paraméter használatával. A **_levelO** az aktuális mozgóképet az URL-lel helyettesíti:

```
swf_actiongeturl("http://test.com/test.html",      "_levell");
```

A Flash reader a képkockákat kétféle minőségben képes megjelentetni, az **swf_actiontog-glequality()-vel** lehet a jobb és rosszabb minőség között váltani, A következő kód ellenőrzi, hogy a minőséget megváltoztatták-e, és ha nem, akkor megváltoztatja:

```
if(!isset($flash["quality"]) or !$flash["quality"]) {
    swf_actiontogglequality() ;
    $flash["quality"] = true;
```

Az swf_actionwaitforframe() ellenőrzi, hogy a képkocka (példánkban az 5. képkocka) be van-e töltve, és ha nem, akkor átugorja a második paraméterben meghatározott számú (ebben az esetben 3) műveletet. **Ezt** használhatod üzenet megjelenítésére, miközben a képkockák még töltődnek. A logikáját fordítottnak találtam: ha nincs még kész a képkocka, jobb lenne visszafele ugrani a szkriptben. Remélem, a jövőben lesz ilyen fejlesztés, éppúgy, ahogy a műveleteken is lesz címke, hogy egyszerűbb legyen a megfelelő műveletre való ugrás:

```
swf actionwaitforframe(5, 3);
```

Az **sw£_actionsettarget()-tel** műveletek célját állíthatod be, így több, egyszerre játszott Flash mozgóképet irányíthatsz. Eddig még nem láttam ennek a kódnak igazán jó példáját, vagy olyan weboldalt, amely több mozgóképet játszana. Az igazat megvallva, a több mozgóképet olyan figyelmet zavaró dolognak tartom, amely elűzi a vásárlókat, és olyan unott tizenéveseket vonz az oldalra, akik úgysem vásárolnak semmit.

Gombok

A gombok lehetnek **TYPE_MENUBUTTON** vagy **TYPE_PUSHBUTTON** típusúak, ahol a **TYPE_MENUBUTTON** elengedi a fókuszt a gombról, amikor az egér nincs rajta, a **TYPE_PUSHBUTTON** pedig még ilyenkor is a gombon tartja a fókuszt. A gomb kialakítása a **swf_startbutton()-nal** kezdődik. Az **swf_addbuttonrecord()** az első paraméterrel információt ad a gomb használatáról a gomb állapotainak megadásával (ami lehet egy vagy mind a **BSHitTest, BSDown, BSOver** és **BSUp** közül), egy objektumazonosítóval és a gomb aktuális frame-ben értelmezett mélységével. Különböző **swf_oncondition()-** és műveleti függvényekkel meghatározod a gomb műveleteit, végül pedig az **swf_endbutton()**-paranccsal befejezed a gomb definiálását:

```
$button = swf_nextid();
swf_startbutton($button, TYPE^PUSHBUTTON)
swf_addbuttonrecord(BSDown, Sshape, 20);
swf_oncondition(ButtonEnter) ;
swf^actiongeturl("http://test.com/test.html ", "_levell");
swf^oncondition(ButtonExit) ;
swf^actiongeturl("", " leveli");
swf_endbutton();
```

Az swf_oncondition()-függvény a gomb típusától függően két lista egyikéből fogad konstansokat. A konstansok a **HTML mouseover-** és **mouseout-jához** hasonló műveleteket jelentenek. A további részleteket keresd a dokumentációban a **www.openswf.org-oldalon.** A **TYPE_MENUBUTTON** a következőket fogadja:

- **IdletoOverUp**
- **OverUptoIdle**

- **OverUptoOverDown**
- **OverDowntoOverUp**
- **IdletoOverDown**
- **OutDowntoIdle**
- **MenuEnter (IdletoOverUp | IdletoOverDown)**
- **MenuExit (OverUptoIdle | OverDowntoIdle)**

A TYPE_PUSHBUTTON a következőket fogadja:

- **IdletoOverUp**
- **OverUptoIdle**
- **OverUptoOverDown**
- **OverDowntoOverUp**
- **OverDowntoOutDown**
- **OutDowntoOverDown**
- **OutDowntoIdle /**
- **ButtonEnter (IdletoOverUp | OutDowntoOverDown)**
- **ButtonExit (OverUptoIdle | OverDowntoOutDown)**

Képadatbázisok

Ha egy dokumentum-formátumban, például PDF-ben vagy Microsoft Wordben szövegben tárolsz képet, mérlegeld annak lehetőségét, hogy a szöveget és a képeket párhuzamosan tárolod egy adatbázisban, vagy a szöveget egy kereshető adatbázisban tárolod, a képeket pedig egy külön könyvtárban, a szövegben a képekre mutató linkekkel. Egy jól megtervezett adatbázis alapú megközelítés jobb keresési opciókat kínál, mint a PDF-dokumentumok egy halmaza, és egyszerűbb, mint dokumentumok sorozatához egy adatbázist indexként illeszteni.

Ha egy üzleti diagramot képi összetevőkből építesz fel, ezek az összetevők lehetnek arányiak. Ekkor is gond nélkül tárolhatod a diagram szerkezetét és az összes képi összetevőt egy adatbázisban. Ebben a pillanatban a folyamatábrák és hálózati diagramok jutnak eszembe, ahol a képelemek apró ikonok, amelyeket különböző szövegcímkével számtalanszor újra felhasználnak. Az ilyen diagramok adatbázis alapú megjelenítése sokkal kisebb lehet, mint maga a grafikus megjelenítés, és a számítógépnek sem kerül sok idejébe az egyes diagramok megrajzolása.

MySQL

A kezdők számára a MySQL a legegyszerűbb adatbázis, amelynek gyors az olvasáshoz való hozzáférése, így nagyon megfelel az olyan dokumentumrendszerekre, ahol a legtöbb hozzáférés weboldalról érkezik. A MySQL nem rendelkezik hosszú bináris mezőkkel a képek

tárolására, így jobban jársz, ha külső fájlként tartod meg a képeket, és az adatbázisban csak a képekre vonatkozó hivatkozásokat tárolod.

Egyéb SQL-adatbázisok

Az olyan egyéb adatbázisok, mint a PostgreSQL, Oracle és SQL Server, hivatkozási integritást és egyéb értékes tulajdonságokat nyújtanak az üzleti alkalmazásoknak, például a képértékesítőknek. Ha online árulsz képet, és a számlázási szoftvered Oracle-t használ, használj a képi adatbázishoz is Oracle-t. A hivatkozási integritásból és a többi hasznos szolgáltatásból következő magas többletráfordítások miatt itt is inkább külső fájlként tárold a képeket, ne pedig belső bináris objektumként. Ha valaki ragaszkodik az adatbázisból való képazonosításhoz, tárold a képhosszúságot és a CRC-t az adatbázisban. A CRC-t a 8. fejezetben magyarázom el.

Hyperwave

A Hyperwave egy a dokumentumok kereshető objektumként való tárolására kialakított adatbázis. A dokumentumnak bármilyen képformátumot tudnia kell tartalmazni, mert a kereshető referenciák a dokumentumokkal párhuzamosan vannak tárolva. A Hyperwave-et a 6. fejezetben mutatom be részletesen.

A képi modul telepítése

Unix-rendszerben a letöltések listája megtalálható a telepítési utasítások között. A lista tartalmazza a képi modul és a különböző extrák, például a JPEG tömörítő rutinok forrását. Egy jó Linux-disztribúcióban az összes rész RPM-ként benne van, de kevesen telepítik alapként a PHP-t, és senkit sem ismerek, aki a PHP-telepítési listában az összes opciót kiválasztaná.

Windows és Windows NT alatt a képi modul letöltésben érhető el. Csupán annyit kell tenned, hogy a php/extensions-ből Windows NT és Windows 2000 alatt a c:/winnt/system32-be, a Windows más verziói alatt windows/system-be másolod a php_gd.dll-t. Mialatt másolsz, másold le a php_exif.dll-t is, így használhatod a read_exif_data()-t. Ezután a php.íni következő sorai elől távolítsd el a pontosvesszőt (;): "•

```
extension=php_gd.dll
extension=php_exif.dll
```

Ha a PHP-t modulként futtatod, újra kell indítanod a webszervert, hogy újra beolvassa a php.ini-t, és ekkor már használd ki az alkalmat a PDF-modul telepítésére.

Képek megjelenítése

Ha egy oldalon belül képet jelenítesz meg, vannak olyan trükkök, amelyekkel az egész profinak íog tűnni, és vannak olyan PHP-függvények, amelyek segítségével egyszerűen tökéletessé teheted az oldalad. A getimagesize()-zal információt szerezhetsz a képfájlokról.

A függvény a fájl nevét kéri az elérési úttal együtt, eredményként pedig - ha a fájl GIF, JPEG, PNG vagy SWF típusú képfájl - egy a fájlról információkat tartalmazó tömböt ad. Bármilyen fájlhivatkozást, így (PHP 4.0.5-től felfelé) URL-eket is használhatsz, a getimagesize()-ba pedig képet tartalmazó sztringet is betehetsz - ami akkor fordulhat elő', ha nem fájlt használsz a képekre, például ha egy e-mailhez csatolt állományból olvasod be a képet, és így mented el adatbázisba.

A getimagesize()-nak van egy opcionális második paramétere, amellyel további infromációkat tudhatsz meg a JPEG-képekről. Ezt az extra paramétert bizonyos dokumentációkban „hívásidőátadás hivatkozással" ("call time pass by reference") paraméterként jelzik, mint a getimagesize($file, &$extra)-ban. Amikor kipróbáltam ezt PHP 4.0.5-ben, az alább látható hibaüzenetet kaptam. Ezután kipróbáltam a kódot & nélkül, mint a getimagesize($file, $extra)-ban, és tökéletesen működött, ami azt jelzi, hogy a getimagesize() a hivatkozás által átadott paramétert belsőleg azonosítja:

Figyelem: Ha hivatkozással kívánod átadni a paramétert, módosítsd a *getimagesizeQ deklarációját. Ha engedélyezni kívánod a hivatkozással történő hívásidő átadást, állítsd át az allow_caU_time_pass_reference értékét true-ra (igaz) az INI fájlban. Későbbi verziók azonban nem feltételenül fogják ezt támogatni.*

Mindig használd az Alt-ot az image tag-ben

A vakok mindig a szövegre támaszkodnak, amikor a weboldaladat látogatják, és számukra az image-tag alt-paramétere a legjobb iránytű a képekhez. Ha az oldaladon van egy kép Hillary Rodham Clintonról, az alt="Mrs. Clinton" alt-paraméter hozzáadása kicsiny segítség, míg ha valami olyasmit írsz, hogy alt="2000. január 20., Hillary Rodham Clinton lányával a Fehér Ház lépcsőin áll.", az nagyon hasznos lehet.

A JPEG-fájlok leírásait tárolhatod adatbázisban, és az 5. és 6. fejezetben leírt függvényekkel férsz hozzá ezekhez. Ugyanakkor tárolhatod a jPEG-fájlban a leírásokat szöveges mezőben is, és getimagesize()-zal keresheted vissza őket. A getimagesizeQ egy extrás nevű mezőt tartalmazó tömböt ad eredményül, ami általában üres, hiszen nagyon kevés képszerkesztő engedi meg, hogy ténylegesen szöveget írj a mezőbe. Ha van egy olyan képszerkesztőd, amelyikkel szöveget rakhatsz az extras-mezőbe, tedd be az alt-szöveget, és egy szerzői jogi figyelmeztetést.

A GNU Image Manipulation Program (Gimp) segítségével, amely a gimp.org címen érhető el (Unix, Windows és Windows NT alá), megváltoztathatod a JPEG-megjegyzése-ket, ha másként mented el a fájlt (File, Savé As). Megemlítettem néhány további mezőt is a fejlesztőknek, így elképzelhető, hogy a Gimp későbbi verziói megjelenítik és törlik az EXIF- és egyéb mezőket is.

Használd a méretinformációt az image tag-ben

Ha megadod a **height = "" és width** = ""-paramétereket az **<img>** tag-ekben, a látogatód böngészője az egyes képeknek megfelelő téglalapot fog megjeleníteni, majd beszúrja a képeket, ha azok letöltődtek. Ha kihagyod ezeket a paramétereket, a böngésző kis ikont helyez oda, ahol a képnek kellene megjelenniük; úgy formázza az oldalt, ahogy az a kis ikonnal megfelelő, majd az oldal elemei ugrálva igazodnak újra, ha a kép letöltődik. Sokkal színvonalasabb az oldalad megjelenése, ha használod a **height-et** és a **width-et.**

Vázlatképek létrehozása

Amikor egy kép olyan nagy, hogy egy átlagos képernyőn görgetni kell, hogy az egész lát-szódjon, jobban jársz, ha a nagy képeket kis vázlatokkal helyettesíted, és az olvasó a vázlatra kattintva kapja meg a teljes méretű képet.

Képek létrehozása

Következzenek az image-függvények és használatuk összefoglalása. Miután ezt elolvastad, rágd át magad a „Szöveg létrehozása GIF-, JPEG- vagy PNG-képekben" és „Ábra létrehozása GIF-, JPEG- vagy PNG-képekben" Gyors megoldások részeken, hogy megérezd, milyen könnyű ezekkel a függvényekkel egyszerű képeket létrehozni:

- imagearcQ - Egy ellipszis mentén rajzol egy vonalat. Ha két paraméter, a **height és width** egyenlő, akkor egy körívet kapsz.
- **imagefilledarcQ** - Ugyanazt az ívet rajzolja, mint az **imagearc(),** majd egy további paraméter alapján eldöntheted, hogy a körcikket kitöltve egy kördiagram egy részét hozod létre, vagy az ív elejét és végét a húrral összekötve a húr és az ív által határolt területet tököd ki.
- **imageellipse()** - Ellipszist rajzol, illetve amennyiben a magasság és a szélesség meg egyezik, akkor kört.
- **imagefilledelh'pse()** - Ellipszist rajzol, és a meghatározott színnel kitölti. A imageellipse()-hez hasonlóan az **imagefilledellipseQ** is kört rajzol, ha az ellipszis magassága és szélessége megegyezik.
- **imagechar()** - Vízszintesen ír egy karaktert.
- **imagecharupQ** - Függőlegesen ír egy karaktert.
- **imagestringO** - Vízszintesen ír egy karakterekből álló sztringet.
- imagestringupO - Függőlegesen ír egy karakterekből álló sztringet.
- **imageAlphaBlending()** - 24 bites, alfa-csatorna információt tartalmazó kép (PNG) színkeverési módját állíthatod be vele. A beállítás a rákövetkező, két képpel dolgozó függvényeket is érinti. A függvény új a **PHP** 4.0.6-ban.
- **imagecolorallocate()** - Színt rendel egy képhez. Az első hozzárendelt színt a háttérre alkalmazza. Ha más színűre akarod állítani a hátteret, az **imagefill()-lel** teheted meg.
- **imagecolordeallocate()** - Törli a szín-hozzárendelést, így újra hozzárendelheted vala mihez a színt.

imagecolorat() - Segítségével megkapod az adott pixelben levő szín színkódját, így használhatod a színt máshol is. Amikor a képet színskála helyett valódi színekből készíted **(PHP** 4.0.6), az **imagecolorat()** a színt adja vissza.

imagecolorclosestQ - RGB-színkódot fogad el, és a legközelebb álló szín színkódját adja vissza (nem szükségszerű, hogy az valódi színből legyen).

imagecolorexact() - RGB-színt fogad el, és az adott szín kódját adja vissza, vagy -1-et, ha nem találja a színt. Logikus lenne, ha az ilyen színekre nullát vagy hibát adna vissza, mivel a nullát a PHP hamisnak értelmezi. De a nulla szabályos színkód, így egyelőre be kell érni a -1-gyel. Talán egy későbbi verzióban átírják ezeket a függvényeket, hogy valódi hibát adjanak vissza.

imagecolorcIosestalphaQ - Egy színt és egy alfa-értéket fogad el, és a legközelebbi szín színkódját adja vissza.

imagecolorexactalpha() - Egy színt és egy alfa-értéket fogad el, és a szín színkódját adja vissza, illetve a -1-et, ha a szín nincsen a színskálában.

imagecolorresolve() - Egy színt fogad el, és a szín vagy a legközelebbi változatának színkódját adja vissza. Úgy tűnik, hogy az **imagecolorresolve()** ugyanazt a műveletet végzi el, mint az **imagecolorclosest(),** de másmilyen kiválasztási módszert használ.

imagecolorresolvealpha() - Ugyanúgy működik, mint az **imagecolorresolve(),** egy alfa-paraméterrel kiegészítve.

imagegammacorrect() - Segítségével gamma-korrekciót hajthatsz végre egy képen, így a Macintosh-ról vett képet kijavítva megjelenítheted más gépeken vagy fordítva. A függvénynek a bemenő kép gammájára, illetve a kimenő képre alkalmazott gammára van szüksége. Az Apple RGB-meghatározása 1,8-es gamma-beállítást tartalmaz, míg a PC-ké 2,2-est. Ha olyan weboldalt készítesz, amelyik különböző képeket jelenít meg PC-n és Macintosh-on, akkor célszerű a képeket 1,0-s gammával beszkennelni, majd a szkriptet úgy megírni, hogy 2,2-del másolja őket egy könyvtárba (amelyet nevezz images-nek), és 1,8-del egy másik könyvtárba (ezt pedig nevezd imagesmac-nek). Ez az egyszerű átváltás kevesebb erőforrást használ, illetve kevesebb színbeli torzulást okoz, mintha oda-vissza konvertálnál a formátumok között. Ekkor ugyanis legrosszabb esetben egy Macintosh-ról származó JPEG-et átkonvertálnál PC-s gammára, elmentenéd az images-könyvtárba, majd menet közben visszakonvertálnád Macintosh gammára. Ez két konverziót és sokkal több hibát jelentene.

imagecolorset() - Színt állít be a színskálán.

imagecolortransparentQ - Átlátszónak állíthatsz be egy színt. A GIF támogatja az átlátszóságot, a PNG-képekben azonban azt jobban szabályozhatod az alfa-értékekkel.

imagecolorsforindexQ - Egy a piros, zöld és kék összetevőket tartalmazó tömb formájában ad vissza egy színt a színskálából.

imagecolorstotal() — A színskála színeinek számát adja vissza, ezzel eldöntheted, hogyan mentsd el a képet, illetve azt is, hogy pixelenként hány bit legyen. Jelenleg az imagegifQ and imagepng() függvényekben nincsen olyan opció, hogy adott pixelméretben mentse el a képet.

imagecopyO - Egy kép téglalap alakú részét a kép egy másik részébe másolja.

imagecopymerge() - Ugyanazt csinálja, mint az imagecopy(), egy extra lehetőséggel: a másolat az általad meghatározott mértékben olvad bele a képbe. Az összeolvadás 0-tól 100-ig terjed. 0-nál gyakorlatilag semmi nem történik, 100-nál pedig teljes a helyettesítés.

imagecopymergegrayO - Ugyanazt csinálja, mint az impagecopymerge(), egy extra lehetőséggel: a célterületet először szürke árnyalatúra konvertálja, így az összes szín a másolt képből jön. A függvény jelenleg még fejlesztés alatt áll, így nem mutatok rá példákat.

imagecopyresized() és imagecopyresampled{) - Ezeket a függvényeket az „Átmére-tezés és újbóli mintavételezés" című részben tárgyalom.

imagecreateQ - Színskála alapú képet hoz létre (8 bites szín).

imagecreatetruecolor() - Valódi színekből hoz létre képet (24 bites).

imagetruecolortopaletteQ -True color (24 bites) képet palettaképre konvertál (8 bites színmélység), ennek akkor veheted hasznát, ha egy hirdetőd JPEG-ben küldi a bannerjét, amelyben egyszerű sima színeket használ. Ahelyett, hogy betömörítenéd a bannert, és ezzel kockáztatnád a színhelyességet, próbáld a bannert PNG-be (vagy GIF-be) másolni.

imagecreatefromgif() - GIF-fájlból vagy URL-ből hoz létre képeket. Jóllehet a szerzői jogi helyzet nem teszi lehetővé, hogy a PHP szerzői GIF létrehozására alkalmas kódot tegyenek a programba, az ilyen fájlok beolvasása nem korlátozott, így könnyedén írhatsz szkriptet GIF beolvasására, amelyet utána PNG-ként menthetsz.

imagecreatefromjpegO - JPEG-fájlból vagy URL-ből hoz létre képeket.

imagecreatefrompngQ - PNG-fájlból vagy URL-ből hoz létre képeket.

imagecreatefromwbmpO -Windows bitmap-fájlból vagy URL-ből hoz létre képeket.

imagegifQ - Elküldi a képet a böngészőnek, vagy GIF-formátumban fájlba menti.

imagepngQ - Elküldi a képet a böngészőnek, vagy PNG-formátumban fájlba menti.

imagejpegO - Elküldi a képet a böngészőnek, vagy JPEG-formátumban fájlba menti.

imagewbmpO - Elküldi a képet a böngészőnek, vagy Windows bitmap-formátumban fájlba menti.

imagedestroyO - Eltávolítja a képet a memóriából, amikor végeztél vele.

imagecreatefromstringQ - Képet hoz létre egy sztringben tárolt képből, így beolvashatsz adatbázisokba mentett vagy e-mailek csatolt állományaiból dekódolt képeket. Új függvény, amely még fejlesztés alatt áll, így nem próbáltam ki, hogy milyen képformátumokkal működik.

imagesetpixel() - Egyszerű pontot írhatsz vele. imagelineQ - Vonalat rajzol az egyik x,y-pontból a másikba. imagedashedline() - Szaggatott vonalat rajzol az egyik x,y-pontból a másikba.

- **imagerectangle()** -Téglalapot rajzol két sarokpont közé.
- **imagefilledrectangle()** - Téglalapot rajzol, és kitölti a kiválasztott színnel.
- **imagepolygon()** - Sokszöget rajzol a tömbben meghatározott pontok listájából.
- **imagefilledpolygon()** - Sokszöget rajzol, és kitölti a kiválasztott színnel. A téglala pokat, sokszögeket és ellipsziseket a Gyors megoldások „Diagram létrehozása GIF-, JPEG- vagy PNG-képekben" című részben mutatom meg.
- imagesetbrushQ - Ez a függvény fejlesztés alatt van, vonalak rajzolásához választ hatsz vele ecsetképet. A kép a memóriában levő bármelyik kép lehet, és mivel pszeudoszínként van definiálva, a függvény színparaméterével választhatod ki vonal vagy sokszög **rajzolására.**
- **imagesetthickness()** - Szintén egy új függvény, amellyel a vonalrajzoló függvények hez választhatod ki a vonal vastagságát.
- **imagepsloadfont()** - PostScript-betűtípust tölt be egy fájlból.
- **imagepsbbox()** - PostScript Type 1 betűtípussal készült, téglalappal határolt szöveg méreteit adja vissza.
- **imagepsencodefont()** - PostScript-betű karakterkódolási vektorát változtatja meg.
- **imagepsfreefont()** - A PostScript betű által használt memóriát szabadítja fel, amire akkor lehet szükséged, amikor rengeteg betűtípust egyszerre használsz a szkriptben.
- **imagepsextendfont()** - Kiterjeszt vagy tömörít egy PostScript-betűtípust.
- **imagepsslantfontQ** - Dőlést ad egy PostScript-betűtípusnak.
- **imagepstext()** - PostScript-betűtípust használva szöveges sztringet ír egy képre.
- **imageloadfont()** - Egy speciális formátum használatával bitmap-betűtípust tölt be fájlból.
- **imagefontheight()** - Egy betűtípus magasságát adja vissza.
- **imagefontwidth()** - Egy betűtípus szélességét adja vissza.
- **imagettftext()** - TrueType-betűtípussal ír szöveget.
- **imagettfbbox() - Az imagettftext()-tel** írt szöveget határoló téglalap méretét adja vissza.
- **imagefill()** - Az egész képet egy színnel tölti ki.
- **imagefilltoborder()** — Meghatározott színnel tölt ki egy képet a szín által meghatáro zott keretig.
- **imagesettile()** - Ez is új függvény, akár a **imagesetbrush().** Segítségével pszeudó ki töltési színt adhatsz meg a képnek, és a képet használhatod alakzatok kitöltésére. **Az imagefilltoborder()-hez** hasonló függvényekben való kitöltéshez egyszerűen válaszd a **IMG_COLOR_TILED** színt.
- **imageinterlaceQ** - Megmutatja, hogy az interlace-opció be van-e kapcsolva a képen, és segítségével be- vagy kikapcsolhatod azt. Bizonyos képek esetén az interlace lehetővé teszi a kép teljes letöltése előtt annak nagy vonalakban történő megjeleníté sét. Mivel sok kép esetében ennek nincs értelme, előfordul, hogy a függvénynek nin. haszna. Bizonyos fájlformátumoknál az interlacing megnövelheti a fájlméretet, ezer; használat előtt teszteld.

imagesx() - A kép szélességét adja vissza.

imagesyO - A kép magasságát adja vissza. Az imagesx()- és imagesy()-függvények-kel számításokat végezhetsz a a kép mérete alapján. Ez az olyan függvényekkel betöltött képek esetén nagyszerű, mint az imagecreatefromjpeg().

imagetypesQ - A PHP által támogatott képtípusok listáját adja vissza. Segítségével dinamikus kódot írhatsz, amely GIF-eket használ a PHP régebbi verzióival működő oldalakon, és PNG-t használ ott, ahol ezt a PHP újabb verziói támogatják.

Képek változtatása

A képek változtatása magában foglalja annak veszélyét, hogy a gondosan elkészített grafikai munkából zavaróan rossz kép válik. A PHP 4.0.6 elő'tti verzióiban a képfüggvények 8 bites módban működtek, így lehetett velük GIF- és 8 bites PNG-képeken dolgozni, de JPEG-eken nem. A PHP 4.0.6 által bevezetett imagecreatetruecolor()-függvénnyel 24 bites színekkel dolgozhatsz PNG- és JPEG-képeken.

Színek változtatása

A GIF 8 bites színt (256 különböző' szín), a JPEG 24 bitest (24 millió különböző szín), a PNG pedig mindkettőt használhatja. Amikor a PNG 8 bites színt használ, mind a GIF, mind a PNG olyan színskálát használ, ahol mind a 256 színnek lehet 24 bites értéke. Ez azt jelenti, hogy ügyfeled kérhet meghatározott RGB-szím, például #f2334c-t, te pedig bármilyen fájlformátumban használhatod a színt. De vajon a szín pontosan ugyanaz marad?

A GIF és a PNG pontos színeket tárol, így az ügyfelednek a #f2334c fog megjelenni. A JPEG viszont egy közelítést tárol (amely egyre pontatlanabb lesz, ahogy növeled a tömörítést), így a szín nem lesz pontos, de a tömörítés alacsony foka mellett ez nem észrevehető. Ha használod a imagejpeg()-t, és a opcionális minőségi paramétert 100-nál kisebbre állítod (az alapértelmezett érték a 100), veszítesz a színhelyességből. Az olyan képek esetében, mint a céglogók, mindig mutasd meg a megbízónak az eredményt.

A Macintosh-nak és a PC-nek különbözik a gamma-beállítása, így a PC-n készített képek túlságosan világosnak tűnnek a Macintosh-on, a Macintosh-on készített képek pedig túlságosan sötétek PC-n. A LCD-monitorok a 24 milliónál sokkal kevesebb színt képesek megjeleníteni, így a színek kevésbé pontosnak tűnhetnek. Mindezek a technikai problémák nehezítik, hogy az ügyfelek minden alkalommal a megfelelő színt lássák. Hogyan tudsz hát ezen segíteni?

Mindig eredeti képből dolgozz. Ha folyamatosan változtatsz egy képet, különösen ha egy JPEG-et, tartsd meg az eredeti fájlt és a változtatások listáját. Majd az új képet úgy hozd létre, hogy az eredeti képen hajtsd végre az összegzett változtatásokat. Hozz létre egy images nevű könyvtára a weboldaladon, illetve az oldaladon kívül egy originals-nak nevezett másikat. Az images-hez bárki hozzáférhet, de az originals-hoz nem, míg a PHP mindkettőhöz hozzáfér. Ha változtatást akarsz végrehajtani, például megnyitod Kanada térképét, városneveket és hőmérsékleteket írsz rá, nyisd meg az eredetit és írj rá mindent. Csábító lenne megnyitni az eredetit, ráírni a városneveket, elmenteni, és utána már csak a

hőmérsékleteket kellene naponta ráírni, de JPEG esetében a kép kétszeri változtatása bármilyen szintű tömörítés mellett megnöveli a kép torzulásait.

A másik lehetséges megközelítés, ha az eredeti képeket PNG-kém tárolod, elvégzed a frissítést, például a városnevek hozzáadását, majd PNG-ként elmented. A napi frissítéseknél használhatod a imagecreateirompng()-t a PNG-képek originals-ból való megnyitására, hozzáadod a hőmérsékleteket, majd a imagejpeg()-gel JPEG-et hozol létre az images-be.

Átméretezés és újbóli mintavételezés

Képet átméretezni nem nehéz. Ami nehéz lehet, az az optimális kép előállítása. Az éles sarkú egyszerű geometriai alakzatok esetében a sarkok egyenetlenné válhatnak, a fénykép színei pedig elmosódnak. Hogyan kaphatsz tökéletes képet?

A pixeles átméretezés az éles vízszintes és függőleges élek esetében működik, ahol az új méret az eredetinek páros töredéke, például pontosan a fele. Az egyéb technikák az átlós vonalakat és színárnyalatokat részesítik előnyben. A 11.2 ábra átméretezés előtt mutatja a szöveget. A 11.3 ábra azután mutatja a szöveget, hogy azt harmadára kicsinyítettük, majd az eredeti méretűre nagyítottuk. Könnyű észrevenni a 11.2 ábra tiszta, éles, egyenes vonalai és a 11.3 ábra egyenetlen görbéi közti különbséget. A PHP tartalmazza a képek átmérete-zésére való imagecopyresized()-képfüggvényt.

A 11.4 ábra a bilineáns újbóli mintavételezéssel, a 11.5 pedig a hatodfokú (bicubic) ismételt mintavételezéssel átméretezett szöveget mutatja. A hatodfokú újramintavételezés az egyenetlen élek eltüntetésére való kísérletével elmosódottá teszi a képet, de ez engem mindig csupán arra késztet, hogy meg akarjam törölni az olvasószemüvegem. Amikor ezt írom, a PHP 4.0.6 fejlesztés alatt áll, és tartalmazza az új imagecopyresampled()-függvényt, amely a 11.4 és 11.5 ábrákhoz hasonló átméretezést tesz majd lehetővé.

1

resize

11.2 ábra Szöveg átméretezés előtt

resize

11.3 ábra Szöveg pixelátméretezés után

11.4 ábra Szöveg bilineáris átméretezés után

resize

11.5 ábra Szöveg hatodfokú átméretezés után

Külső programok

Segédprogramok használatával tömeges változtatásokat hajthatsz végre képeken. A külső programok futtatását a 7. fejezetben mutatom be. A batch-programokkal konvertálhatsz képeket, hogy minden kép ugyanakkora legyen vagy ugyanolyan tömörítési szinten legyen. Ha egy programmal, például a Gimp-pel kötegelt konverziót hajtatsz végre, meg kell elégedned egy átlagos eredménnyel, vagy pedig az egyedi konverziók végrehajtásához használd a Gimp szkriptnyelvét, vagy képenként hívd be a Gimp-et, hogy végezze el az adott képhez legjobban illő változtatást. Én jobban szeretem az összes szkriptet PHP-n belül tartani, és a külső programokat képenként egyszer behívni.

Az image-függvények valódi színeszköztárra alakítanak át, hogy a PNG-formátumot teljesen ki lehessen használni. Az újabb 3D-s képformátumok valószínűleg nagyon népszerűek lesznek, ahogy a széles sávú Internet-hozzáférés terjedésével a 3D-s virtuális bemutatótermek használhatóvá válnak. Sok programnyelvvel kapcsolatos tapasztalataim alapján kijelenthetem, hogy ha egy új technológia támogatása megjelenik egy szkriptnyelvben, akkor az első ilyen nyelv a PHP.

A megfelelő formátum kiválasztása

Most már eleget tudsz ahhoz, hogy egy alkalmazáshoz a megfelelő fájlformátumot válaszd, illetve a szóba jöhető formátumokat összehasonlítsd. A Flash támogatása Unixra korlátozott, így bizonyos honlapok esetén ez gyors döntést jelenthet. A böngésző támogatása befolyásolhat bizonyos választásokat, éppúgy, ahogy a választott betűtípusok is.

A kódolás egyszerűsége szintén döntő tényező lehet, bár ilyen szempontból nincs nagy különbség a formátumok között, kivéve, ha valami nagyon Összetett dologgal akarsz megbirkózni. Amint ez történik, valószínűleg egyetlen formátumra leszel korlátozva.

n

Gyors megoldások

Képek listázása

Ez a megoldás a 8. fejezetben kifejlesztett megoldások egyszerűsített változata, amelyben a képinformáció a teljes fájlmegjelenítés részeként íródik ki. A kód megmutatja, hogyan futhatsz végig a fájlokon, kilistázva a képek fájlattribútumait.

A getimagesize() egy fájlnevet és egy további, opcionális paramétert tartalmazhat, amelyet itt $extra-nak nevezek. Eredményként tömböt ad, a példában ezt $att-nak nevezem, amely a képre vonatkozó információkat tartalmaz. A $att-ba négy hagyományos, illetve JPEG-ké-pek esetén további két érték kerül, de a vizsgált képek között egyikről sem volt további információ, és a két hagyományos érték közül csak kettő', a magasság és szélesség az, amelyik hasznos. így a kód az összes értéket eltárolja, de csak a magasságot és szélességet jeleníti meg. A $extra-ba egy a képben tárolt információkat tartalmazó tömb kerül - ez a fajta információ jelenleg csak a JPEG-képekről érhető el, és a vizsgált fájlok között nem volt JPEG. Jóllehet a kód összegyűjti és egy megjelenítésre alkalmas sztringbe helyezi az információt, a sztringet nem helyeztem a kimeneti HTML-táblázatba.

Képinformációk gyűjtése

A következő függvény a 8. fejezetben bemutatott fájlattribútumok beolvasására kifejlesztett függvény módosított változata. A függvény egy könyvtár elérési útját fogadja el, és egy a könyvtárban levő képekről információt tartalmazó tömböt ad vissza. A kód első része eltávolítja az elérési út végén levő visszaperjelet, majd megnyitja az elérési út által mutatott könyvtárat. A kód ciklussal végigfut a fájlokon, .jpg, .jpeg, .gif, .png vagy .swf kiterjesztésűeket keresve, majd összegyűjti a fájlokból a getimagesize()-zal megkapott információt. Az információt a tömbbe rakja, és a függvény végén visszaadja:

```
function get_directory_file($path) {
    if(substr(Spath, -1) == "/")
        í
        $path — substr($path, 0, -1); }
    $path_id = opendir($path);
    while($file_name = readdir($path_id) ) {
        if ($file_name != "." and $file_name != "..") {
            $file["type"] = @filetype($path . "/" . $file_name); if
            ($file["type"] == "dir") { $file_array = get directory file($path.
            "/" - $file name);
```

```
            print("<br>found type: " . gettype($found) ); if
            (isset($found))

               $found = array_merge($found, $file array);

            else

               $found = $file_array; }
        else
            {
            $file["size"] = filesize($path . "/" . $file_name);
            $x5 = strtolower(substr($file_name, -5));
            3x4 = substr($x5, -4);
            if($x4 == ".jpg" or $x5 == ".jpeg" or $x4 == ".gif"
            or $x4 == ".png" or Sx4 == ".swf")
               {
               $att = getimagesize($path . "/" . $file_name,
                  $file["extra"] ) ;
               if (isset($att[0] )) {$file["width"] = $att[O];}
               if(isset($att[l])) {$file["height"] = $att[l];}
               $found[$path][$file_name] = $file;



closedir($path_id);
if ( !isset($found))
    {
    $found    =    array();
    }
return($found);
```

Hivatkozás:	oldal:
Fájlok listázása az attribútumokkal együtt	**263**

Képinformáció kiíratása

A következő kód a 8. fejezetben bemutatott táblázatformázó függvényeket használja, és egy külön table.html nevű fájlba mentettük el:

```
include("./table.html");
```

A következő kód a get_directory_file()-ból megkapja a képek tömbjét, majd ciklussal végigfut rajta, megszerzi onnan az információt és táblázatba formázza:

```
$found    =    get    directory_file(Spath,
""); reset( $ f ound);
```

```
$print_ümit = 10;
print("<table border=\"3\">" . tr("<td colspan=\"7\">"
    . "<font color=\"Green\"Xem>Directory</ern></font></td>") ) ;
while (üst ($d, $dv) = each ($f ound) )

    if(is_array($dv))

        print(tr("<td colspan=\"7\">"
            . "<font color=\"Green\"xem>" .
            $d . "</emx/fontx/td>")          tde("Height") . tde("Width")
        . tr(tde("File") . tde("Size") .     tde("Extra")));
        tde("Channel")  . tde("Bits") while
        (üst ($f, $fv) = each(Sdv))

            if(is array(?fv))

                if ($print_ümit)

                    if (!isset ($fv["channel"])) {$fv["channel"] = "Snbsp;";}
                    if(!isset($fv["bits"])) {$fv["bits"] = "Snbsp;";}
                    if(!isset($fv["extra"])) {$fv["extra"] = "Snbsp;";}
                    if(is_array($fv["extra"]))

                        $e = "";
                        reset($fv["extra"] ) ;
                        while (üst ($ek, $ev) = each ($fv [ "extra" ]))

                            if(strlen($e))

                                $e .= "<br>";

                            if($ek == "APP13")

                                $e .= $ek . ": " . iptcparse($ev);

                            else
                                {
                                $e .= "k:      $ek .     v:     . $ev;
                                }
                            }
                        }
                    else
                        {
                        $e = $fv["extra"] ;
                        } print(tr(tdl($f) .
                    tdl($fv["size"])
                        . tdl($fv["height"]) . tdl($fv["width"])
                        . tdl($fv["channel"])  . tdl($fv["bits"])
                        . tdl($e)));
                    $print_ümit--;
```

A iptcparse()-függvény dekódolja az International Press Telecommunications Council (IPTC) szabványának (www.iptc.org) megfelelően formázott információkat.

A 11.6 ábra az előző kód eredményét mutatja, és igazolja, hogy a legtöbb képben nincsenek csatornák, bitek vagy extra értékek. Körülbelül 40 000 képet néztem végig az Interneten, és csupán néhánynál találtam megemlítve az Adobe image editort, mint a képen utoljára használt képszerkesztőt.

Hivatkozás: **oldal:**

Képfájlok attribútumainak kiíratása **275**

\Directory					
^:/petennoulding/web/root/phpblackbooMmages/images					
\File	*\]Size*	*\Heigkt*	*\Width*	*\Ckannei*	*\Bits \Extra*
bs.gif	Í255	:22	61	j	
kingparrotjpg	í60810	,500	**[250**	ij_	
me.jpg	**|6530**	**|1OO**	**[80**		
notme.jpg	150024	250	**280**	|	
PHPSydney.com.	**gif|6405**	**jlÖÖ**			

11.6 ábra Képek listája attribútumokkal

A 11.7 ábra ugyanazt a **get_directory_füe()-lal** kapott tömböt mutatja, képattribútumok helyett a képeket megjelenítve.

Előfordul, hogy a képeket szép rendben kell a vásárlóknak vagy látogatóknak megmutatnod. A következő kód azt mutatja meg, hogy jelenítsd meg a getimagesize()-hoz hasonló függvényekkel gyűjtött információk felhasználásával helyesen a képeket.

A kód ugyanazon a tömbön fut egy ciklussal végig és ugyanolyan táblázatot formáz, de a képeket a sorokban jeleníti meg. Az **iptcparse()-t** kihagytam, mert itt nincs értelme. A képet egy **<img>** tag-gel helyeztem be. A képméret-információ mondja meg a böngészőnek, hogy mekkora terület kell a képnek. Az alt-információ a képfájl neve, illetve ha olyan képszerkesztőt használsz, amellyel szöveget tehetsz a JPEG-szövegmezőbe, akkor az alt-mezőbe rakhatod az ott levő szöveget. Ennek alternatívája az alt-szöveg kikeresése az adatbázisból. Az 5. és 6. fejezetben mutatom be az ehhez használható adatbázisokat:

```
$print_ümit = 10;
print("<table border=\"3\">" . tr("<td colspan=\"7\">"
    . "<font color=\"Green\"><em>Directory</em></font></td>")
while (üst ($d, $dv) = each ($found) )
    {
    if(is_array($dv))

        print(tr("<td colspan=\"7\">"
        . "<font color=\"Green\"xem>" . $d . "</em></fontx/td>")
        tr (tdeC'File") . tde("Size") . tde("Height") .
        tde("Channel") . tde("Bits") while tde("Width") tde("Extra")));
        (üst ($f, $fv) = each($dv)){
            if(is_array($fv)) { if
                ($print_ümit)

                    if (üsset ($fv["channel"] )) { $fv [ "channel" ] =
                    "Snbsp;";} if (üsset ($fv["bits"] )) {$fv["bits"] =
                    " ";} if (üsset ($fv["extra"] )) { $fv [ "extra" ] =
                    "Snbsp;";} if(is_array($fv["extra"]))

                        $e = "";
                        reset ($fv["extra"]);

                        while (üst ($ek, $ev) = each ($fv [ "extra" ] ) )

                            if(strlen($e))

                                $e .= "<br>";

                            if($ek == "APP13")

                                $e .= iptcparse($ev);
                                $ek
                                }
                            else

                                $e .= $ek v: $ev;
                                "k: }
                            }
                        }
                    else
                        {
                        $e = $fv["extra"];
                        } print(tr(tdl($f) .
                    tdl($fv["size"])
                        . tdl($fv["height"]) . tdl($fv["width"])
                         . tdl($fv["channel"]) . tdl($fv["bits"])
                        .
                    tdl($e))); $print
                    limit--;
```

A iptcparseQ-függvény dekódolja az International Press Telecommunications Council (IPTC) szabványának (www.iptc.org) megfelelően formázott információkat.

A 11.6 ábra az előző kód eredményét mutatja, és igazolja, hogy a legtöbb képben nincsenek csatornák, bitek vagy extra értékek. Körülbelül 40 000 képet néztem végig az Interneten, és csupán néhánynál találtam megemlítve az Adobe image editort, mint a képen utoljára használt képszerkesztőt.

Hivatkozás: **oldal:**
Képfájlok attribútumainak kiíratása **275**

\Directory				
ji *y/petermoulchng/wsb/root/phpblackbook/images/images*				
\File	í|&ze			
■bs.gif	255 :22	61 [		
kingparrotjpg	60810 500	250 |		
me.jpg	6530 100	|80 |		
notme.jpg		280 j		
50024 250		i?o i ... _		

11.6 ábra Képek listája attribútumokkal

A 11.7 ábra ugyanazt a get_directory_file()-lal kapott tömböt mutatja, képattribútumok helyett a képeket megjelenítve.

Előfordul, hogy a képeket szép rendben kell a vásárlóknak vagy látogatóknak megmutatnod. A következő kód azt mutatja meg, hogy jelenítsd meg a getimagesize()-hoz hasonló függvényekkel gyűjtött információk felhasználásával helyesen a képeket.

A kód ugyanazon a tömbön fut egy ciklussal végig és ugyanolyan táblázatot formáz, de a képeket a sorokban jeleníti meg. Az iptcparse()-t kihagytam, mert itt nincs értelme. A képet egy <img> tag-gel helyeztem be. A képméret-információ mondja meg a böngészőnek, hogy mekkora terület kell a képnek. Az alt-információ a képfájl neve, illetve ha olyan képszerkesztőt használsz, amellyel szöveget tehetsz a JPEG-szövegmezőbe, akkor az alt-mezőbe rakhatod az ott levő szöveget. Ennek alternatívája az alt-szöveg kikeresése az adatbázisból. Az 5. és 6. fejezetben mutatom be az ehhez használható adatbázisokat:

			Directory
			i:/petermDiűding/yi'eb/root/phpbiackbook/i}nages/images
File		*Heigh*	*Widthllmage*
bs.gif		**22**	**01 ■- Backspace**
kingparrot.jpg	60810	**500**	**250 *m* 1^SVP\ 9| :"J^^K^ ^^^B^ ^B^f: is.3T ^^^^^^S"^ W m.^WP' J**
- * .	**6530**	**100**	

11.7 ábra Képlista teljes méretű képekkel

```
$found = get directory_file($path, "");
reset($found);
$print limit = 10;
print("<table border=\"3\">" . tr("<td colspan=\"5\">"
    . "<font color=\"Green\"Xem>Directory</emx/fontx/td>") ) ;
whiledist ($d, $dv) = each ($found) )

    if(is^array($dv))

        print(tr("<td colspan=\"5\">"
            . "<font color=\"Green\"><em>"    $d . "</em></fontx/td>")
            . tr{tde("File") . tde("Size") .    tde("Height")  . tde("Width")
            tde("Image")));
        while (üst ($f, $fv) = each($dv)) {
            if(is_array($fv))
                {
                if($print_limit)
```

```
{
print(tr(tdl($f) . tdl($fv["size"])
. tdl($fv["height"]) . tdl($fv["width"]) .
tdl("<img src=\"./images/" . $f . "\"" . "
width=\"" . $fv["width"]  . "\"" . " height=\""
. $fv["height"] . "\"" . " border=\"0\" alt=\""
    . $f . "\">"))),
$print_limit--;
```

print ("</table>") ;

A 11.8 ábra az egy lépéssel továbbvitt képlistát mutatja, ahol minden kép korlátozott méretben jelenik meg. Mivel egyes képek jóval nagyobbak, mint mások, a nagy képek méretét maximalizálva még lehet őket olvasni, és oldalanként több képet is meg lehet jeleníteni. Figyeld meg, hogy ezeket a képeket nem átméretezem, csak a megjelenítést korlátozom. Ez tökéletes megoldás például Intranet-alkalmazások esetében.

Directory				
ij/peterffiouíding/web/root/phpblackboaMntages/muges,				
Fiié	*Size*	*Height*	*Width*	*Image*
bs.gif	**255**	**22**	**61**	Bockspcce i
kingparrot.jpg	60810	500	250	
me.jpg	6530	100	80	
Dotme.jpg	**50024**	250	280	El
PHPSydncy.com.gif	**6405**	**100**	**190**	

11.8 ábra Képlista előnézeti méretű képekkel

A következő kód hozzáad néhány dolgot az előző kódhoz. A **$image_limit** a képek maximális magasságát és szélességét határozza meg. A közepén levő kód kiválasztja a legna-

gyobb méretű képet, elosztja a méretét a $image_limit-tel, majd a kapott arányt használva arányosan osztja le a méreteket. A print-utasítás két sorát megváltoztattam, hogy az eredeti méret helyett a kiszámított méretet használják:

```
$found = get_directory_file($path, "");
reset($found);
$print limit = 10;
$image__limit = 100;
print("<table border=\"3\">" . tr("<td colspan=\"5\">"
    . "<font color=\"Green\"><em>Directory</em></fontx/td>") ) ;
while (üst ($d, $dv) = each ($found) )
    i
    if(is_array($dv))

        print(tr("<td    colspan=\"5\">"
            .    "<font                          $d    .    "</em></fontx/td>")
            color=\"Green\"Xem>" .               tde("Height")    .
            tr(tde("File")    .    tde("Size")   tde("Width")
            .    tde("Image")));
        while (üst ($f,    $fv)    =
        each($dv))

            if(is_array($fv))

                if ($print_ümit)
                    {
                    $h = $fv["height"] ; $w =
                    $fv["width"] ; if($w >
                    $image limit)

                        $div   =   $w   /   $image   limit;
                        $h   =   $h   /   $div;
                        $w   =   $w   /   $div;
                        }print(tr(tdl($f)     .
                    tdl($fv["size"])
                        .   tdl($fv["height"])    .   tdl($fv["width"]
                        .   tdl("<img   src=\"./images/"    .    $f    .    "\"
                        .   "   width=\""   .   $w   .   "\""
                        .   "   height=\""   .   $h   .   "\""
                        .   "   border=\"0\"   alt=\""   .   $f   .
                    "\">")) $print_limit--;



print("</table>"    );
```

Ha sok fájlt szeretnél megmutatni az Interneten, fontold meg a képek másolását és átmére-tezését, amit a fejezet más részeiben bemutatott image-függvények használatával végezhetsz el. A felesleges ráfordítások csökkentése végett legyen egy **/image/**-könyvtárad, ahol az eredeti képeket tárolod, és egy **/imagesmall/,** ahol a gyorsnézeti változatokat. A képek adminisztrációja során gyújtsd ki egy tömbbe az /image/-ben levő képeket, és hasonlítsd

össze az /imagesmall/-lal. Ha egy fájl megvan az /image/-ben, de nincsen az /imagesmall/-ban, akkor a megfelelő image függvényekkeL olvasd be a. fájlt, méretezd át, majd mentsd el ugyanazon a néven az /imagesmall/-ban is.

Az előző adminisztrációs képmegjelenítést úgy is módosíthatod, hogy az kiszámolja az új méreteket, meghívja az image-függvényeket, hogy azok lemásolják, átméretezzék és elmentsék az eredményt, végül pedig meg is jelenítsék azt a gyorsnézeti böngésző listában.

Szöveg létrehozása PDF-dokumentumban ClibPDF-fel

Egy weboldal szkriptjéből akkor érdemes PDF-fájlt készíteni, ha olyan egyedi betűtípussal akarsz szöveget megjeleníteni, amely normál böngészőben nem elérhetők. A következő kód azt mutatja meg, hogyan készíts a hagyományos Times New Román betűtípus használatával szöveget, amelyhez utána bármilyen betűtípust és szöveget hozzáadhatsz. A font neve bármilyen érvényes PostScript-betűtípus lehet, és a betű méretét - függetlenül attól, hogy milyen mértékegységet használsz a dokumentumban - pontokban kell meghatároznod:

```
define ("portrait", 0);
$doc_name = "clibpdftest.pdf";
$doc_path = dirname($PATH_TRANSLATED)  . "/";
$font["face"] = "Times-Roman";
$font["size"] = 40;
if($doc = cpdf_open(0))
    {
    print("<br>Opened." );
    cpdf_page_init($doc,  1,  portrait,  288,  360);
    cpdf_begin_text($doc) ;
    cpdf_set_font($doc,  $font["face"],  $font["size"],  "NULL");
    cpdf_show($doc,  "test  text");
    cpdf_show_xy($doc,  "test  at  xy",  1,  1);
    cpdf_end_text($doc) ;
    cpdf_finalize($doc) ;
    if(cpdf_save_to_file($doc,  $doc_path  .  $doc_name))
        { print("<br><a  href=\"./"  .  $doc_name  .
        "\»>"
            .  $doc_name  .}
    cpdf_close(\$doc) ;
```

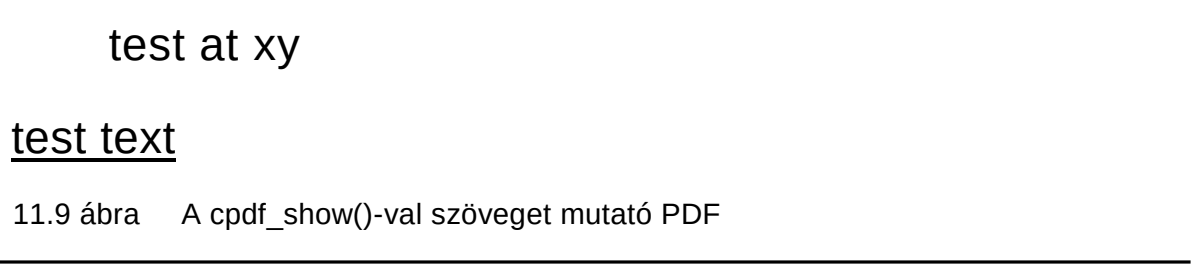
A kód egy PDF-fájlt hoz létre az aktuális könyvtárban, majd linkkel mutat a fájlra. Ez segítségedre válik a tesztelésnél, hiszen a példákat elmentheted hivatkozásul. Ezt a megközelítést olyan adminisztratív vezérlőpanel esetében is használhatod, amellyel új oldalakat hozol létre honlapodon, vagy a frissített adatokkal újra létrehozod az oldalakat.

A cpdf_open() dokumentumot nyit meg, és a $doc-dokumentumazonosítót adja vissza, amely minden más ClibPDF-függvény első paramétere. A cpdf_page_init() az oldal tájolását és a méretet (pontokban) állítja be. A cpdf_begin_text() szövegblokkot kezd el, a

cpdf_end_text() pedig lezárja azt. A szövegblokkon belül a cpdf_set_font()-tal állíthatod be a betűtípust, majd a cpdf_show() vagy a cpdf_show_xy() használatával szöveget hozhatsz létre. A cpdf_show() az aktuális pozíciót használja, a cpdf_show_xy() viszont a balalsó sarokhoz viszonyított x,y-koordinátákkal megadott pozíciót fogadja el. A koordináták egysége alapértelmezésben 1 inch, mivel semmi más nincs a cpdf_page_init()-ben beállítva.

A cpdf_finalize() befejezi a dokumentumot, a cpdf_save_to_file() a szkripttel azonos könyvtárban azt lemezre írja, a cpdf_close() pedig eltávolítja a memóriából. A kód a következő linket adja eredményül, a PDF-dokumentum pedig a 11.9 ábrán látható:

```
clibpdftest.pdf                                                    ^
```



11.9 ábra A cpdf_show()-val szöveget mutató PDF

Szöveg létrehozása PDF-dokumentumban PDFlib-bel

Ez a példa az előző megoldáshoz hasonló megközelítést alkalmaz, de a ClibPDF-függvények helyett PDFlib-függvényeket használ. A legszembetűnőbb különbség a dokumentum létrehozásában a pdf_new() és a PostScript fontnevek helyett használt szabványos fontnevekben van. Szabványoson az operációs rendszereden levőket értem. A pdf_findfont() hamisat ad vissza, ha nem találja a betűtípust:

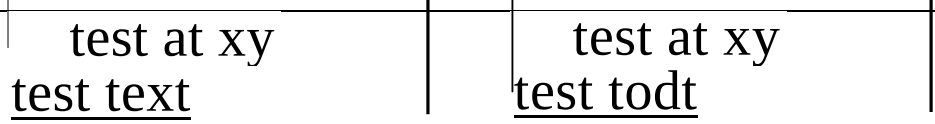
```
$doc_name = "pdflibtest.pdf";
$doc path = dirname ($PATH TRANSLATED) . "/";
Sfont["face"] = "Times New Román";
$font["size"] = 40;
$doc = pdf_new();
pdf_open_file($doc, $doc_path . $doc_name);
pdf_begin_page($doc, 288, 360);
$serif = pdf_findfont($doc, $font["face"], "hőst", 1);
if ($serif)

   pdf_setfont($doc, $serif, $font["size"]);

else
```

```
   print("<br>Could not find font: " . \$font["face"] );

pdf_set_value($doc, "textrendering", 1);
pdf^show($doc, "test text");
pdf_show_xy($doc, "test at xy", 40, 40);
pdf_end_page(Sdoc);
pdf_close($doc);
pdf_delete($doc);
print("<br><a href=\"./" . $doc name . "\">" . $doc_name . "     "   );
```

A pdf_set_value()-függvény a második paraméterben megnevezett értéket beállítja a harmadik paraméterben megadott értékre. A textrendering-parancs megváltoztatja a szöveg rajzolását. (A 11.1 táblázatban felsoroltam a textrendering lehetséges értékeit.) A 11.10 ábra azokat az eredményeket mutatja, amelyek két különböző értékkel való kísérletezés során jöttek ki. A bal oldalon kitöltött szöveget találunk a 0 értékből, a jobb oldalon pedig rajzolt (körvonalazott) szöveget az l-es értékből.

11.1 táblázat A textrendering értékei.

	Leírás
0	Kitöltés — Egyszínű szöveg.
1	Vonás — Körvonalazott szöveg.
2	Kitöltés és vonás — Egy szín különböző színű körvonalakkal
3	Láthatatlan
4	Kitöltés és hozzáadás a kivágandó görbéhez — A kivágandó görbével a szöveget görbékre formázza
5	Vonás és hozzáadás a kivágandó görbéhez
6	Kitöltés, vonás, és hozzáadás a kivágandó görbéhez
7	Hozzáadás a kivágandó görbéhez



11.10 ábra A pdf_show()-val kitöltött szöveget (bal oldal) és körvonalas szöveget (jobb oldal) mutató PDF

Szöveg létrehozása GIF-, JPEG-vagy PNG-képekben

Ez a példa ugyanazt a megközelítést használja, mint az *előző* megoldás, de PDFlib-függvények helyett képfüggvényekkel.

A képfüggvények tudnak TrueType-betűtípust olvasni, de ehhez külön telepítési munkára van szükség, így a példa az öt bepített betűtípus egyikét használja, amelyet a **$font["size"]** állít be és az **imagechar()** használ. A fájl beállítási kódját megváltoztattam, hogy a **realpath()-t** használja arra, hogy a relatív fájlnevet, amelyet egy weboldalon használsz, a PHP-fájlfüggvények számára alkalmas abszolút névre konvertálja.

Az **imagecreate()** a megadott szélesség és magasság felhasználásával hoz létre képet a memóriában. Arra nincs lehetőség, hogy közvetlenül lemezre mentsük, de mivel nem valószínű, hogy a képet többboldalas PDF-szerű dokumentumokban használnánk, erre nincs is szükség. Az első **imagecolorallocateQ** egy olyan színt határoz meg, amely a képen bárhol használható, és egyúttal az alapértelmezett háttér lesz. Az **imagepng()** lemezre menti a képet, az **imagedestroy()** pedig eltávolítja a memóriából. Mindössze ezekre van szükséged egy üres képfájl létrehozásához.

A **második imagecolorallocate()** a **black** színt határozza meg a szöveg számára, és ezt használja a két szövegíró függvény. Az első, a **imagecharQ** egy karaktert ír a képfájlhoz, függetlenül a megadott szövegtől. A második szövegíró függvény, az **imagestring()** egy teljes sztringet ír egy csapásra, a többi paramétere pedig megegyezik az imagechar()-évél. Mindkét függvény elfogadja az **imagecreate()** képazonosítóját, illetve egy betűtípus azonosítót, ahol az 1-től 5-ig a beépített, utána pedig a felhasználó által definiált fontok vannak. A szöveg elkezdéséhez mindkettő elfogad x,y-koordinátákat. A 0,0 a kép bal alsó sarkát jelöli, és a x,y-koordináták a szöveg első karakteréig számolnak. A függvények szöveges sztringet, illetve utána egy **imagecolorallocate**()-bői származó színazonosítót is elfogadnak:

```
$image["name"] = "imagetest.png";
$image["path"] = realpath("./" . $image["name"]); $ font["size"] =
5;
$image["id"] = imagecreate(288, 360); $white
= imagecolorallocate($image["id"], Oxff, $black Oxff, Oxff); 0)
= imagecolorallocate($image["id"], 0, 0, ;
imagechar($image["id"], $font["size"], 10, test text", $black);
10, imagestring($image["id"], $font["size"], 40, "test at xy", $black)
40 imagepng($image["id"], $image["path"]);
imagedestroy($image["id" ] ) ;
print("<br><a href=\"./" . $image["name"] . "\"
target=\"_blank\">" . $image["name"] . "</a>" );
```

Az imagecolorallocate() képazonosítót és három, a piros, zöld és kék színkomponenseket jelképező számot fogad el. A példában a feketét decimális számrendszerben 0,0,0-ként definiáltam, a fehér, szintén decimálisban a 255,255,255 lenne. A decimális helyett a fehéret hexadecimális jelölésben definiáltam, ahol a színek a OxO-tól a Oxff-ig terjednek, és megegyeznek a **<font>**- és más tag-ekben használt HTML-színparaméterekkel. A kékeszöld berill

(aquamarine) egy fonttag-ben a **color="#7fffd4",** amely megegyezik a következő imagecolorallocate()-tel:

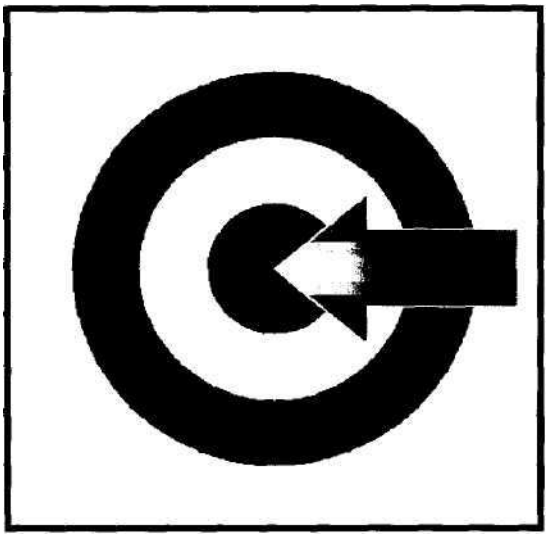
```
$aquamarine = imagecolorallocate($image["id"] , 0x7f, Oxff, 0xd4);
```

Ez az PNG-példa. Mennyire bonyolut GIF-, JPEG- vagy Windows BMP-képeket létrehozni? Egyszerűen cseréld ki a **imagepng()** sort a következővel:

```
imagegif($image["id"],  $image["path"]);
imagejpeg($image["id"],  $image["path"]);
imagewbmp($image["id"],  $ image["path"]) ;
```

Most jön a trükkös rész. A képfüggvények mögötti alapkönyvtár a 256 színes képszerkesztésen alapul, ahogy a GIF-ekben használják. Van a könyvtárnak új, 24 bites színeket használó (JPEG és hozzá hasonló képek által használt) verziója, illetve egy új **imagecreatetruecolor()**-függvény, amely a képek létrehozását magasabb felbontásban végzi. Kipróbáltam a béta-verziót, így mire ezt olvasod, minden bizonnyal tökéletes működik.

Megjegyzés: Némely böngészők a beállításaidtól függetlenül cache-elik a képeket, így képek tesztelésekor gyakran kell a képeket frissítened, hogy a megváltoztatott oldalt kapd meg.



11.11 ábra Céltábla sémája egy PNG-képen

Diagram létrehozása GIF-, JPEG-vagy PNG-képekben

A 11.11 ábrán látható példadiagram egy egyszerű céltábla, amelynek közepébe nyíl mutat. Ha viszont az előző rész szövegfüggvényeivel kombinálod, hasznos üzleti grafikát hozhatsz létre. A diagram jelezheti a hónapra vonatkozó eladási tervedet, vagy lehet demográ-

fiai elemzés, korcsoportokkal az egyes körökben. Mindenekelőtt azonban a kód az alapvető téglalapok, körök és sokszögek létrehozását mutatja be, amelyeket bármilyen diagramba beépíthetsz.

A kód hasonlít a szöveges példában a PNG-kép létrehozására használt kódhoz. Az imagecreate()-ben 400:'40Q-asra változtattam a kép méretét, és van két további imagecolorallocate()-függvény, amelyek segítségével a nyíl paprikapiros lesz, sárga élekkel.

Mivel ezekben a függvényekben nincsen árnyékolás vagy 3D, lehet, hogy szívesebben rajzolod az alapdiagramot egy díszes 3D rajzolóprogrammal, az eredményt a képbe importálod, és utána csak néhány szót vagy számot szúrsz be. A többi diagramot, például a folyamatábrákat legegyszerűbb a lapos keretben elkészíteni, amelyeket könnyen létrehozhatsz ezekkel a kódokkal.

Az imagerectangle() fekete téglalapot hoz létre, a külső éle körül egy pixel vastagsággal, ahova csinos keretet rajzolhatsz. A PHP 4.0.6-tól felfelé az imagesetthickness() használatával beállíthatod a vonalvastagságot. A korábbi verziókban a példában is mutatott technikát használd, ahol az 5 pixel vastagságú keretet 5, egyenként egy pixellel kisebb keretből hoztam létre. Az imagerectangle() a képazonosítót, a téglalap egyik sarkának koordinátáit, a vele átellenes sarok koordinátáit és egy színt fogad el.

A kör egy olyan ellipszis, amelynek magassága és szélessége megegyezik, így céltáblát a imagefilledellipse()-függvénnyel rajzolhatsz, amely a kép azonosítóját, az ellipszis középpontjának x,y-koordinátáit, a magasságot és a szélességet fogadja el. A külső kör fekete, feketével is van kitöltve, a következő fehér, fehérrel van kitöltve, majd a következő, utolsó kör ismét feketével van kitöltve. Ha sok vonalból álló mintát, például koncentrikus köröket kell rajzolnod, for()-ciklussal tudsz a külső körből a belsőkbe haladni.

A nyíl egy hét pontból álló sokszög, pontjait egy tömbbel adod meg. A külső sárga nyíl a Souter_arrow-tömbben van definiálva, ez a tömb második paraméterként bekerül az imagefilledpolygon()-függvénybe, míg a harmadik paraméter a tömbből használt pontok számát határozza meg. A sokszög összes pontját x,y-párként kell megadni.

A belső, paprika színű nyíl a külső nyílhoz hasontó tömböt használ, amelynek pontjai egy vagy két pixellel beljebb vannak. A következő trükkel készíthetsz szép keretet: rajzold meg a sokszöget kétszer; egyszer a központi színnel kitöltve, majd a keret színével megrajzolva, de kitöltés nélkül. Az eredmény a vonalvastagságtól és a használt szoftver pontosságától függ. Az egyenes függőleges vagy vízszintes vonalak a kép létrehozásának módjától függetlenül működnek, de az átlós vonalak mindig egy kicsit szőrösnek tűnnek. Az ebben a példában használt technikával jobban kontrollálhatod a relatív vonalvastagságot, különösen átlós vonalak esetében.

A kód hátralevő része elmenti a képet későbbi felhasználásra. Amikor tesztelsz, jobb a képeket lemezre írni, mert így összehasonlíthatod a kapott eredményeket, amelyek a sikeres és hibás képeket is tartalmazzák, segítve ezzel a különböző képfüggvények viselkedésének tanulmányozását:

```
$ image["name"] = "imagediagram.png";
$image["path"] = realpath("./" . $image["name"]);
$image["id"] = imagecreate(400, 400);
$white = imagecolorallocate($image["id"] , Oxff, Oxff, Oxff) ;
$black = imagecolorallocate($image["id"], 0, 0, 0);
$yellow = imagecolorallocate($image["id"], Oxff, Oxff, 0x00);
$pepper = imagecolorallocate($image["id"] , Oxff, 0x33, 0x00);
imagerectangle($image["id"] , 0, 0, 399, 399, $black);
imagerectangle{$image["id"], 1, 1, 398, 398, $black);
imagerectangle($image["id"], 2, 2, 397, 397, $black);
imagerectangle($image["id"] , 3, 3, 396, 396, $black);
imagerectangle($image["id" ] , 4, 4, 395, 395, $black);
imagefilledellipse($image["id"] , 200, 200, 300, 300, $black);
imagefilledellipse($image["id"] , 200, 200, 200, 200, $white) ;
imagefilledellipse($image["id"] , 200, 200, 100, 100, $black);
$outer_arrow = array(200, 200, 270, 140, 270, 170, 380, 170, 380, 230,
   270, 230, 270 ,260);
imagefilledpolygon($image["id"] , $outer_arrow, 7, $yellow); $inner_arrow =
array(202, 200, 268, 144, 268, 171, 379, 171, 379, 229,
   268, 229, 268 ,256);
imagefilledpolygon($image["id"] , $inner_arrow, 7, $pepper);
imagepng($image["id"], $image["path"]); imagedestroy($image["id"]);
print("<br><a href=\"./" . $image["name"] . "V
target=\"_blank\">"
   . $image["name"] . "</a>" );
```

Most már minden eszközöd megvan ahhoz, hogy honlapodon dinamikus grafikákat használj. Ha összetettebb képekre van szükséged, azokat offline kell elkészítened a Gimp, a Paint Shop Pro vagy az Adobe Photoshop használatával, amihez szükséged lehet egy tapasztalt képszerkesztőre. Természetesen arra mindig van lehetőség, hogy munka közben figyelve egy profit, az összes általa végzett műveletet PHP-képfüggvényre fordítsd, hogy azt kódodba később beépítsd.

i

12. fejezet

Nemzetközi beállítások

Áttekintés

Van egy honlapod, amelyet 80 ország 400 millió látogatója keres fel, hogy az oldalaidba beleolvasson. Mennyien értik a nyelvet, amit használsz - a mozaikszavakat, technikai kifejezéseket, a nyelvjárást és szlengedet? Használtál olyan színeket, amelyek „jó szerencsét" vagy „halált" jelentenek? Ha egy nyelvre meghatározott karakterkészletet használtál, a karaktereket balról jobbra, jobbról balra, vagy esetleg függőlegesen kell olvasni? Mi a helyzet a dátumokkal, számokkal, pénznemekkel és a copyrightjelekkel?

Ha honlapoddal a világ elé állsz, rengeteg dolgot kell végiggondolnod. Egyszer belenéztem egy honlap naplózásába, és azt találtam, hogy egy hónapon belül 82 különböző országból kattintottak oda. A honlap angolul van, és a nyitó oldalnak magas a látogatottsága, ezért megnéztem az alacsonyabb szintű oldalak adatait is. A 82 ország legtöbbjéből ezeket az oldalakat is megnézték, de a 82 ország közül a legtöbbnek nagyon magas a nyitó oldalról való kilépési aránya. Mindez mit jelent?

Több mint 200 ország van a világon, de némelyikben az Internethez való hozzáférés valamilyen okból korlátozott, így valószínűtlen, hogy ezekből az országokból látogatóid érkeznek. Vannak országok, ahol a hozzáférés költségei olyan magasak, hogy csak az ország exportőrei próbálnak vevőt találni áruiknak. Az eladással foglalkozó oldalak általában 80 különböző országból kapnak látogatókat, így valószínűleg ennyi országban elég alacsonyak az Internet-hozzáférés költségei ahhoz, hogy vásárlók széles választékát vonzzad honlapodra.

Körülbelül 15 országból érkező látogatók mélyednek el egy angol nyelvű oldalon, a többiek megnézik a nyitó oldalt, majd továbbmennek. A többi mintegy 60 ország magas kilépési aránya azt jelzi, hogy azokban az országokban kevesen tudnak annyira angolul, hogy oldaladról vásároljanak. Tudnak annyira angolul, hogy keresőkben rákeressenek a software vagy PlayStation szavakra, de annyira nem, hogy az oldaladon elnavigálhassanak, kiválasszák a megfelelő PlayStation-modellt, és kitöltsék a megfelelő űrlapokat. Néhány egyetemi kutató végigmegy az oldalakon, éppúgy, ahogy az adott ország helyi keresőmotorjai is, de ez nem elég az üzlet fenntartásához.

Ahhoz, hogy az ezekből az országokból érkező látogatókat megtartsd az oldalaidon, a saját nyelvükön kell az oldalakat megjelenítened. A saját nyelvükön kell elmagyaráznod a betűszavakat, a szlenget és a helyi jellegzetességeket. Tudod mi az a levonó? Azt a szót hallottad már, hogy ludáj? És azt, hogy csurka? (Ezek azt jelentik, hogy matrica, ovális, copf.) Ausztráliában a babakocsit pram-nek nevezik, Amerikában viszont stroller-nek. Számtalan hasonló példát lehetne találni.

A kulturális különbségek nemcsak a nyelvre korlátozódnak, a színhasználatra is kiterjednek. Vannak országok, ahol az esküvőn a menyasszony fehéret visel, de Kínában fehéret a gyász színeként viselnek az emberek. Ausztráliában szívesen hordanak narancssárga vagy zöld kitűzőt, vagy leginkább mindkettőt, de vajon bölcs döntés-e a zöld Észak-Írországban, a narancs pedig Írország többi részén? (Ha megvannak a megfelelő színek a honlapodra, kérlek ellenőrizd, hogy a színvakok is el tudják-e olvasni.)

r

Ez a fejezet azt mutatja meg, hogy építheted fel úgy a kódodat, hogy választhass a látogatóidnak megfelelő szavak, színek és egyéb lehetőségek között. A bemutatott megoldások elegendő alapot nyújtanak ahhoz, hogy következetesen és hatékonyan cserélgess, miközben az esetleges jövőbeli változtatások könnyen elvégezhetőek maradnak.

Nyelv vagy ország észlelése

A böngésződben nézd meg az Edit | Preferences | Language (Szerkesztés | Preferenciák | Nyelv) vagy az ennek megfelelő menüpontot. Több nyelvet is kiválaszthatsz, és beállíthatod a fontossági sorrendet is. Ha angolul és koreaiul olvasol, a listát en ko-ra vagy ko en-re állíthatod, attól függően, hogy melyik nyelvet részesíted előnyben.
A www.w3.org/WAI/ER/IG/ert/iso639.htm-oldalon a böngésződben használható nyelvek két- vagy hárombetűs kódjainak listáját találod. A nyelvkódokhoz országkód is tartozhat. Az országkódok hivatalos listáját a www.din.de/gremien/nas/nabd/iso3166ma/codlstpl/ /en_listpl.html oldalon találod. Az angol kódja az en, de az USA-ban használt angolt így jelölik: en-US. A nyelvkódok általában kis-, az országkódok viszont nagybetűsek, de ezt nem minden böngésző tartja be.

Nyelv vagy ország észlelése a böngészőből

Ha a böngészőben beállítottad egy vagy több nyelv kódját, a böngésző minden egyes oldallehíváskor tartalmazza a kódokat. A szkripted vagy a webszervered a megfelelő nyelvű oldallal válaszolhat. Ha a felhasználó céges hálózatot használ, vagy Internet-cafében van, lehet, hogy nem tudja a böngészőben megváltoztatni az országkódot, így a kódot csak irányelvnek használd. Az oldaladat a böngésző által kiválasztott nyelven küldd el, de minden lehetséges nyelvre mutasson link, hogy a felhasználó kiválaszthassa a neki legjobban megfelelő nyelvet, és session-rekordjába eltárolhassa. Ha megadják a nyelvkódok listáját, és az első nyelv nem elérhető, addig menj a listán, amíg nem találsz megfelelő nyelvet. Ha nincsen ilyen, akkor legyen az angol az alapértelmezett nyelv, mert a különböző szabványok ezt határozzak meg alapértelmezésben.

Bizonyos esetekben a HTTP hivatkozási sztringet megnézve a domainnévből vagy az IP-címből megállapíthatod, melyik országból jött a látogató. Ez a technika azonban rengeteg hibát okozhat, így csak nagy vonalakban hagyatkozhatsz rá. Itt van egy egyszerű példa. Azon a héten, amikor éppen tengerentúli utat terveztem, az AltaVista elkezdte ezt a technikát használni, hogy az embereket az országspecifikus oldalra irányítsa. Amikor beírtam, hogy altavista.com, automatikusan az altavista.com.au-oldalra jutottam. Amikor megpróbáltam beírni azt, hogy altavista.co.uk, megint az altavista.com.au-oldalra jutottam. Akkoriban az altavista.com.au-oldalakon nem volt megfelelő időjárás-jelentés a tengerentúlról, így átváltottam a Yahoo-ra, az AltaVista pedig elveszített egy régi felhasználót.

Ha kizárólag a domainnevük vagy IP-címük alapján kategorizálod a felhasználóidat, gyakran tévedhetsz, és az embereket megakadályozod abban, hogy azt válasszák, amit akarnak. Ha azt veszed észre, hogy valaki egy ‚de-re végződő domainről látogatja a honlapod, lehet, hogy az a valaki egy németországi német, de az is lehet, hogy egy francia lakos, aki a mun-

kahelyi hálózaton keresztül internetezik, amely történetesen Németországban csatlakozik az Internethez. Az egyik ügyfelem, egy multinacionális pénzintézet központja egy másik országban lévő leányvállalaton keresztül kapcsolódik az Internethez, mert ott a csatlakozási költségek alacsonyabbak. A te CompuServe-ügyfeled, aki egy USA-beli proxyn keresztül kapcsolódik a hálózathoz, lehet egy Olaszországban dolgozó ausztrál üzletember, aki görög oldalakat próbál felkeresni, ahol görög nyelvtudását gyakorolhatja, mert a családja onnan származik, és Görögország lesz útjának következő állomása. Az AltaVista önkényes átirányítása sok ügyfelébe kerülhetett.

Nyelv vagy ország észlelése az Apache-csal

Az Apache nagyszerű termék remek lehetőségekkel, és amennyiben csak HTML-t használsz, az Apache nyelvátirányítója a megfelelő oldalt adja neked. A látogató beállítja nyelvi preferenciáit a böngészőben, te beállítasz néhány opciót az Apache-ban, és az Apache a megfelelő oldalt szállítja. Ha van egy index.html-oldalad angolul és olaszul, az angol változatot nevezd index.html.en-nek, az olasz változatot pedig index.html.it-nek. Az Apache .conf-fájlba pedig a következőket szúrd be:

```
AddLanguage en .en
AddLanguage it .it
LanguagePriority en
it
```

Ha van az Apache-ról egy jó könyved, mint amilyen a Greg Holdén és Nick Wells által írt *Apache Server Commentary* (The Coriolis Group, Inc.), vagy hasonló lehetőségekkel rendelkező másmilyen webszervert használsz, más trükköket is alkalmazhatsz arra, hogy a böngészőjükben levő nyelv alapján irányítsd látogatóidat a honlapodon - de ez a nem mindig legjobb megoldás. Lehet, hogy a japán látogatóid böngészőjében a jp van beállítva, de az angol nyelvű oldalaidra kíváncsiak, mert gyakorolni akarják angoltudásukat. Vagy ha japánul is akarják olvasni az oldalaidat, lehet, hogy a cég írországi leányvállalatának irodájában ülnek, és nem tudják a böngésző nyelvbeállítását megváltoztatni, mert a hálózati adminisztrátor lezárta a beállításokat.

A legjobb megoldás a PHP

Szabályozd az ország és a nyelv beállítását a PHP-kódodban, így látogatóidnak a böngészőktől függetlenül a legjobb megoldást adhatod. Engedd látogatóidnak, hogy szabadon bejelentkezzenek, így saját maguk állíthatják be és menthetik el profiljukban a nekik legjobban megfelelő beállításokat. A szabadon olyan bejelentkezést értek, amelyet saját maguk alakíthatnak ki, regisztrációs folyamat, megkötések és zaklató marketinges kérdések nélkül. A lényeg abban van, hogy ösztönözd felhasználóidat arra, hogy a bejelentkezéssel testre szabják a honlapodat.

Milyen választási lehetőségeik legyenek?

- *Nyelv - A* látogatók meghatározhassák, hogy milyen nyelven akarják olvasni az olda lakat.
- *Karakterkészlet* - Bizonyos nyelvek speciális karakterkészletet igényelnek, de ezek nem minden böngészőben vannak meg. Készülj fel erre az esetre is.

Kiíratás iránya - Vannak nyelvek, amelyeket jobbról balra kell kiírni, és vannak, amelyeket függőlegesen.

Ország - Ha a honlapod országspecifikus információkat közöl, engedd, hogy a felhasználók kiválasszák az alapértelmezett országot. írasd ki mind az országnevet, mind az országkódot, mert vannak országok, amelyeknek több mint egy kódjuk van, és vannak kódok, amelyek több országra vonatkoznak.

Dátumformátum - Mutasd meg a vásárlónak a dátumformátumot, amelyet használsz, amikor országot választanak: HH-NN-ÉÉÉÉ, NN-HH-ÉÉÉÉ vagy ÉÉÉÉ-HH-NN.

Számformátum - Mutasd meg a vásárlónak, hogy milyen számformátumot használsz országára: mind a tizedesjel helyén használt karaktert (, vagy .), mind az ezreseket elválasztó karaktert (, vagy .).

Keresési alapbeállítások - Ha van keresési lehetőség, mentsd el az olyan opciókat, mint a szülői ellenőrzés, az ország, a keresés típusa, az oldalanként megjelenített eredmények száma.

Színek - Ha színezett hátteret használsz, add meg a felhasználóknak a színek kikapcsolásának lehetőségét. Vannak olyan színezett hátterek, amelyek nem működnek LCD-monitorokkal, összeütközhetnek a WebTV-hez hasonló technológiákkal, és lehet, hogy más országokban negatív kulturális jelentésük van.

Felhasználói név - Ha az oldalad „Szia Péter" vagy a felhasználói név hasonló használatával kezdődik, ne feledd, hogy a gyakori nevek hamar elfogynak, és lehet, hogy Péternek „Péteri 10"-ként kell bejelentkeznie. Add meg a látogatóknak a lehetőséget, hogy barátságos nevet adjanak meg.

Email-cím - Ha megpróbálod a látogatók e-mailcímét összegyűjteni, add meg nekik a lehetőséget, hogy kijavítsák, ha időközben megváltoztatták. Gyakran előfordul, hogy kezdetben hamis címet vagy freemailes címet adnak, és csak akkor adják meg az igazi címüket, amikor már megbíznak az oldaladban.

Jelszó - Sok olyan jelszóval védett oldal van, amelyeknél nem lehet a jelszót megváltoztatni. Engedd meg a felhasználóknak, hogy megváltoztassák jelszavukat, és készülj fel arra, hogy hogyan kezeled az országspecifikus billentyűzetről érkező speciális karaktereket. Ha valaki kana billentyűzeten gépeli be jelszavát, majd Tokióból Helsinkibe repül, onnan is be tud jelentkezni?

Egyéb információ - Ha gyűjtőd az életkor, a nem és a marketingosztály által kért többi hülyeséget, készülj fel, hogy az emberek addig hazudni fognak, amíg nem bíznak meg az oldaladban, és akkor javítják ki az adataikat, ha annak eredményét látják. Jobban hirdethetsz bannerrel, ha az emberek megadják valódi életkorukat. Ha megmondod a felhasználóknak, hogy mire használod az adataikat, és ez miért jó nekik, akkor valószínűbb, hogy őszintén válaszolnak. Az, hogy látogatóid milyen arányban adják meg személyes adataikat, nagyban függ kultúrájuktól és az országuktól.

Nyelvi követelmények

A HTML nyelvmegjelenítési opciók a www.w3.org/TR/html401/struct/dirlang.html-ol-dalon vannak leírva, tartalmazzák a két karakteres nyelvkódokat, az országonkénti nyelvváltozatokat, és a szöveg megjelenítésének irányát.

A böngésző először egy lang=-paramétert keres a szöveget körülvevő tag-ben, például egy idézettag-ben. Ha ez nincs megadva, a böngésző kifele haladva keresi a lang=-paramétert a bekezdés- vagy fejezettag-ekben. Ha nem talál ilyen tag-et, akkor a böngésző egy HTTP "Content-Language"-fejIécet keres, és végül a böngészőben beállított alapértelmezést használja. Megjeleníthetsz olyan angol nyelvű oldalt, amely egy francia filozófus által írt, franciául megjelenített bekezdést tartalmaz. Ha a francia egy görög filozófustól idéz, a görög idézet a francia bekezdésen belül megjelenhet görögül.

Ha a nyelvet egy nyelv országspecifikus változataként határoztak meg, mint amilyen az en_US, a böngésző vagy valami különlegeset csinál, vagy figyelmen kívül hagyja az országkódot, és az alapnyelvet használja. A böngészők különböző szinten támogatják a nyelvi lehetőségeket, és néha a Cascading Style Sheet (CSS) és a stíluskiterjesztő nyelv (XLS) ellentmondó követelményein kell átvergődniük.

A HTML tartalmazza a dir = LTR- és dir = RTL-opciókat, amelyekkel meg lehet határozni, hogy a szöveg balról jobbra vagy jobbról balra íratódjon ki. Ha az oldalad Unicode-(www.unicode.org/index.html-) karakterkészletet használ, a Unicode-beállítás felülírja a dir=-t. Vannak olyan karakterkészletek, amelyek függőlegesen íródnak ki, és a legújabb böngészők támogatják a függőleges megjelenítődést. Az oldalak vertikális szövegekhez való formázásához a CSS, XLS és XLT (XML alapú formátumok szótárakhoz és szaknyelvekhez) legújabb információit kell elolvasnod.

Alkalmazásspecifikus vagy honlapspecifikus beállítás

riajjöhonlapod számtalan alkalmazást integrál, minden alkalmazás máshogy fogja testre szabni az eredményt. A vásárlóid feszültté válhatnak, ha az egyik oldalon beállítják a preferenciáikat, majd egy másik oldalon ezt meg kell ismételniük. Hogyan oldhatod ezt meg?

Néhány honlapon megpróbáltam beállítani, hogy az alkalmazások egy közös módszer használatával visszakeressék és elmentsék a beállításokat, de rá kellett jönnöm, hogy rengeteg kódot gyakran frissítenek, és az egyes frissítések kitörlik a teljes fejlesztésedet. Hozzátehe-ted a változtatásaidat egy nyílt forráskódú projekthez, de egy népszerű nyílt forráskódú alkalmazáson 200 ember is dolgozhat, és mindegyik másmilyen módszert akar használni. Kevés reményed van arra, hogy rövid idő alatt jelentős változtatást végzel el, és könnyen megtörténhet, hogy valaki hamarosan megváltoztatja a te változtatásaidat is.

A legjobb megközelítésnek azt tartom, ha hagyod az egyes alkalmazásokat, hogy azok a létező mechanizmusaikon keresztül terjesszék és használják a beállításokat, te pedig koncentrál) az egyes alkalmazások ínit.mc-fájljára. Normális esetben minden alkalmazásnak van

init.inc vagy init.php nevű inicializáló fájlja. Ebben a fájlban az alkalmazás beállításaira vonatkozó definíciók vannak, és egy kód, amely visszakeresi a profilokban levő, az alkalmazás beállításait felülíró egyéni beállításokat. A kód elején az alkalmazás változóit feltöltheted a honlap inicializáló kódjának értékeivel, vagy az init.inc-fájl végére szúrhatsz egy kis kódot, amely a fájl értékeit a honlap értékeivel helyettesíti.

Az egész honlapra vonatkozó azonos beállítás lenne a legjobb, de egy alkalmazás nagymértékű módosítása esetén meg lennél lőve. Próbáld a kódodat az alkalmazás kódja köré rakni, vagy az alkalmazását a sajátod köré, hogy az minél jobban illeszkedjen, és egy új változat esetén minél kevesebb munkád legyen. Lehet, hogy a vásárlóid nem fognak megdicsérni, hogy csak egy központi beállítás van, de azért biztosan panaszkodnak, ha több is van. Az alkalmazások integrálásához szükséges extra változtatások bőven megérik a ráfordítást.

t

GNU-recode

A GNU újrakódoló szoftver az egyes karakterkészleteket egy másikra fordítja, és jelenleg mintegy 150 karakterkészlet között fordít. Jóllehet, a fordítás nem tökéletes, ami elsősorban azért van, mert nem minden karakternek van az összes karakterkészletben pontos megfelelője. Az újrakódolásról a **www.gnu.org/software/recode/** címen találsz információt, de felhasználói kézikönyvet a weben kell keresned.

Az újrakódoló legalább 10 éves és ennek megfelelően le van tesztelve, így a fordítások a lehető legjobbak. Ennek ellenére manuálisan le kell ellenőrizned a fordításokat. Ha másik nyelvre és karakterkészletre fordítasz, ellenőriztesd az eredményt olyan valakivel, aki állandóan használja a célnyelvet, így a technikai pontatlanságot és a gyakran elkövetett hibákat is ki tudja szűrni.

Az ASCII-t több névvel is meghatározhatod, így ascii-vel vagy us-szal. A latin karakterkészletek több, egymáshoz kapcsolódó karakterkészletet tartalmaznak, többek között a Latin- 1-et, más néven a 11-et, amely a nyugat-európai nyelvek alapvető karakterkészlete.

A PHP-ban két újrakódoló függvény van, a **recode_string()** és a **recodefileQ. A recode()** a **recode_string()** aliasa. A következő kód a recode_string()-et mutatja be, ahogy a speciális karakterek eltávolításával egy sztringet az ASCII-ről a képernyőn való megjelenítésre alkalmassá tesz:

```
print(recode^string("ascii..flat",  "Test code with diacritical
    marks". " kaacute;, SAacute;, SAtilde;, &Aring;, and a EURO
    Seuro;"));
```

A következő kódban a **recode_file()** a bemeneti fájlt kódolja újra, ASCII-ről a Latin-1 karakterkészletre:

```
$test = fopenCtest.txt', 'r');
$recoded = fopen('recoded.txt', 'w');
recode file ( "ascii..latin-1", 5test, $recoded);
fclose ($test);
fclose(Srecoded);
```

íttiaÁlíi í

GNU-gettext

A GNU-gettext-jét a www.gnu.org/software/gettext/gettext.html-oldalon mutatják be, kézikönyvét pedig a www.gnu.org/manual/gettext/html_mono/gettext.html-oldalon találod. A gettext a *domain* (tartomány) kifejezéssel köti össze a fordításokat a honlapod egy területével, jellemzően egy alkalmazással. Ha oldalaidon van e-mailszolgáltatás és bevásárlókosár is, a vásárlási és levelezési kifejezések egy nagy fordítási listába való kombinálása túl nagy munka lenne. Általában egyszerűbb a levelezést fejlesztőknek összeállítani a saját fordítási táblázatukat, a bevásárlókosár üzemeltetőinek pedig egy másik listát létrehozni. A levelezéssel kapcsolatos listának add a mail.mo nevet; ezt a listát a mail-domain használatával érheted el. A bevásárló kosár valami olyan nevet és domaint kap, hogy shop.mo és shop. A domainnek egyezni kell a .mo előtt szóval.

Ot gettext-függvény van:

- bindtextdomain() — A domamt ahhoz a könyvtárhoz társítja, amely a domainhez tar tozó fájlt tartalmazza.
- textdomainQ - A következő gettext() által használandó domaint állítja be.
- gettext() - Fordítást hív meg.
- dgettext() - A kiválasztott domaintől eltérő domain használatával *végez* el egy fordítást.
- dcgettextQ - A dgettextQ egy olyan változata, amely kategóriaválasztást tartalmaz.

A domain beállítása lassú, mert a kódnak fájlokat kell találnia, tehát állítsd be egyszer a domaint, és ne változtasd. Ha alkalmanként speciális domainre van szükséged, például a hiba üzenetekhez, használd a dgettext()-et. Ha egy honlapon folyamatosan két domaint kell cserélgetned, mérlegeld a két MO-fájl összefésülésének lehetőségét.

A PHP 4.0.7 Dev.-ben a gettext Unix-os verziója megbízható, a Windows-os verzió pedig CGI-módban működik jól. Azonban a Windows-os verzióval van egy „egyszer használatos" probléma, amikor Apache-modulként használják. A Win32 modulváltozat csak az első igényelt nyelvet fogadja el, az ezt követő igényléseket figyelmen kívül hagyja. A fájlokat szintén csak egyszer ellenőrzi, az ismételt ellenőrzést visszautasítja. Ha a nyelvet először németre állítod, és utána állítod át spanyolra, mindaddig németül kapod a fordításokat, amíg nem indítod újra a webszervert. Ha rossz névvel telepítesz egy könyvtárat, majd használod a gettext()-et, a fordítás nem fog működni. A könyvtár átnevezése nem oldja meg a problémát, mivel a gettext-kód a régi könyvtár információt tartja a cache-ben. Az egyetlen megoldás a webszerver újraindítása, vagy a CGI-vákozat használata (ahol minden beállítás szkriptenként frissül). Ha a gettext-nyelvkönyvtárak és -fájlok új telepítését teszteled, CGI-módban használd a PHP-t.

Helyesírás

A legtöbb honlapkészítőnek akkor kell először a helyesírás-ellenőrzővel foglalkozni, amikor levelező alkalmazást írnak, másodszor pedig akkor, amikor lehetővé teszik, hogy a fel-

használók, akik nem programozó tartalomszerkesztők, közvetlenül vihessenek tartalmat az oldalra. Mindkét alkalmazás különböző kihívásokkal szembesít minket, amelyekkel érdemes a helyesírással való küzdelem előtt foglalkozni.

Ha az oldaladon plüssmacikat árulsz a gyűjtőknek, az oldaladra tartalmat írók a technikai kifejezések adott halmazát fogják használni. Jó előre összeállíthatod szótárukat, és mindegyikük használhatja ugyanazt a szótárt. Ha nyilvános e-mailszolgáltatást hozol létre, nem tudsz a felhasználóknak szótárt készíteni, így egy alapszótárt kell létrehoznod, és lehetővé kell tenned, hogy az egyes felhasználók létrehozzák az általuk használt szavakból álló saját szótárt.

A plüssmackós honlapot egy alapnyelven kell megírni, és le kell fordítani más nyelvekre. Létre kell tehát hozni egy szótárt az oldalt megíróknak, és hagyni, hogy az egyes fordítók kialakítsák a saját, nyelvspecifikus szótárukat. A nyilvános levelező honlapon minden nyelvhez kell egy szótár, illetve a felhasználóknak ki kell tudniuk választani, hogy az adott levélhez milyen szótárt, illetve az adott nyelvhez milyen alapértelmezett szótárt akarnak használni.

Mit csinál a helyesírás-ellenőrződ, ha egy felhasználó afrikaansul ír és zulu szöveget idéz? Vagy próbáld meg bekezdésenkém ellenőrizni a szöveget?

Négy lehetőséged van a PHP-val:

1. Választhatsz felhasználásra kész nyílt forráskódú alkalmazást és az alkalmazás által használt valamilyen helyesírás-ellenőrzőt, beleértve a Java és JavaScript alapú rendsze reket.
2. Megírhatod a saját szó-összehasonlító alkalmazásodat.
3. Használhatsz Apsell-t.
4. Használhatsz Pspell-t.

Szóegyeztetés

A szavakat ellenőrizheted soundex-indexek, metaphone-egyenértékek és a Levenshtein-összehasonlításon alapuló kifejezés egyeztető használatával (ahogy azt ebben a részben megmutatom). Mindhárom módszer alkalmas az elírt szavak alternatíváinak megtalálására. Mivel az én agyam mindig SQL-ben gondolkodik, létrehoznék egy adatbázist, amelyben az egyik index a szavak listája lenne, a másik pedig a szóegyeztető függvény eredményei. Ezt követően ugyanezen függvény használatával rákeresnék a bejövő szavakban rejlő hasonlóságokra. Ha a felhasználó a *money* helyett azt gépeli be, hogy *monie*, akkor a soundex-nek és a metaphone-nak jelezniük kell, hogy a *money* úgy hangzik, mint a *monie*, és megfelelő választás a helyesírás-ellenőrzőnek. Nézd meg először a függvényeket, majd használatukat.

soundexQ

A soundex() egy mutatót eredményez, hogy hogyan hangzik a szó, és hangbeli egyezőség alapján lehet vele szavakat keresni a szótárban. A következő példa néhány szót hasonlít össze a soundexQ használatával: a kód van a bal, az eredményül kapott soundexQ-indexek

pedig a jobb oldalon. A *dia* egy szoftvertermék neve, a **soundex()** a dia *d* betűjét használja az soundex-index első betűjének. A **soundex()** nem veszi figyelembe a magánhangzókat, így a dia 000-t kap a betű után. A *diablo* egy mókás és gyors kocsi neve, és a d után 14-et kap a *b* és / betűk miatt. A másik két szó adott soundex-indexek, amelyek eléggé különböznek ahhoz, hogy jelezzék, hogy a szavak nem lehetnek odaülök:

```
print("<br>"    soundex("dia"));              D000
print("<br>"    soundex("diablo"));           D140
print("<br>"    soundex("dialectric")) ;      D423
print("<br>"    soundex("diacritical") );     D263
```

metaphoneQ

A metaphoneQ a **soundex**() egy alternatívája, amelyet a **www.lanw.com/java/phonetic** oldalon mutatnak be. A soundexQ-támogatás be van építve bizonyos adatbázisokba, így ha ezen adatbázisok valamelyikét használod, a **soundexQ** a megfelelő választás. Ha nem, akkor használj bármilyen adatbázist és **metaphone()-t.**

A soundex()-hez hasonlóan a metaphoneQ is egy mutatót eredményez arra vonatkozólag, hogy hogyan hangzik a szó. Hasonló szavak keresésére használható, és a soundexQ-szel való hasonlóságok miatt a következő kód az előző példa szavait használja **metaphoneQ** -nal. A **metaphoneQ** értékeinek összehasonlítása bonyolultabb, mint a **soundexQ** értékeié, hacsak nem korlátozod az összehasonlítások hosszát. A következő példában az összehasonlítás a metaphone-érték első öt, de akár az első három karakterére korlátozva nem hozna eredményt:

```
print("<br>"    metaphone("dia"));                  T
print("<br>"    metaphone("diablo"));             TBL
print("<br>"    metaphone{"dialectric"));       TLKTRK
print("<br>"    metaphone("diacritical") ) ;    TKRTKL
```

levenshtein()

A Levenshtein-távolság azon karakter számát jelenti, amelyeket hozzáadni, törölni vagy megváltoztatni kell annak érdekében, hogy két kifejezés megegyezzen. Jóllehet, nincs sok értelme egyező szavakat találni egy helyesírás-ellenőrzőben, hasznos lehet olyan feladatokra, mint nevek és címek egyeztetése. A **metaphoneQ** és a **soundexQ** erősen érzékeny a szavak első betűire, és nem tekintené egyezőnek a következő helyneveket: *The Great Sandy Desert* és *Great Sandy Desert.* A **levenshteinQ-t** egyező kifejezésekre és helynevekre használd. A Gyors megoldások „Kifejezések és helynevek egyeztetése **levenshteinQ-nel"** című részében mutatok erre példát.

Egyezés

Tegyük fel, hogy egy online utazási irodát hozol létre, Van egy listád a városokról, és segíteni akarsz ügyfeleidnek, hogy a megfelelő városba foglaljanak repülőjegyet. Hogyan tudsz egy félregépelt városnevet a listával egyeztetni? Tedd a városokat adatbázisba, soronként egy várost. Ahogy berakod az egyes városokat, számold ki a város metaphone-ját, soundex-ét, és tárold ezeket egy másik mezőben. A másik mezőt - a gyors visszakeresés érdekében - indexeld. Amikor a vásárló egy városnevet gépel be, először keresd ki a nevet az adatba-

zisból, mert még az is lehet, hogy helyesen írja. Ha nincs eredmény, a vásárló által begépelt nevet add be **metaphone()-ba** vagy **soundex()-be,** és így keress az adatbázisban egyezést.

Ha a pontos egyeztetés nem vezet eredményre, az SQL **liké** eszköz segítségével keress rá a metaphone előtagjára. A metaphone-ban levő karakterek és a visszaadott sorok száma alapján a bináris keresés ekvivalensét is elvégezheted. Ha túl kevés sort ad vissza, használj kevesebb betűt. Ha túl sok a sor, növeld meg a keresésben használt betűk számát. Mivel a soundex mindig négy karakter hosszú, minimalizálod a keresések számát, de semmi sem akadályoz meg abban, hogy elvégezz egy második metaphone-keresést (a pontos egyezőség keresése után) is négy karakterrel, amely pontosan ugyanazt az eredményt adja. A remek keresési lehetőségnek köszönhetően értékesített utak jutaléka hamar behozza a nagyobb memória és gyorsabb processzor árát, amelyek a többszörös adatbázis-keresés végrehajtásához szükségesek. Ezután pedig minden további értékesített út tiszta nyereség.

Aspell

Az A az Auld-ot (régi) jelenti az Aspellben. Csak a régebbi Aspell helyesírás-könyvtárakkal működik, és a Pspell váltja fel. Ha egy kódban Aspellt találsz, és fel tudod frissíteni a kapcsolódó könyvtárakat, akkor válts Pspellre.

Az Aspell néhány függvényt használ a szavak helyesírásának ellenőrzésére:

- **aspell_new()** - Új szótárt tölt be a többi függvény számára.
- aspell_check_raw() - Úgy ellenőrzi a szót, ahogy az be van gépelve, ami azt jelenti, hogy csak teljes egyezést fogad el. A megfelelő függvény technikai kifejezések esetén.
- **aspell_check()** - Szófajonként ellenőrzi a szót, a nyelvtani eset megváltoztatásával és a szóközök lenyesésével. A függvény technikai kifejezéseket nem tartalmazó általános szöveg esetén jó megoldást nyújt.
- **aspell_suggest()** - Olyan szavakat javasol, amelyek megegyezhetnek az adott szóval. Használd a függvényt arra, hogy alternatívákat kapj, amikor egy szót az **aspell_check()** elutasít.

Pspell

A Pspell, a *Portable spelling cbecker* (Hordozható helyesírás-ellenőrző) az Aspellt váltja fel. A Pspell az Aspell-könyvtárakhoz javasolt illesztőfelület. Az Aspell-könyvtárak a **http://aspell.sourceforge.net-,** a Pspell-könvtárak pedig a **http://pspell.sourceforge.net**-oldalon érhetők el (mindkettőre szükséged lesz). A PHP 4.0.7dev Win32 bináris állománya nem tartalmazza sem az Aspellt, sem a Pspellt.

A Pspell szkriptedben való használatához először nyiss meg a nyelvhez egy szótárt, ahogy itt látod:

```
Scountry["language"]["code"]     =    "en";
if(!$dictionary    =    pspell    new($country["language"]["code"] ) )
```

```
    print ( "<br>pspell__new failed . " ) ; }
```

Bizonyos nyelveknek több változata is van, így a **pspell_new** második, opcionális paraméterével meghatározhatod a változatokat. A következő példa ellenőrzi az országkódot, majd bizonyos országokra megváltoztatja a paramétereket. Lehet, hogy egy amerikai kereskedő a brit változatot akarja használni, amikor Nagy-Britanniába küld e-mailt, majd a kanadai változatot használja a Kanadába írt leveleihez:

```
$country["language"]["code"]    =    "en";
$country["country"l["code"]    =    "US";
switch($country["country"]["code"])
    í
        case   "CA":
        $option = "canadian";
        break; case "EN":
        $option = "british";
        break;   y-case "\3S"y^
$optipn = "american"; break; }
if(isset($option))
    {
$dictionary = pspell_new($country["language"]["code"], $option); } else
    í
    $dictionary = pspell_new($country["language"]["code"]); }
if(!$dictionary) {
    print("<br>pspell_new failed.") ;
    }
```

A következő kód ugyanennek a műveletnek egy hosszabb változata, amely további függvényekkel újabb opciókat tesz a kódhoz. A **pspell_config_create()** új szótárkonfigurációt határoz meg, a pspell_new_ignore() utasítja a konfigurációt, hogy az *n* betűnél (a példában ez 3) rövidebb szavakat figyelmen kívül hagyja, a **pspell_new_mode()** pedig utasítja a szótárt, hogy dolgozzon tovább, és keressen további lehetséges szavakat. **A pspell_config_personal()** egyéni szótárt definiál az alapszótárhoz. A **pspell_config_repl()** egy fájlt definiál, amely olyan szópárokat tartalmaz, amelyeket a helytelenül írt szavak helyett ajánl fel a program. A **pspell_config_runtogether()** bekapcsolja azt az opciót, amely olyan összetett szavakat is engedélyez, mint például a *sourceforge*, amely egy hasznos weboldal neve:

```
$country["language"]["code"]    =    "en";
$configuration    =    pspell    config
create($country["language"]["code"]);
pspell    config_ignore($configuration,    3);
pspell_config_mode($configuration,    PSPELL_BAD_SPELLERS);
```

```
pspell_config personal($configuration, "/home/me/spell/custom.pws")
pspell_conf ig__repl($configuration, "/home/me/spell/replace.repl");
pspell_config_runtogether($configuration, true); if(!$dictionary =
pspell_new_create($configuration) }
    í
    print("<br>pspell_new    failed.");
```

Amikor megvan a megfelelő szótár, a következő kóddal ellenőrizd a szavak helyesírását:

```
$word   =   "PHP";
if(pspell_check($dictionary, $word))

    print("<br>Ok:     "     .     $word);

else
    í
    print("<br>Reject:     "     .     $word);
```

Ha a szó olyan új szó, amelyet el akarsz menteni egy egyéni szótárba, használd a következő két sor valamelyikét. Ezek a függvények a PHP 4.0.2-ben jelentek meg. Számtalan párhuzamosan használható függvény van, és rendelkezem némi végfelhasználói tapasztalattal arra, hogy a különböző szótárkonfigurációs függvények közül a legjobb választást mutassam meg neked. A legegyszerűbbnek a session-opció tűnik. Miután hozzáadsz valamit a szólistához, ne felejtsd el azt a szkript vége **előtt pspell_save_wordlist{)-tel elmenteni:**

```
pspell_add_to_personal($dictionary,      $word);
pspell_add_to_session($dictionary,      $word);
pspell_save_wordlist($dictionary) ;
```

A pspell_clear_session() mindent töröl a session szólistádból. Ezt opcióként kínálhatod fel azoknak az embereknek, akik tévedésből adtak hozzá szót és újra akarják kezdeni. Jobb lenne egy visszavonási lehetőség, mint a kiválasztási listával megjelenő többszintű visszavon (undo) némely irodai alkalmazásokban, de ehhez Pspellben nincsenek meg a megfelelő függvények:

```
pspell_clear_session($dictionary) ;
```

A **pspell_store_replacement()-tel** elmentheted a hibásan írt szót és annak a helyes megfelelőjét. Állandóan használd a **pspell_save_wordlist()-tet** a frissített helyettesítő lista mentésére:

```
pspell_store_replacement($dictionary, "frieght", "freight");
```

A **pspell_suggest()** egy szót fogad el, és tömbben adja vissza a lehetséges egyezéseket. Ezután a listát egy HTML kiválasztási listában is megjelenítheted, a felhasználó pedig kiválasztja megfelelő helyettesítőt. Adj neki arra is lehetőséget, hogy a saját helyettesítőjét begépelje, és mentsd el azt a **pspell_store_replacement()** használatával:

```
$list = pspell suggest($dictionary, $word);
```

A **pspell_new_personal()** a **pspell_new_create()** egy alternatívája a személyes szótárral való használat esetére. A **pspell_config_save_repl()** pedig a **pspell_config_repl()** egy alternatívája.

Több-bájtos karakterek

A szabványos számítógépes karakter egy bájtot foglal, és egy bájt 256 karaktert jelképezhet. Mit tegyél, ha a nyelvedben 50 000 karakter van? Használj két bájtot, így összesen 65 536 karakter jeleníthető meg. A PHP a több-bájtos karaktereket az mb_-függvényekkel támogatja. Ezek a függvények kísérletiek, és az első kísérletek a különböző japán karakterkészleteket használják.

A japán és a kínai írott nyelv piktogramokat használ, ami nagyjából egy szót, egy szó részét vagy egy kifejezést jelképez. Egy újság körülbelül 5000 piktogramot használ, egy könyv pedig nagyjából 50 000-et. Gondold végig, milyen nehéz a nyugati karakterkészletekben meg- | különbözteti az *1-et* (az egyes szám) az /-tői (a kis L betűtől), majd ezt szorozd meg 1000-rel. A piktogramok pontos megjelenítéséhez több hely kell, és Japánban több karakterkészletet használnak, hogy a különböző médiákban pontosan jeleníthessék meg a karaktereiket.

Van egy speciális karakterkészletük, amelyet mobil kommunikációs eszközeikre, nevesen telefonokra fejlesztettek ki, hogy a karakterek a kicsiny LCD-képernyőn is olvashatóak legyenek. Mielőtt megpróbálsz betörni a japán piacra, gondolj arra a 11 millió ingázójukra, akik mobiltelefonjukon olvassák az Internetet. Nem használják a WAP-ot (Wireless Application Protocol - Vezeték nélküli Alkalmazás Protokoll), egyből túlléptek a WAP-on, ■ és az Internet-hozzáférés soha nem látott módját, a HTML-hozzáférés egy *i-mode-nak* nevezett változatát választották.

A több-bájtos függvények (amelyek listája jelenleg a **www.php.net/manual/en/ /ref.mbstring.php-oldalon** található) közé tartozik a **mb_send_mail(),** azon szabványos levelezési függvények egy változata, amelyek a japán ISO-2022-JP karakterkészletbe kódolják a leveleket. **A mb_send_mail()** a mail()-függvény köré ágyazza magát, elvégzi a kódolást, majd normális elküldésre a mail()-nek továbbítja a levelet.

A PHP-nak vannak függvényei a HTTP-kérésekben levő több-bájtos karakterek kezelésére, beleértve a POST- és GET-inputokat, plusz olyan függvények, amelyek az outputot fordítják. Vannak a szabványos sztringfüggvényeknek helyettesítői, amelyek a több-bájtos karaktereket egy karakterként számolják, és elég összetettek lehetnek, hiszen egy sztring tartalmazhatja több-bájtos karakterek szakaszait, illetve a több-bájtos módot ki- és bekapcsoló ellenőrző karaktereket.

Amikor oldalt, illetve bármi olyat küldesz vagy fogadsz, amit a HTTP úgy értelmez, mint egy fájlt, MIME-fejléc van az elején a fájl leírására. A fogadó ebből tudja, hogy a fájl szöveges/html vagy kép/jpeg, és ennek megfelelően dolgozza fel. Vannak több-bájtos függvények a MIME-fejlécek a Japánban használt MIME-fejlécek ISO-2022-JP karakterkészletből és karakterkészletbe való konvertálására.

HTML-ben 1 ehetőség van a speciális karakterek numerikus hivatkozássá való kódolására, így a speciális karaktereket anélkül tudod a böngészőbe továbbítani, hogy a hálózat vagy a böngésző problémáiba ütköznél. Amíg a böngésző nem értelmezi a speciális karaktereket, addig egyszerű szövegként jelennek meg. A több-bájtos függvények között van olyan, amely a több-bájtos karaktereket HTML numerikus formátumra, illetve fordított irányban konvertálja.

Gyors megoldások

Országinformáció létrehozása

Hogyan láss neki a világ 239 országáról szóló információ feldolgozásának? Kezdd egy saját országodra vonatkozó bejegyzéssel, a további országokat pedig úgy add hozzá, ahogy a honlapod tartalma megkívánja, A példa Ausztráliának készít egy bejegyzést, de ezt helyettesítheted bármelyik országgal.

A következő lista az összes olyan nemzetközi információt megjeleníti Ausztráliáról (országkódja AU), amelyről már szó volt a fejezetben. Ha a PHP nem kap országkódot, a legtöbb alapértelmezett beállítás megfelelő Ausztráliára is, a number_format() a helyes, nn,nnn.nn-formátumban jeleníti meg a számokat. Beraktam a null-lal levő definíciókat is, csak azért, hogy a példában használt adatok teljes tartományát megmutassam:

```
$country["characterset"]["direction"] =
null; Scountry["characterset"]["name"] =
null; $country["color"]["bg"] = null;
$country["color"]["edge"] = null;
$country["country"]["code"] = "AU";
$country["country"]["name"] = "Australia";
Scountry["date"]["full"] = "1 F j, Y";
$country["date"] ["mysql"] = "Y-m-d";
$country["date"]["standard"] = "F j, Y";
$country["language"]["code"] = "en";
$country["language"]["name"] = "English";
$country["number"]["decimai"] = null;
$country["number"]["thousands"] = null;
$country["time"] = "H:i";
```

Fontos a mezők értelmezésének módszere. Ebben a listában a nem használt mezők null-ra vannak állítva, ezzel jelzik, hogy a mező nincsen beállítva, és az ezután következő kódok nak az alapértelmezett értéket kell használni. Ha egy mező nincs meghatározva vagy üres, az adatbázisban null-ként van tárolva, így megkülönböztetheted a szükséges és a nem szükséges mezőket. Amikor egy üres bejegyzést visszakeresel az adatbázisból és PHP-ban tárolod, a PHP törli a mezőt, amint azt az „Országinformáció visszakeresése" című megol dásban láthatod. Az üres/nem használt koncepció végigvitelével jelezheted, hogy mit nem kell beállítani vagy a böngészőnek továbbítani, és a böngésző használhatja a lokális alapér telmezéseket, i-

A számformázás maradhat null az adatbázisban, ami nincs változóba beolvasva. Bármely a number_format()-ot tartalmazó kód egyszerűen használhatja az alapértelmezéseket a formázási paraméter number_format()-ból való kihagyásával. A number_format()-nak van egy sajátossága: vagy meg kell határoznod a tizedesjelet és az ezreseket elválasztó karaktert is, vagy nem kell egyiket sem. A bejövő profilok ellenőrzéséhez a következő kódhoz hasonló igazoló kód szükséges. A példa azt ellenőrzi, hogy a tizedesjel vagy az ezres elválasz-

tó meg van-e határozva a másik nélkül, ha igen, a hiányzót egy megfelelő értékre állítja be. (Ha a tizedesjel a pont, akkor az ezreseket valószínűleg a vessző választja el.) A példa ki-törli a **Scountry["number"]-t,** amennyiben a tizedesjel és az ezreseket elválasztó hiányoznak, így a kódnak később csak a **number** jelenlétét kell ellenőrizni. Azt javaslom, tedd ezt a kódot minden olyan űrlap után, amelyben a felhasználó felülírhatja a beállításokat, vagy ahol a honlap adminisztrátora felfrissíti az országbeállításokat:

```
if (isset <$country["number"] ["decimal"]) or
isset($country["number"]["thousands"]))

    if(!isset($country["number"]["decimal"]))

        if ($country["number"] ["thousands"]     ==    ",")

            $country["number"]["decimal"]    =    ".";

        else { $country["number"]["decimal"]      =
            ",";


    i f ( !isset($country["number"]["thousands"])) if

        ($country["number"] ["decimal"]     ==    ",")

            $country["number"]["thousands"]    =    ".";

        else

            $country["number"]["thousands"]    =    ",";


    }
elseif(isset($country["number"]))

    unset($country["number"]);
```

Ha az adataid már tiszták, tesztelheted őket azzal, ha úgy használod őket, ahogy a szkriptben lesznek használva. A következő példa a **number_format()** és a **$country** beállításainak használatával megjelenít egy számot:

```
function    numf($number,     $decimals=2)

    global    $country;
    if(isset($country["number"]))

        return(number^formát($number,     $decimals,    _
            Scountry["number"]["decimal"],    _
            Scountry["number"]["thousands"]));


    else
```

```
        return(number_formát($number,      $decimals));

print("<br>" numf(50985.7273,     4));
print ("<br>" numf(50985.7273,     6));
print("<br>" numf(50985.7273));
print("<br>" numf(50985.7237));
print ("<br>"              numf(12345 67 8 9012 34 567 8 90.12 34 567 8 90,     10))
```

A következő lett az eredmény. Vedd észre a **number_format()** második paraméterét, amely nem más, mint az általad kért tizedeshelyek száma. A **number_format()** a végén nullákkal tölti fel a számot (a második sorban láthatod), ha a tizedeshelyek számára vonatkozóan nagyobb számot adsz meg, mint amennyi tizedesjegy van a számban. A harmadik sor a **number_format()** felkerekítésének eredményét mutatja, amikor a tizedesjegyeket csonkolnia kell. A negyedik sorban a **number_format()** lefele kerekít. Az ötödik sor pedig egy gyorsteszt arra, hogy hány számjegy megy át a formázáson;

```
50,985.7273
50,985.727300
50,985.73
50,985.72
12,345,678,901,234,567,000.0000000000
```

Apró dolgok, például kerekítési hibák problémát okozhatnak a pénzügyi adatokat megjelenítő oldalak számára, így győződj meg arról, hogy a szerződésedben elhárítasz minden olyan felelősséget, ami a szám pontosságának igazolására vagy a szám megjelenítési módjára vonatkozik. Kérd meg ügyfeleidet, hogy az adatokat olyan listával ellenőrizzék, amilyeneket itt használtam.

Ahogy az Internet és a weboldalak fejlődnek, egyre több információt kell tárolnod, például azt, hogy milyen szülői tartalomosztályozó rendszert használnak az adott országban. Az itt és a következő megoldásokban bemutatott struktúra úgy van kialakítva, hogy a *létező* kód megbontása nélkül szúrhass be új mezőt.

Országinformáció tárolása

A MySQL a többszörös platformokra és az adatok gyors beolvasására a legkönnyebben használható adatbázis, így ebben a példában ezt használom az országinformációk tárolására. A MySQL-t az 5., a többi adatbázist pedig az 5. és 6. fejezetekben mutatom be.

A példa az előző megoldás adataival kezdődik, amelyeket a **$country** nevű tömbben tárolok. Az adatbázis-megjelenítés örökli a tömb kétszintű névstruktúráját, az alulvonás karaktert (_) határoló jelként használva. Ez azt jelenti, hogy az aiulvonást távol kell tartanod a tömbhöz adott új kulcsoktól:

```
$country["country"]["code"] = "AU";
Scountry["country"]["name"] = "Australia";
$country["date"]["full"] = "1 F j, Y";
$country["date"]["mysql"] = "Y-m-d";
```

```
$country["date"]["standard"]  = "F  j,  Y";
$country["language"]["code"]  = "en";
$country["language"]["name"]  = "English";
$country["time"]    =    "H:i";
```

Az adatok a weblaphoz rendelt általános adatbázisba kerülnek, ahol a session-ök és a felhasználói profilok vannak tárolva. A teszt weboldalon az adatbázist a session-kód választja ki, így az adatbázis-kiválasztási lépés nem kell ebbe a kódba. A következő kód eldob minden country nevű létező' táblázatot, így új táblázatot hozhatsz létre. Mialatt tesztelsz, jobban jársz, ha a drop table- és subsequent create table-kódokat a kezed ügyében tartod, hogy a táblázatot bármikor újra létrehozd, ha a próbálkozásaid összekeverik az adatbázist.

A mysql_query() egy SQL-lekérdezést futtat le, az adatokat eredményező lekérdezésekre egy eredményazonosítót ad vissza, illetve az adatbázishoz való létező kapcsolatot igényel, amely a szkript elején van beállítva, amikor az adatbázis alapú session-öket elindítod. Az 5. fejezet bemutatja a mysql_-függvényeket, a 19. fejezet pedig elmagyarázza a session-öket:

```
$sql = "drop table if exists country";
if(mysql_query($sql) )

   print<"<br>SQL worked.");

else

   print ("<br>SQL failed:      $sql      error:      mysql_error());
```

Vedd észre, hogy a kódban az SQL be van építve egy változóba, és a változó van az adatbázis lekérdezési függvénybe, illetve a későbbi hibaüzenetbe táplálva. Ez biztosítja, hogy a hibaüzenet pontosan azt az SQL-t jeleníti meg, mint amit a lekérdezésben használtunk. Meglep, hogy hány professzionális alkalmazás jelenít meg - több hónapnyi tesztelés után is -kézzel begépelt SQL-t a hibaüzenetbe - ami nem ugyanaz az SQL, mint ami a lekérdezésben szerepel.

Hivatkozás:	**oldal:**
Adatbázisok használata session-ökre	161

Az adatbázis-táblázat $country["country"] ["name"] szerint van indexelve, mert az országnév a legpraktikusabb azonosító, és ez az, amit a látogató használni fog az országspecifikus beállítások kiválasztására. A következő SQL-kód létrehoz egy country nevű táblázatot, amelyben a $country minden egyes értékéhez tartozik egy mező, az elsődleges index pedig az országnév. Vedd észre, hogy ahol kétszintű nevek vannak, ott a mezőnevek a tömb kulcsnevei alulvonással vannak összefűzve. így a kód automatizálhatja az adatok bevitelét és kivitelét, ezáltal bármilyen új mezőt adsz a táblázathoz, az automatikusan hozzáadódik a $country-hoz is. Az SQL a kezdő indexet not null-ként definiálja, mert néhány adatbázis, így a MySQL is ezt kívánja:

```
$sql = "create table country"
    . " ("
    . "country_name varchar(60) not null,
    . "characterset_direction enum('ltr',  rtl'),
    . "characterset_name tinytext, "
    . "color bg tinytext, "
    . "color__edge tinytext, "
    . "country_code char(2) not null, "
    . "date_full tinytext, "
    . "date_mysql tinytext, "
    . "date_standard tinytext, "
    . "language__code tinytext, "
    . "language name tinytext, "
    . "number_decimal tinytext, "
    . "number_thousands tinytext, "
    . "time tinytext, "
    . "primary key (country_name) "
    . ")";
if(mysql query($sql)) { print
   ("<br>SQL worked.");

else {
    print("<br>SQL    failed:
    " l                              $sql . ", error: " . mysql_error()
```

Ha létrehoztad az adatbázis-táblázatot, adatra van szükséged. A következő" kód a létező $country-tömbből való automatizált beszúrását mutat meg. Olvasd el a 9. fejezetet, és hozz létre egy adminisztrációs űrlapot, hogy más országok bejegyzéseit is hozzáadhasd, vagy használd a jelen fejezet elején említett egyik adatforrást. (Az SQL be tud olvasni külső fájlt, vagy PHP-val beolvashatsz külső fájlt, így más weblapon levő oldalakat, és az adatokat betáplálhatod SQL-be.)

A kód ciklussal végigfut a $country-tömbön, és az SQL-beli használathoz minden egyes elemét egy name='value'-párrá konvertálja. Ha egy elem tömb, a kód az alsóbb szintű tömbön is végigfut, és a namel_name2 kétrészes nevet használva name/value (név/érték) párt hoz létre. A párok az első kapcsolást kivéve vessző-szóközzel (", ") vannak az SQL-hez kapcsolva. Ezután az SQL-t betápláljuk a lekérdezésbe (a tömbökkel kapcsolatos trükkökró'l a 3. fejezetben olvashatsz):

```
$sql    =    "insert    intő    country
set"; $sep    =    "    ";
reset($country);
while (üst ($kl,      $vl)      =     each ($country)
    ) {
    if(is_array ($vl))
        {
        while (üst ($k2,      $v2)      =
            each($vl>) { if(!is_null($v2))
```

```
            $sql .= $sep     $kl      " . $k2          . $v2 .
            $sep = ", ";


   elseif(!is_null($vl))
       l
      $sql .= $sep . $kl           $vl . ""';
      $sep = ", ";


print("<br>SQL: " . $sql);
if(mysql_query($sql))
    {
    print <"<br>Inserted.");
    }

else
    í
    print("<br>Insert failed with error: " . mysql_error()) ;
    }
```

Mi történik, ha megpróbálod kétszer beszúrni a rekordot? Szúrd be ezt a kódot, hogy hibát produkáljon:

```
if(mysql_query($sql))
    {
    print ("<br>Inserted." );
    }

else
    {
    print("<br>Insert failed with error: " . mysql_error() ) ;
    }
```

íme az eredmény:

```
Insert failed with error: Duplicate entry 'Australia' for key 1
```

Országinformáció visszakeresése

Hogyan keresd vissza az egy országra vonatkozó összes információt? A következő' kód kiválasztja egy adatbázis egy elemét, beszúrja az adatokat a $country-tömbbe, majd megjeleníti őket annak igazolására, hogy a visszakereső kód jól működik.

A kód az Ausztráliára vonatkozó elemet választja ki az **international** nevű adatbázisból. Hozzáadtam az adatbázis-kiválasztást az egy központi adminisztrációs szerverrel vagy webszerverek sorozatával rendelkező honlapokhoz. Az országtáblázatot tárolhatod a központi szerver egyetlen adatbázisában, a központi szerverről eléred az országokra vonatkozó bejegyzéseket, majd tárold az országinformációkat a látogató session-rekordjában. Ez azt jelenti, hogy csak egy országtáblázatot kell fenntartanod, és az I/O tevékenység nagy része a session-adatbázisban zajlik. Ha a session-adatbázison belül az adatokat diszkrét mezőkben tartod, naponta elemezheted az oldalad látogatóinak országonkénti megoszlását:

```
$database["database" ] = "international";
if(mysql_select_db($database["database"]))
    {
    print("<br>Select worked." );

else

    print("<br>SQL failed: " . $sql . ", error: " . mysql_error());
```

Az SQL-select minden mezőt és visszakereső kódot kiválaszt, és az országrekordot egy Srow nevű tömbben hozza a memóriába. A tömbben minden olyan névvel rendelkező mezőhöz tartozik egy elem, amely a $country-ban levő névre lefordítható. A MySQL-függvény egy sort indexelt tömbként, asszociatív tömbként vagy mindkettőként adhat vissza, jelen kódhoz pedig az asszociatív tömb tökéletesen megfelel. Mivel a lekérdezés egy elemet az elsődleges kulccsal választ ki, az csak egy megfelelő elem lehet vagy egy sem; a megkettőződések miatt nem kell aggódnod:

```
$country["country"] ["name"]    =    "Australia" ;
$sql    =    "select    *    from    country    where    country_name='"
    .    $country["country"]["name"]    .
"'"; if($result    =    mysql_query($sql))

    if($row    =    mysql_fetch_assoc($result) )

        print("<br>Fetch worked.");

    else {
        print("<br>Fetch failed:
                                        $sql      error:      mysql_error())

else
    {
    print("<br>SQL failed:       $sql      error:      mysql_error() ) ;
```

A $row-ban az adatok nevenként vannak kulcsozva namel vagy namel_name2 formájában. A következő kód ciklussal végigfut a tömbön, felbontja a neveket, és hozzáadja a megfelelő elemeket a $country-hoz. Az ellenőrzés kedvéért minden egyes elemet kiírat, az üres értékeket pedig null-ra fordítja. Ha meggyőződtél arról, hogy a kód megfelelően működik a honlapodon, csak töröld ki a print-utasításokat:

```
print ( "<tablextr><td><em>Field</em></tdxtd>snbsp;
    .    "<td><em>Value</em></tdx/tr>") ;
while<list<$k,    $v)    =    each($row))
    {
    $n    =    explode("_",    $k);
    if(isset<$n[l]))

        if(is_null($v))

            print("<tr><td>"      $n[0]     "</td><td>" . $n[l]
```

```
                "<td><em>Null</em></tdx/tr>") ;

        else
            {
                                                "</tdxtd>"
            print ("<trxtd>" . \$n[0] . "<td>" . $v .    $n[l]
            $country[$n[0]][$n[l]] = $v;


    else
        {
        if(is_null($v))
            {
            print ("<trxtd>" . $n[0] . "</tdxtd>  .
                "<tdXem>Null</era></tdx/tr>") ;

        else

            print ("<trxtd>"    .    $n[0]    .    "</tdxtd>  </td>"
                .    "<td>"    .    $v    .
            "</tdx/tr>") ; $country($n[0]]    =    $v;


print("</table>");
```

íme az eredmény, az üres mezők is ki vannak íratva. Hasonlítsd össze ezeket az eredményeket a fejezet későbbi részében a $country-ból visszaolvasott értékek eredményeivel:

```
Field                          Value
country          name          Australia
characterset     direction     Null
characterset     name          Null
color            bg            Null
color            edge          Null
country          code          AU
date             full          1 F j, Y
date             mysql         Y-m-d
date             standard      F j, Y
language         code          en
language         name          English
number           decimai       Null
number           thousands     Null
time                           H:i
```

Itt az ido, hogy teszteljük a $country-t. A következő kód beolvassa a $country-tömböt, megjeleníti minden elemét, illetve kibontja őket, amennyiben azok is tömbök:

```
reset (Scountry) ;
print ("<tableXtrxtdXem>Field</emX/tdXtd> </td>"
    .    "<tdXem>Value</emx/tdX/tr>"    )
; while (üst ($kl,    $vl)    =    each
($country) )
    {
    if(is_array($vl))
```

```
        whiledist ($k2, $v2) = each($vl))

           print ("<trxtd>" . $kl                    $k2
                        . $v2 .       </tdxtd>'


    else {
        print ("<trxtd>" . $kl . "</td><td> </td>"
                       . $vl .


  print("</table>");
```

A $country megjelenítésének a következő lett az eredménye. Hasonlítsd ezt össze az *előző* eredménnyel. Az összes mező, ami üresre volt állítva, szépen kitörlődött. Amikor azt akarod tesztelni, hogy a mező bármi másra van-e állítva, mint az alapértelmezett érték, teszteld a mezőt isset()-tel:

```
Field             Value
country  name     Australia
country  code     AU
date     full     1 F j, Y
date     mysql    Y-m-d
date     standard F j, Y
language code     en
language name     English
time              H:i
```

Session-ök használata az országinformációkra

Az országadatok oldalról oldalra való továbbításának legjobb módja a session-ök használata (a session-öket a 19. fejezetben mutatom be). A következő kód használatával regisztrálnod **kell** a **$country-t** a session-kezelővel. Azt javaslom, hogy amíg a látogató nem jelentkezik be, vagy nem kattint egy nyelvspecifikus linkre, az oldalaid addig a böngésző nyelvkódját használják. Ha a látogató először a linket választja, nyiss egy session-t, és használd azt a nyelv továbbítására. Ha a látogató bejelentkezik, indítsd el a session-t, és tárold el a profiljából kapott információkat:

```
if(!session_is_registered("country"))
    {
    session_register("country");
```

Amikor egy session elkezdődik, egy cookie-t tartalmazó HTTP-fejrészt küld ki. A session-azonosító a cookie-ban van. A HTTP-fejrészt az előtt kell elküldeni, hogy a kód bármilyen adatot elküldene, így a szkriptedben egy kis tervezésre van szükség. Ha egy oldalon nem kezdődik session, akkor a **session_register()** kezd egyet, ami azt jelenti, hogy kiküld egy cookie-t. Megváltoztathatod a php.ini-t, hogy az automatikusan kezdjen session-öket;

kezdj session-t manuálisan a szkripted elején, mielőtt bármilyen outputot elküldene; vagy győződj meg arról, hogy a session_register() az első output előtt van. Ha túlságosan későre hagyod a session-kódot, ilyen üzeneteket kaphatsz:

```
Warning: Cannot send session cookie - headers already sent
(Figyelem: Nem tud session cookie-t küldeni - fejlécek már
elküldve)

Warning: Cannot send session cache limiter - headers already
sent (Figyelem: Nem tud session cache korlátozót küldeni -
fejlécek már elküldve)

Warning: open(/tmp\sess_03258d2598ba860e4883a8331ef434c4, O_RDWR)
    failed: No such file or directory (2) (Figyelem: Az
open(/tmp\sess_03258d2598ba860e4883a8331ef434c4, O_RDWR)
    nem működik: nem létezik ilyen fájl vagy könyvtár(2))
```

Üzenet keresése más nyelvben a GNU-gettext-tel

A gettext használatához telepítened kell azt. Ha a PHP Windows bináris verzióját használod Windows NT alatt, akkor csak a következőt kell tenned:

1. Másold a php_gettext.dll-t a c:/Program Files/php/extensions-ből a c:/winnt/system32-be.
2. Másold a gnu_gettext.dll-t a c:/Program Files/php/dlls-ből a c:/winnt/system32-be.
3. Változtasd meg a c:/winnt/php.ini egy sorát ;extension = php_gettext.dll-ből extension=php_gettext.dll-re.
4. Indítsd újra a webszervered (ha a PHP modulként fut).

Megjegyzés: Windows esetén a c:/winnt/system32 helyett c:\windows\system-et használj.

A gettext teszteléséhez szükséged van fordítófájlokra, és mivel a PHP-ban egyetlen egy sincsen, letöltöttem a gettext forrásnyelvű könyvtárát, és a forráscsomag fájljait használtam. Vannak segédprogramok az MO-fájlok szövegformátumú PO-fájlokból való létrehozására. PO-fájlokat bármilyen szövegszerkesztővel létrehozhatsz, utána pedig futtass segédprogramot a PO-fájl MO-fájlra való konvertálására. Vannak GMO-nak nevezett MO-fájlgyűjtemények, és úgy találtam, hogy a gettext-tel való használathoz a GMO-fájlokat egyszerűen át kell nevezned MO-ra.

A PHP Win32 verziójában a tools/gettext nevű könyvtár tartalmazza a fordító segédprogramot. Unix alatt a segédprogramok a gettext-tel együtt telepítődnek. Nem teszteltem a segédprogramokat, mert jobban kedvelem az egyik későbbi Gyors megoldásban bemutatott adatbázis alapú fordítást.

Locale névvel kezdődő könyvtárszerkezetre van szükséged az összes fájlhoz. A tesztoldalaim egy /international nevű könyvtárban vannak, így létrehoztam az /ínternational/locale-könyvtárat. A következő szinten minden olyan nyelvhez könyvtárra van szükséged, ame-

lyeket az oldaladon használsz, és a könyvtár neve a kétbetűs nyelvkód legyen. így létrehoz tam a /locale/de-, /locale/en- és /locale/es-könyvtárakat. Minden egyes nyelven belül szük séged lesz egy LC_MESSAGES nevű könyvtárra, és ebben a könyvtárban ___ .mo nevű fáj lokra, ahol a fájlok neve __ a domained neve.

A domainnév a specifikus alkalmazások listájának azonosítására szolgál. A levelező alkal mazásod összes egyedi szövegét a mail.mo-ban tárolod és a mail domainnevet használod. A következő példa a test domainnevet használja, így az összes nyelv könyvtárában van egy test.mo-fájl.

A példa első lépésben a LÁNG környezeti értékeit de-re állítja. A gettext azt igényli, hogy a locale LC_ALL legyen null-ra állítva, jóllehet, ez rendszerenként és a gettext verzióiként változik. A bindtextdomain() a textdomaint a ./locale-nál kezdődő könyvtárszerkezethez kapcsolja. A textdomain() kiválasztja a domainszöveget, a gettextQ pedig egy sztring for dítását kéri, és az eredményt adja vissza:

```
if(putenv("LANG=de")) { print
    ("<br>putenv  worked.");
    }
else
    {
    print("<br>putenv failed. ");
    } if (setlocale(LC_ALL, "") ===
null)
    {
    print("<br>Language not supported");
    }
print{"<br>Bind  text  domain:  "  .  bindtextdomain("test" ,
"./locale")) ; print("<br>Text  domain:  "  .  textdomain("test"));
print("<p>Test  text:  "  .  gettext("Unknown  system  error"));
```

Az eredmény a következő lett. Vedd észre, hogy a bindtextdomain() az elérési utat a könyvtárszerkezethez és nem a nyelvhez adja vissza. A nyelvet a bindtextdomain()-pa-rancs után is be lehet állítani, de még a gettext()-függvény előtt:

```
putenv worked.
Bind text domain: i:/usr/home/international/locale
Text domain: test
Test text: Unbekannter Systemfehler
```

A domainkiválasztási folyamat lassú, így el van különítve a fordító függvénytől. Ha a fordí tások nagy részét az egyik domainnel hajtod végre, a fennmaradó néhány fordítást pedig egy másikkal, a másik domainre használd a dgettext()-et, ahogy itt látod:

```
print("<br>d  test  text:  "  .  dgettext("test",  "Unknown  system
error"));
```

Az eredmény:

```
d test text: Unbekannter Systemfehler
```

Létezik egy dcgettextQ-parancs is, amelyik kategóriaválasztást ad a dcgettext()-hez, de

amikor ezt a kódot teszteltem, nem állt rendelkezésemre az MO-fájlok kategorizált mintája, így meg kellett elégednem az angolul hagyott eredménnyel. Amikor valami gond van a fájljaiddal, a gettext alapértelmezetten a le nem fordított szöveget adja vissza:

```
print("<br>dc test text: " . dcgettext("test", "Unknown system
error", 6));
```


Szöveg keresése más nyelvekben SQL-lel

Előfordul, hogy szöveges üzeneteket és tartalmat akarsz lefordítani, de egyetlen létező rendszer, - mint amilyen a gettext -, sem felel meg számodra. Ekkor használj SQL alapú adatbázismegoldást. Bármilyen adatbázist használhatsz, a használt kód minden SQL alapú adatbázisra lefordítható. Az alkalmazás az olyan gyors elérésű adatbázisokhoz való, mint a MySQL, és nincs szükség tranzakciókra vagy a PostgreSQL-hez hasonló adatbázisok bármilyen más különleges tulajdonságára.

Az első lépés a nyelvek táblázatának definiálása a következőkben bemutatott SQL használatával. A táblázat duplán indexelt egyfelől a nyelvkódokkal, másfelöl az adott nyelv saját nyelvén, így gyorsan kereshetsz bármelyikkel. Ha úgy döntesz, hogy ezt a táblázatot arra használod, hogy a nyelvmegjelenítő függvényeket beletáplálod, akkor adj hozzá mezőket, amelyek a karakterkészleteket és az egyéb nyelvmegjelenítési jellemzőket tárolják. Az SQL-t közvetlenül beírhatod az olyan adatbázis-adminisztrációs alkalmazásokba, mint a phpMyAdmin, vagy szúrd be azt a szkriptedben egy adatbázisfüggvénybe. Az SQL-t és az adatbázisfüggvényeket az 5. fejezetben magyarázom el:

```
drop table if exists
language; create table
language
    (
    code char(2) not null,
    name varchar(60) not null,
    primary key (code),
    key name (name)
```

Szükséged van egy szöveges hivatkozási táblázatra, a példa az en-táblázatban eltárolt Eng-lish-t használja. Nem számít, hogy milyen alapot használsz a honlapodhoz, feltéve, hogy megnevezel egy megfelelő táblázatot. A *megnevezett* táblázat az a táblázat, amely a fordítás előtt az összes szöveget megkapja, és amely a fordítások során szétosztja a szövegazonosításra használt elemazonosító számot. Mivel a megnevezett táblázat az en, és az elsődleges kulcshoz tartozó bejegyzés auto_increment-re van állítva, a többié viszont nem, ellenőrizni tudod a többi táblázat frissítéseit.

A text-mező korlátlan hosszúságú lehet, így ha akarsz, akár teljes oldalakat is belerakhatsz. A comment-mező egy kisebb szöveges mező, ahova a fordítók rövid megjegyzéseket írhatnak a fordítás pontosságáról. Ha akarsz fordításkezelőt, létrehozhatsz egy párhuzamos táblázatot az átfogó megjegyzéseknek, és egy naplófájt a változtatások nyomon követésére. A translator-bejegyzés a kulcsa egy olyan táblázatnak, amely a szöveget utoljára megváltoztató személyt listázza ki, a date pedig az utolsó változtatás dátuma:

```
drop table if exists en;
```

```
:rea:e  taoie  en
    (
    entry  int(10)  unsigned  default  not null autó increment,
    '0' text  text  not  null,
    translator  int(10)  unsigned  '0' not null,
    default date  timestamp(14), comment
    mediumtext  not  null, primary  key
    (entry), key  text_key  (text(20))
```

Másold át az en-táblázatot de-, es-, illetve minden olyan nyelvre, amelyet használni akarsz. A megnevezett hivatkozási táblázattól eltekintve az **auto_increment** az összes táblázatban el van távolítva a bejegyzési mezőből. Ha a weblapodon a tartalom fele németül van beírva, a rendszernek először akkor is az en-táblázatban kell egy bejegyzést lefoglalni, és az en-be-jegyzés számát kell használni, amikor a de-táblázatba szúr be.

Egy nagy fordítási projektben előfordulhat, hogy nyelvenként több fordító van, és lehetnek olyan fordítók, akik több nyelvre fordítanak. Azonosítanod kell, hogy ki csinálta az egyes fordításokat, és a **translator** nevű táblázat pontosan erre való. Minden fordító egy bejegyzési számot, nevet és e-mailcímet, illetve egyéb szükséges elérési információt kap. A **date**-nek nevezett időjelzés feljegyzi, hogy a bejegyzés mikor volt utoljára frissítve, így ellenőrizheted, hogy az régi e-mailcímek és egyéb adatok még aktuálisak-e:

```
drop  table  if  exists
translator; create  table
translator

    entry  int(10)  unsigned  default  '0'  not
    null, name  tinytext  not  null, email  tinytext
    not  null, date  timestamp(14), primary  key
    (entry)
```

Ha űrlapot akarsz a nyelvtáblázatok frissítésére létrehozni, a 9. fejezetben olvashatsz az űrlapokról. A következő SQL használatával adj hozzá mintabejegyzéseket. A MySQL nulla időjelzést használ a beszúrásokra, a frissítésekhez pedig az aktuális dátumot adja. A fordítás érvényességét a dátumok összehasonlításával ellenőrizheted, és írhatsz szknptet azokhoz a hstaelemekhez, amelyek nincsenek lefordítva, vagy a hivatkozási szövegnél régebbi dátummal vannak lefordítva:

```
insert intő en (entry, text) values ('1', 'memory exhausted');
insert intő de (entry, text) values Cl[1],
[1]virtueller Speicher erschspft');
insert intő en (entry, text) values '2', 'Written by');
insert intő de (entry, text) values '2', 'Geschrieben von[1]);
insert intő en (entry, text) '3', 'untranslated
insert intő (entry, text) values ('3', 'unübersetzte
de values message')
```

A következő kód fordítást talál a **$text-nek,** a **$from** nyelvről a **$to** nyelvre fordítva azt. Az SQL elvégez egy **join-t,** veszi a két táblázat adatait, és egy közös értéken egyezteti a rekor-

dókat, amely a példában **en.entry=de.entry.** Mivel a két táblázat a bejegyzés alapján van indexelve, az összekapcsolás majdnem azonnali. A $from-szöveg kikeresése azért lesz gyors, mert index van a szövegmező első karakterén, és az indexnek van elég karaktere ahhoz, hogy a megfelelő bejegyzést megtalálja. Ha az adatbázis mérete hatalmas, az adatbázis által a szövegmezőből a szövegindexbe másolt karakterek számának módosításával finomíthatod a keresést:

```
$text = "untranslated message";
$from = "en";
$to = "de";
$sql = "select " . $to . ".text from " . $to . ", " . $from
    . " where " . $to . ".entry = " . $from . ".entry"
    . " and " . $from . ".text="' . $text . "'" ;
if($result = mysql_query($sql))
    {
    if($row = mysql_fetch_assoc($result) )

        print("<br>" . $text
                $from $row["text"]
                $to .

    else

        print("<br>Not found: " . $text);


else

    print("<br>SQL failed: $sql mysql_error() ) ;
                                error:
```

Az eredmény az alábbi lett. Annak érdekében, hogy a fordítás bármely nyelven megfelelően jelenjen meg, a megfelelő karakterkészletet kell használnod:

```
en: untranslated message
de: unübersetzte Meldung
```

Ezt a példarendszert használhatod németről angolra való fordításhoz, de elő fog fordulni, hogy az angol fordítás nem lesz a legjobb. A fordítások nem mindig kétirányúak, mivel a fordító a legjobb szavakat választja ki, amelyek megfelelnek az eredeti szó szövegkörnyezetének. A németről angolra való fordítás működik a weboldalakon gyakran használt rövid kifejezések esetében, mert mindenki ugyanúgy használja például a *kattints az egér jobb gombjára* kifejezést, illetve sok angol szónak az eredete megegyezik a német megfelelőjével. A nem azonos eredetű szavak esetében több probléma is előfordulhat. Az muktitut nyelvben több szó is van a *hón.* Lefordíthatod az angol *hó* szót inuktitutra, de visszafele nem végezhetsz el ilyen egyszerű fordítást. Az ilyen jellegű fordítás elvégzéséhez az angol szavak listáját olyan kifejezésekkel kell kiegészítened, mint *a.porhó,jeges hó* és így tovább.

Ez az oka annak, hogy a példa a bejegyzés azonosító száma, nem pedig az aktuális szöveg szerint van kulcsozva. Ugyanarra a szövegre lehet több bejegyzés is. Ha egy pontosabb németről angolra fordító táblázatot szeretnél, hozz létre egy **de_en** nevű táblát, és használd

az olyan szövegek pontos angol fordításának a tárolására, amelyek angolra fordítva nem ugyanazt adják, mint az ellenkező' irányú fordítás. Csak akkor kell az összetettebb kétszeres fordító segédprogramot igénybe venned, ha a tartalom több nyelven érkezik, de még ekkor is jobban jársz, ha a fordítást kézzel végzed oldalról oldalra, és a fordító táblázatokat meghagyod a rövid, gyakran használt üzenetekre.

Nem mindig lehet a lefordított kifejezésekből mondatot alkotni. Ha van három kifejezés, amely a saját nyelveden logikus mondatot alkot, és ezeket egyesével egy másik nyelvre fordítod, a három kifejezést összefűzve nem biztos, hogy az adott nyelv nyelvtani szabályainak megfelelő értelmes mondatot kapsz.

Végül, de egyáltalán nem utolsósorban látogass el a http://petermoulding.com/language-codes.html-oldalra, és keresd ki az inuktitut nyelvet.

A karaktertípusok ellenőrzése

Az ügyfeleid a szöveged minden karakterét meg tudják jeleníteni? Vajon ugyanaz a szöveg egy másik nyelven is megjeleníthető marad? A PHP karakter-ellenőrző függvényei segítenek abban, hogy eldöntsd, egy sztring alkalmas-e az adott feladatokra. A függvények a locale-beállítást használják, így biztosítva azt, hogy az adott országnak megfelelő karaktereket ellenőrzik.

A függvények kipróbálásához adatokra van szükség, és a következő lista megfelel erre a célra. Az első sor csak betűkből áll, a második már számot is tartalmaz, a harmadik központozást, a negyedik a billentyűzet felső sorából véletlenszerűen vett karaktereket, az utolsó sor pedig egy nem nyomtatható karaktert, a chr(5)-t:

```
$text[] = "PHP";
?text[] = "PHP4";
$text[] = "PHP 4.0.7";
$text[] = "PHP %"&*<*_";
$text[] = "PHP %A&*(*_" . chr(5);
```

Mielőtt egy szót elküldesz a helyesírás-ellenőrzőnek, meg akarsz győződni arról, hogy a szó tartalmaz-e a betűkön kívül más karaktereket, mivel az ilyen karakterek általában technikai kifejezéseket, kémiai formulákat, illetve egyéb olyan szöveget jelentenek, amelyek egy alapszótárban nincsenek benne. A ctype_alpha() ellenőrzi a sztring minden karakterét, és hamisat ad eredményül, ha a betűkön kívül más karakter is van a sztnngben. A következő kód ciklussal végigfut az *előző* szöveges tömbön, és minden egyes elemére ráküldi a függvényt:

```
reset ($text);
whiledist ($k, $v) =
                each($text))

    if(ctype_alpha($v))
        {
       print("<br>Passed: " . $v) ;
        }
    else
```

```
i
print <"<br>Failed:        $v) ;
```

Az eredmény azt mutatja, hogy csak az első sor volt megfelelő:

```
Passed: PHP
Faüed: PHP4
Failed: PHP 4.0.7
Failed: PHP %^&* (*
Failed: PHP %"&*(*
```

Ha a helyesírás-ellenőrződ megengedi, hogy számok is legyenek a szavakban (ami gyakran előfordul), akkor az alfabetikus ellenőrzést az alfanumerikussal kell helyettesítened, ahogy azt a következő példa mutatja. A ctype_alnum() igazat ad vissza, ha az összes karakter betű vagy szám:

```
reset($text);
while (üst ($k,      $v)      =
    each($text)) {
    if(ctype_alnum($v))
        {
        print("<br>Passed:      "      .
        $v); }
    else {
    print("<br>Failed:      "      .      $v);
    } }
```

A ctype_alnum()-ellenőrzés eredményei:

```
Passed:  PHP
Passed:  PHP4
Failed:  PHP  4.0.7
Failed:  PHP  %"&*(*_
Failed:  PHP  %^&*(*_
```

A legalapvetőbb ellenőrzés az, ha csak azt nézzük meg, hogy a karakterek kinyomtatható-ak-e. Hiszen csak az ilyeneket érdemes az új PHP print-függvényekben (amelyeket akkor kezdték tesztelni, amikor ezt írtam) használni, mert csak olyan karaktereket akarsz egy oldalon szerepeltetni, amelyet az ügyfél ki tud nyomtatni (például egy űrlapot, amelyet alá kell írnia és visszafaxolni), és egy szövegfordító rendszerben is csak az ilyen karaktereknek van értelme. A ctype_print() igazat ad vissza, ha minden karaktert kinyomtathatónak talál. A következő kód a ctype_print()-et ugyanezen a korábban használt tömbön próbálja ki:

```
reset($text);
while (üst ($k,      $v)      =
                    each($text))

    if(ctype     print($v))  { print
        ("<br>Passed:      "      .      $v) ;
```

```
else

    print("<br>Failed:    "    .    $v);
```

Az eredmények az alábbiak lettek. A **chr(5)** az egyetlen nem nyomtatható karakter.

\
A **ctype_print()** a **íocale** beállítást használja, de nem ismeri az egyedi böngésző vagy nyomtató képességeit. Éppen ezért a függvény esetenként hibázhat:

```
Passed  PHP
Passed  PHP4
Passed  PHP 4  .0.
Passed  PHP %  "&*
Failed  PHP %  ^SJ*(
```

A ctype-sorozat egyéb karaktertípusokat ellenőriz. Ha programoztál már C-ben, fel fogod ezeket ismerni a C-beli megfelelőjükről.

A **ctype_cntrl()** a vezérlőkaraktereket ellenőrzi, így segít az olyan szövegek megjelölésében, amelyeket bizonyos böngészők vagy nyomtatók szokatlan módon fognak megjeleníteni. Ha a szöveged tartalmaz vezérlőkaraktereket, és a szöveget adatbázisban tárolod, *győződ]* meg róla, hogy bináris kompatíbilis mezőtípusokat és függvényeket használsz.

A **ctype_digit()** a numerikus karaktereket ellenőrzi, így biztonságosan teheted a szöveget numerikus függvénybe vagy adatbázismezőbe. A **ctype_xdigit()** olyan karaktersztringet ellenőriz, amelyben hexadecimális számjegyek vannak. Ezt a szöveg hexadecimális mezőben való tárolása vagy számmá való konvertálása előtt használhatod.

A **ctype_lower()** kisbetűs, a **ctype_upper()** pedig nagybetűs szöveget ellenőriz. Ha egy szó nagybetűs karaktereket tartalmaz, lehet betűszó is, te pedig esetleg meg akarod adni a felhasználóknak azt a lehetőséget, hogy a betűszavakat kihagyja a helyesírás-ellenőrző. Ellenőrizheted vele a weboldalad is, hogy kiszűrd azt a zavaró stílust, amikor a szöveget kis- és nagybetűk véletlenszerű váltogatásával írják.

A **ctype_graph()** a szóköz kivételével teszteli a nyomtatható karaktereket, de én még sosem gondoltam arra, hogy bármire is használhatnám. A **ctype_print()** sokkal hasznosabb függvény.

A **ctype_punct()** olyan nyomtatható karaktereket ellenőriz, amelyek nem üres közök **vagy** alfanumerikus karakterek, így megkülönböztetheted azokat a szövegeket, amelyek egyszerű betűtípusokkal megjeleníthetők, és azokat, amelyekhez kiterjedtebb betűtípusok szükségesek. ' H(j^^|

A ctype_space() az üres közöket ellenőrzi, segítségével megtalálhatod azokat a rejtélyes karaktereket, amelyek esetenként szóközként nyomtatódnak ki vagy jelennek meg, esetenként sehogy, és a karakterszámlálót megbolondítják. Ha egy mezőben száz a karakterek maximális száma, de sohasem tudsz ennyi karaktert bevinni a mezőbe, akkor biztos, hogy ***i*** szöveg végén üres közök vannak. (Amikor egy böngésző egy HTML alapú oldalt értelmer.

a böngészőnek meg kell számolnia és szóközként kell megjelenítenie az üres közöket, de a böngészőknek van egy olyan szabályuk, hogy egymás után egynél több szóközt nem jelení tenek meg. Ha csak a szóközöket számolod és nem azt, amit a böngésző szóköznek tekint, a számolásod hibás lesz.)

Kifejezések és helynevek egyeztetése levenshtein()-nel

A Levenshtein-távolság, amit a levenshtein()-függvény ad eredményül, a két kifejezésben különböző karakterek számát jelenti. A metaphone()-t és a soundex()-et erősen befolyásolja a szó kezdőbetűje, ezért nem tartanák a következő példa két helynevét egyezőnek:

```
$textl = "The Great Sandy
Desert"; $text2 = "Great Sandy
Desert";
```

Ez a két sztring jó példa arra a problémára, amellyel akkor kell szembenézned, ha két olyan nevet egyeztetsz, amelyeket különböző formában lehet írni. Gondolj az egyes térképeken *Saint-te*, más térképeken pedig Sí.-tel kezdődő városnevekre. A tesztben a Levenshtein-távolság alacsony az ilyen hosszúságú szöveges sztnngek esetében, míg a soundexQ eredménye teljesen eltérő:

```
print("<br>" $text2));
print("<br>" levenshtein($textl,
print("<br>" soundex($textl));
               soundex($text2));
```

A kód a következő eredményeket adja. A Levenshtein-távolság csupán 4 egy 22 karakter hosszúságú sztring esetén, mely így megfelelő egyezést jelez. A soundexQ-kódokban nin csen hasonlóság, és ezáltal nem használható ilyen típusú egyeztetés esetén, mivel a hang súly nem a szöveg első betűjén van:

T2 63
G632

Hogyan használd a levenshtein()-t arra, hogy fájlból vagy adatbázisból keress egy táblázatot? Definiáld a helynevek, kifejezések vagy országok listáját névmezővel (a sebesség kedvéért indexelve), ahogy a példában látod. A következő SQL create table-utasításból való töredék a mező- és az indexdefiníciót mutatja. Az SQL-t az 5. fejezet mutatja be teljes részletességgel. Hozz. létre egy táblázatot, és rakd bele a mezőt:

```
country_name varchar(60) not null    ^ KEY
countryname (country_name)
```

Amikor az országtáblázatban keresel, először teljes egyezőségre keress. Ha ez nem vezet eredményre, olvasd be az országok listáját egy tömbbe, és keress a tömbben, ahogy itt látod. A példa a $countries-tömbbe rakott elemeket, a $match-ben levő, egyeztetésre váró elemet, illetve egy az egyeztetést végrehajtó ciklust tartalmaz:

```
$countries[] = "Australia";
$countries [ ] = "Saint Kitts and Nevis";
$countries[] = "USA"; $match = "St Kitts
and Nevis"; $best_ld = 9999999;
$best_match = "";
while (üst ($k, $v) = each (Scountries)) {
    Snew ld = levenshtein($raatch, $v);
    if($new_ld < $best_ld) {
        $best_ld = $new_ld;
        $best_match = $v;


print("<br>Best match:
                              $bestmatch);
```

Az eredmény természetesen **Saint Kitts and Nevis.** Hogyan tudod ezt egy adatbázisban végrehajtani? Ha az adatbázisban van **levenshtein()**-függvény, használhatod azt, de ezzel elkötelezed magad egy számolásigényes keresésnek. Ennél jobb megoldás egy fordítótáblázat létrehozása. Ha a teljes egyezőség nem ad eredményt, olvasd be az adatbázis-táblázatot a memóriába, végezd el az itt mutatott keresést, és engedd meg a felhasználónak, hogy ellenőrizze az eredményt. Amint a felhasználó az OK-ra kattint, mentsd el az elgépelt és a helyes nevet egy indextáblában. Amikor egy új keresés nem vezet eredményre az eredeti táblázatban, *végezz* egy teljes egyezőségre való keresést a fordítótáblázatban, így megtudod, hogy a sztringet egy korábbi keresés során megtalálták-e. Ha a fordítótáblázatban van egyező szó, akkor az adatbázisindexen elvégzett néhány I/O-művelettel sikerült eredményre jutnod. Egy idő után a fordítótáblázatban meglesz az összes gyakori változat, elírás és sajtóhiba.

'ejezet

Internet

Gyors megoldások	oldal:
Base64 kódolás	**447**
Aweb böngészése PHP-val	**448**
Linkek ellenőrzése	**451**
FTP-függvények használata	**453**
Curl használata	**460**
Curl-opciók	461

Áttekintés

A böngésződön keresztül a teljes Internet elérhető. Vajon a PHP is biztosítja ugyanezt a szabadságot a weboldalak olvasásának, információk gyűjtésének és fájlok mozgatásának terén? Igen, a PHP mindent el tud olvasni, amit egy böngésző, sőt, információgyűjtő *robot*-ként vagy pókként is működhet, pontosan ugyanúgy, mint a keresőmotorok által használt crawlerek és pókok.

Ebben a fejezetben olyan eszközöket találsz, amelyek révén más weblapokról olvashatsz, szerverek között kommunikálhatsz és webszerverek karbantartására alkalmas eszközök építésébe foghatsz. Ha egy vagy akár több weblapot tartasz karban, ha Internet-szolgáltató vagy és honlapok ezreit tárolják nálad, akkor itt megtalálod a PHP-függvényeket és kódokat, amelyekre szükséged lesz valahol a weblap-adminisztrációs rendszeredben.

Sok feladat elvégzésére több módot is találsz majd. Ha egy nagy projekten dolgozol, szánd rá az időt, és írj össze mindent, ami szükséges a honlapodhoz, majd válaszd a legegyszerűbb megoldást az igényeidhez. Ha mindarra szükséged van, ami említésre kerül ebben a fejezetben, akkor minden bizonnyal a PHP Curl-függvényeit találod a legjobb választásnak, mert az alapjukat képező Curl-könyvtár nagyon sok mindent támogat. Ha a Curl képességeinek csak egy kis része szükséges számodra, akkor az egyszerűbb utat is választhatod, vagyis használhatod a PHP hagyományos hálózati, csatoló és fájlfüggvényeit.

Szintén érdemes fontolóra venni, hogy egyes függvényeket milyen platformokon fogsz használni. Ha teljes platformok közötti rugalmasságot szeretnél elérni, akkor vedd számba az összes célplatformot, és minden függvény használhatóságát teszteld minden platformon. Ugyanez érvényes a nálad tárolt site-okra is - végig kell gondolnod az igényeidet, és kom munikálnod kell az Internet-szolgáltatóddal vagy Internet-szolgáltatóiddal a kérdéses opci ók támogatásáról, a potenciális biztonsági problémákról és az esetleges kieső időről, ha fris síteniük vagy tesztelniük kell a szervereiken, ■ .■ •

URL -

Az URL (Uniform Resorce Locator, Egységes Forrás Meghatározó) az információk közötti kapcsolat, amely World Wide Webbé teszi az Internetet. Az URL-ek teljes leírása megtalálható a www.w3.org/Adressing/-oldalon. Azok, akik megállapítják a szabványokat, az URI (Uniform Resource Identifier, Egységes Forrás Azonosító) egy típusaként hivatkoznak az URL-re, és mind nagyon részletekbe menőek. Neked csak a főbb részeket kell ismerned, mivel a PHP segítségével elemeire bonthatod az URL-t, és az első projekted mindössze néhány összetevőt fog felhasználni. Itt látható egy minta-URL:

```
http://www.google.com/search?q=sun+sand+surf
```

A parse_url()-függvény eredménye a séma (scheme), a gazda (hőst), a kapu (port), az elérési út (path), a lekérdezés (query) és a töredék (fragment). A következő felsorolás a google.com parse_url() eredményeképp kapott URL komponenseit tartalmazza:

```
scheme: http
hőst:   www.google.com
path:   /search
query:  q=sun+sand+surf
```

Séma vagy protokoll

Az URL első, kettőspont előtti része a *séma* vagy más néven *protokoll.* A **http** a megszokott weblap lekérést jelenti. Az **ftp** a File Transfer Protokolra (Fájl Átviteli Protokollra) vonatkozik. Ha egy látogató elfelejti begépelni a protokollt, a legtöbb modern böngésző a HTTP-alapértéket használja. A protokollt több módon is megkaphatod. A **SSERVER_PROTOCOL** a sémát tartalmazza, egy perjelet, és azt követően a protokoll verzióját. A **$HTTP_REFERER** vagy hivatkozó sztring az oldal lekérésére használt URL-t tartalmazza, vagyis az oldalon lévő szkript a hivatkozó sztring elejéről megszerezheti a sémát egy egyszerű kóddal, amely itt látható:

```
$referrer_parts =parse_  url($HTTP_REFERER)
print($referrer_parts   "scheme"]);
```

Host

Az URL második része a *host,* amely a *domaint* is tartalmazza. A host-ot az *előző* részben leírtak szerint kaphatod meg a prase_url()-ből. Ha kézzel kell kinyerned az értéket, akkor a kezdeti :// és az első sima perjel között találod.

A domain egy weblap általános neve, a host pedig a szerver neve a domainen belül. A **www.coriolis.com** egy hasznos weboldal, ha könyvekre van szükséged. A **coriolis.com** a domainnév és a **www** a webszerver neve a domainen belül. A web sötét őskorában egy három webszerverrel, egy mailszerverrel és egy ftp-szerverrel működő weblap a következő host-okkal rendelkezett volna: **www.petermoulding.com, ww2.petermoulding.com, ww3.petermoulding.com, mail.petermoulding.com** és **ftp.petermoulding.com.** Aztán az emberek rájöttek, hogyan tehetnek több webszervert egy webcím mögé, vagyis a lap host-címei a **www.petermouíding.com-ra** csökkentek. Manapság az emberek már nem adnak nevet a szervernek, így a host-nevük **petermoulding.com-ra** redukálódik, ami megegyezik a domainnevükkel.

Elérési út

Az *elérési út* a könyvtárnevek sorozata és az oldal neve, ami egy oldal megtalálásához szükséges. A példaként felhozott **google.com** URL-ben az elérési út egyetlen könyvtárnevet tartalmaz: **/search.** Az elérési útnak fontos szerepe van a PHP-szkriptekben, mert az URL relatív elérési út neveit abszolút szervernevekre kell fordítanod a PHP fájlfüggvényei számára. A **realpath()**-függvény elvégzi a fordítások egy részét, de nem mindet, vagyis hasznos az elérési út kinyerése az URL-ből. Az elérési út az első sima perjel és az utolsó sima perjel között található, feltéve, hogy van az elérési út végén perjel. Az utolsó / és a kérdőjel (?) közötti rész az elérési út része is lehet és az oldal neve is. Ezt gyakran csak az oldal kiterjesztése, például .html, alapján lehet eldönteni, és ha nincs kiterjesztés, akkor az utolsó részt tekintsd az elérési út részét képező könyvtárnévnek.

Oldal

Az *oldal* rendszerint az URL-nek utolsó perjel és a kérdőjel közötti része, és az oldal nevét tartalmazza. Nem szükséges, hogy az URL-nek ez a része valóban egy oldal neve legyen. Az URL-nek ezt a részét a webszerverek, mint az Apache, automatikusan össze tudják párosítani egy könyvtámével, és egy indexet készítenek a könyvtár tartalmához, vagy automatikusan átirányítanak egy speciális oldalra. A példaként említett google.com URL-ben nincs oldalnév. A Google arra hagyatkozik, hogy a webszerver képes a nem teljes URL-eket egy alapértelmezetten beállított oldalra irányítani. Elméletben az URL-t egy záró perjellel kellene begépelni, „http://www.google.corn/search/?q = sun + sand + surf ", de a legtöbb webszerver e perjel nélkül is működik.

A szkriptedhez szükséged lesz az oldal nevére. Az aktuális elérési út/oldal (path/page) kombinációt a $PHP SELF változóból lehet megtudni, az oldalnevet pedig a basename($PHP_SELF)-ből, mint itt látható:

```
print("<br>".$PHP_SELF);
print ("<br>basename <) : " .basename ($PHP__SELF) ) ;
```

Az eredmény:

```
/phpblackbook/internet/url.html
basename():url.html
```

Töredék

A *töredék* kiterjesztés egy oldalhoz, amelynek révén az oldal egy meghatározott részét lehetd elérni. Ha az xyz.html-oldal egy name=abc paraméterrel ellátott <a> anchor tag-et tar maz, akkor az xyz.html#abc-lekérésre a böngésző csak az anchor-tag utáni részét jeleníti meg az oldalnak. A töredékek nem jelentek meg a PHP-ban, mikor Netscape 4.76-tal tesz teltem. A legjobb magyarázat szerintem az, hogy a böngésző elküldés előtt kivonja a töredéket, és csak akkor használja, mikor megkapta az oldalt. Azonban nem minden böngésző I működik így, mivel a szerveremen néhány URL-lekérésben találtam töredékeket.

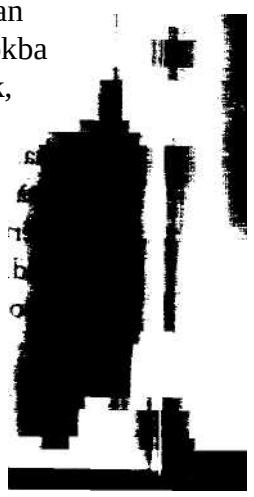
Lekérdezés

K.1V

Az URL *lekérdezés* része mind, ami a kérdőjel után van. A PHP a lekérdezéssztringet a SQUERY_STRING nevű változóba teszi. A google.com-példánál maradva a lekérdezés a q=sun+sand+surf. Az egyenlőségjel egy név=érték párosítást jelöl, a q a lekérdezés ni a sun+sand+surf pedig az értéke. A pluszjel a szóközöket képviseli. Ha több név/érték pár szerepel, akkor egy "&"-szimbólummal vannak összekötve. A PHP-ban ritkán van szükség a lekérdezéssztring ismeretére, mert a PHP a név/érték párokat PHP-változókba fejti vissza. A google.com-példában a lekérdezést feldolgozó szkripted úgy kezdődik, mintha a PHP a következő programsort generálta volna:

```
$q    ="sun    sand    surf";
```

JÜ IZ3TJ15 XI;



Különleges karakterek

Ha különleges karakterek vannak a lekérdezésed név/érték párjának érték részében, akkor az urlencodeQ-függvénnyel tudod kódolni azokat. A kódolás minden különleges karaktert egy háromkarakteres ábrázolással cserél le, amely egy százalékjelet (%) és a karakter hexadecimális kódját tartalmazza. Itt látható egy példa:

```
$x =urlencode("special characters liké sand ="); .*?
print("<br>".$x);
print("<br>".urldecode($x) ) ;
```

Az eredmény:

```
special+characters+like+%2 6+and+%3D
special characters üke & and =
```

A PHP a szkript indításakor megadja a változók értékeit, és ekkor automatikusan kikódolja a bekódolt URL-t. Az URL bekódolása a cookie-khoz is használatos, amit a PHP automatikusan kikódol, mikor a cookie-kat beolvassa a memóriába. Ha hosseú bináris sztringet teszel az URL-be, akkor a base64 kódolás rövidebb sztringet eredményez, mint az uríencode(). Azonban a base64-et a PHP nem kódolja ki automatikusan és az URL-kódolás sokkal hatékonyabb hosszú, hagyományos sztringek esetében, amelyek csak néhány különleges karaktert tartalmaznak.

A PHP-ban van egy rawurlencode()- és egy rawurldecode()-függvény is, amelyek az urlencode()- és az urldecode()-függvényekhez hasonlóan átalakítják a speciális karaktereket, de a rawurlencodeQ a szóközöket nem %20 formába alakítja, hanem egy + jelet illeszt a helyükre.

Base64 kódolású szöveges sztringek

A kódolás szükségességének magyarázata az, hogy a számítógépek és a hálózati eszközök bizonyos karaktereket vezérlő sztringként és egyéb különleges feladatokra használnak. Ezért nem használhatod az egy bájt 8 bitjével ábrázolható 256 db karakter mindegyikét. Ez azt jelenti, hogy a 256 karaktert egy másik formátumba kell konvertálnod.

A base64 kódolás elméleti alapja az, hogy a 8 bit 256 kombinációját 6 bit 64 kombinációja- . val is le lehet írni, ha három 8-as csoportot összevonsz 24 bitté, és a 24 bitet négy darab 6-os csoportra osztod. A base64 kódolás karaktertáblája az A-tól Z-ig és az a-tól z-ig terjedő" betűket, a 0-tól 9-ig terjedő számokat és a + valamint / speciális karaktereket tartalmazza, ezért a base64 kódolású adat nyomtatható, és bármilyen közegen, például e-mailen keresztül megbízhatóan továbbítható.

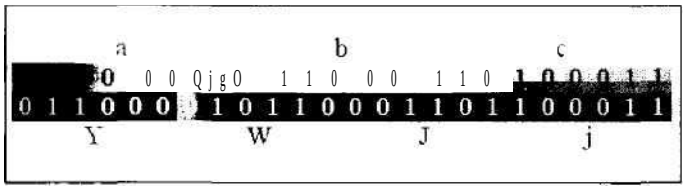
A 13.1 ábra az elmélet alkalmazását mutatja *az. abc-sztringre.* A sztringet bináris formátumba alakítjuk, majd hatos csoportokra osztjuk. Minden hatos csoportot egy olvasható karakterré alakítunk. A hatos csoportokat ábrázoló karakterek nem követik egymást az ASCII -karaktertáblában, vagyis a végeredményhez egy fordítás szükséges.

Az input-adatok nem feltétlenül szép szabályos hárombájtos csomagokban érkeznek, ezért a base64 kódolásban szerepelnie kell az utolsó karakterek kiegészítésének. A következő

fc.

példa bal oldalán az *aaa, aa* és *a* karaktersztringek láthatók. A jobb oldalon vannak a base64-ben kódolt adatok, az első az YWFh, amely az *aaa* sztring kódolva. A második az YWE, ami az *aa-i* kódolja, és a záró = azt jelöli, hogy csak két bájt volt az input-sztringben. A harmadik példa az YQ, az *a* kódolása, ahol a két záró = azt jelöli, hogy az input-sztring csak egy karaktert tartalmazott.

aaa YWFh
aa YWE=
a
YQ==



13.1 ábra A Base64 kódolás 8 bitesről 6 bitesre konvertál

T f tk-b

Mostanra minden bizonnyal már a base64 kódolás lehetséges felhasználásain gondolkodsz, beleértve a bináris adatok továbbítását URL-en keresztül. Már fel is ismerted, milyen problémát jelent az = és a + jel az URL-ben, ahol ezek a karakterek speciális jelentéssel bírnak. A válasz az URL-kódolás. A base64 kódolású sztringet egy urlencode()-függvénybe illeszted. (Az URL-kódoló függvényt base64 kódolás nélkül is használhatod, de három input-karakterből kilenc output-karaktert kapsz, ami túlságosan költséges a bináris adatok számára.) A következő sorban az *aa* dupla kódolását láthatod. Figyeld meg, hogy az = jelből %3D lett:

YWE%3D

Más szerverek böngészése

Hogyan derítik ki a keresőmotorok, hogy mi van az oldalaidon? Úgy böngészik az oldalaidat, mint egy kíváncsi látogató, aki minden linkre és képre rákattint, hogy megnézze, mi történik. Te is böngészhetsz a neten hasonló technikával; „A web böngészése PHP-val" című Gyors megoldásban egy példát is találsz a fejezet későbbi részében.

Mikor a keresőmotorok az oldalaidon böngésznek, egy böngészőprogramot emulálnak, hasonló módon kikódolva a HTML- és egyéb tageket. Miután minden tagét kikódoltak, az információkat lementik és indexelik, vagy eldobják. Ha az egyik tag egy másik oldalra vagy másik honlapra mutató linket tartalmaz, akkor a linket ütemezik egy későbbi böngészésre. Ha az honlapod egy más honlapokra mutató portál, akkor ugyanezzel a technikával indexelheted más honlapok megfelelő oldalait. Számos nyílt forráskódú keresőmotor-projekt létezik, és valamelyik biztosan megfelel az igényeidnek.

Ha mindössze arról szeretnél meggyőződni, hogy még működnek a linkek, akkor lekérhetsz egy oldalt, vagy információkat kérhetsz le egy oldalról, ezáltal sokkal kevesebb hálózati forgalommal ellenőrizheted a linket.

A PHP csatolófüggvényei, a Curl-, az fopoensock()-, a hagyományos fájlkezelő függvények mind el tudnak olvasni egy fájlt egy URL-ről, vagyis számos lehetőséged van. A választásodat az igényeid bonyolultsága és a jövőbeli fejlesztésekkel szemben megkívánt rugalmasság fogja meghatározni.

Csatolófüggvények

A csatolófüggvények lehetővé teszik, hogy PHP-ban megírd a saját webszerveredet vagy böngésződet. Egyiket sem ajánlanám első gyakorló PHP-projekt gyanánt, de vannak olyan esetek, mikor szükséged lehet egy webszerver vagy egy böngésző bizonyos funkcióira, vagyis megéri ezekről az utasításokról olvasni.

Képzeld el, hogy egy Internet-szolgáltató vagy, számos virtuális weblap fut a szervereden, és most tervezel bevezetni a fogyasztóidnak (a virtuális weblapok tulajdonosainak) egy reklámelhelyezési szolgáltatást egy külső ügynökségen keresztül. Az ügynökségnek van egy szervere, amely széles körben kínál bannereket, amelyek megjelenése után jutalékot fizet. A bannerek típus szerint vannak csoportosítva, vagyis a sport témájú oldalak sporttal kapcsolatos hirdetőket választhatnak, és az iskolás gyerekeknek szóló oldalak elkerülhetik a csak felnőtteknek szóló lapok hirdetéseit.

A fogyasztóid különböző banner-hirdetéseket helyezhetnek el az oldalaikon, ezáltal a weblapjuk bevételt generál, míg te minden elhelyezésért jutalékot kapsz. A fogyasztóidat összekapcsolhatod az ügynökség szerverével, megállapodsz az ügynökséggel, hogy fizesse a jutalékodat, és nem foglalkozol közvetlenül a fogyasztókkal. Egy másik megoldás ha reprodukálod az ügynökség oldalait a saját honlapodon, a fogyasztóid továbbra is a saját oldalaidat látogatják, és a PHP csatolófüggvényeivel továbbítod az igényeiket az ügynökség felé. A fogyasztók veled kötnek üzletet, nem az ügynökséggel. Az ügynökség minden tevékenységet a te azonosítód alatt lát, és soha nem tudja meg a fogyasztóid elérési adatait. Egyetlen hatalmas jutalékösszeget kapsz, és továbbítod a pénzt a fogyasztóknak, vagyis levonod az oldal elhelyezésének díjából.

A csatolók PHP-be történő telepítéséhez Unix alatt az —enable-sockets-paraméterrel konfiguráld a PHP-t, mielőtt lefordítod. Windows NT-n (vagy Windows-on) egyszerűen másold be a c:\Program files\php\extensions\php_sockets.dll-fájlt a c:\winnt\system32 könyvtárba (vagy Windows-on a c:\windows\system-konyvtarba), majd add hozzá a PHPini-hez a extension = php_sockets.dll sort. Ha a PHP-t nem CGI-ként futtatod, akkor indítsd újra a webszerveredet. Ne feledd, hogy a PHP csatoló támogatása még mindig kísérleti jellegű, de ha nem volna is az, akkor is biztonsági és tűzfalproblémákat vet fel.

A kiterjesztett osztálymodulok php.ini-ben történő betöltésének alternatívája a dinamikus betöltés a dl()-függvénnyel. A következő kód megmutatja, hogyan használd a dl()-t, és mielőtt betölti a modult, ellenőrzi, hogy nincs-e már betöltve. A PHP a php.ini-ben lévő extension_dir alapján találja meg a betöltendő modulokat, vagyis a dl()-függvényben szereplő fájlnévhez tartozó elérési utat hozzáadhatod az extension_dir-hez. Unix alatt a modulfájlok kiterjesztése .dll helyett .so:

```
if (extension_loaded ("sockets") )
    {
    print("<br>Sockets    loaded.");
    }
else
    {
    print("<br>Sockets not loaded.");
    dlCphp_sockets.dll");
```

Mindössze egy probléma van. Windows NT-t és Apache 1.3.20-at használva a következő hibaüzenetet kaptam. Talán ha majd az Apache 2 elterjed és a PHP kihasználja az Apache 2 fejlettebb többszálas támogatását, akkor lehetővé válik a dinamikus betöltés. A Windows NT mind a többszálas feldolgozást, mind a multitaskinget támogatja, de bizonyos operációs rendszerekben nincs multitasking, csak többszálas fedolgozás, ezért nem használhatják a PHP és az Apache multitaskinget igénylő funkcióit:

```
Fatál error: dl() is not supported in multithreaded servers - use
    Web extension statements in your php.ini
```

A socket()-függvénynek egy domaint, egy típusparamétert és egy protokollt lehet megadni, és a csatoló erőforrás-azonosítóját adja eredményül. A domain nem az elérni kívánt domain neve, hanem vagy **AF_INET,** vagy **AF_UNIX.** A típusparaméter a következők valamelyike: **SOCK_STREAM, SOCKDGRAM, SOCK_SEQPACKET, SOCK_RAW, SOCK RDM** vagy **SOCK_PACKET:**

```
$sock    =socket(AF-INET,SOCK^STREAM,0);                    J
```

A bind()-függvény az AF_INET-paraméterrel megadott csatoló esetén egy Internet-címet kapcsol a csatolóhoz; AF_UNIX-szal definiált csatoló esetén a csatolót egy Unix domain-csatoló elérési útjához kapcsolja. Az első paraméter a socket()-ból származó azonosító. A második az IP-cím vagy az elérési út. **AF_INET** típusú csatoló esetén egy harmadik opcionális paraméterben lehet megadni a portszámot. A **bind()** siker esetén nullát ad vissza, hiba esetén egy negatív hibakódot. A **strerror()**-függvény értelmes üzenetté fordítja a hibakódot: `$result =bind($sock,"127.0.0.1");`

A **strerrorQ** egy csatolófüggvénytől származó hibakód magyarázatát adja meg. A következő kód a strerror() használatát mutatja az imént bemutatott bind()-dal:

```
if($result
    !=0) {
    print("<br>".strerror(Sresult));
```

A HstenQ a bind()-függvénnyel egy címhez kapcsolt csatolón beérkező kapcsolódásoka: ~ gyeli. Az első paramétere a csatolóazonosító, a második a várakoztatott beérkező kapcsc dások maximális száma. A **listenQ** 0-t ad siker esetén, hiba esetén pedig egy negatív hibakódot. A hibakód betáplálható a **strerror**()-függvénybe, mint a következő példában **látha:**

```
$heard = listen($sock, 20);
if($heard != 0)

    print("<br>" . strerror($heard));
```

--w,

Az accept_connect() paramétere a bind()- és a listen()-függvényekben használt csatoló-azonosítója. A függvény a csatolóazonosító alapján elfogad egy bejövő kapcsqlódási kérést. Az accept_connect() egy új csatolóazonosítót ad vissza, amely az éppen elfogadott kapcsolathoz tartozik. Az eredeti azonosító további fogadások céljára nyitva marad. Ha már befejezted ennek a kapcsolatnak a használatát, akkor be kell zárnod. Ha az összes kapcsolattal végeztél, akkor az eredeti csatolót is be kell zárnod:

```
$accepted = accept_connect($sock) ;
if($accepted != 0)

    print("<br>" . strerror($accepted) ) ;
```

A connectQ argumentumában egy socketQ-függvénnyel létrehozott csatolóazonosítója, egy IP-cím és az AF_INET típusú csatolók számára egy opcionális portszám szerepel. Ha a csatoló AF_UNIX típusú, akkor a második paraméter a Unix domain csatoló elérési útja. A connect() siker esetén nullát ad vissza, hiba esetén pedig egy negatív hibakódot, lásd a következő kódot:

```
íresult = connect($sock,
"127.0.0.1"); if($result != 0)
    {
    print("<br>" . strerror($result));
```

A read() argumentumában egy csatolóazonosító, a puffer neve és a maximális olvasási hossz szerepel, visszatérési értéke pedig a beolvasott bájtok száma. Egy opcionális negyedik paraméterrel meghatározhatod az olvasás típusát. Jelenleg a következő opciók elérhetők a negyedik paraméterhez:

- PHP_SYSTEM_READ - A rendszer read()-parancsát használja.
- PHP_BINARY_READ - Bináris-biztos read()-et használ.
- PHP_NORMAL_READ - Az alapbeállítás, ahol az olvasás megáll \n-nél vagy \r-nél.

A következő kód maximum 1000 bájtig olvas a $sock-ból a $data-ba, és a beolvasott bájtok számát adja vissza, vagy hamisat, ha az olvasás kudarcot vall:

```
$data ="";
$read =read($sock,$data,1000) ;
print <"<br>Bytes read:".$read);
```

A write()-függvény paraméterei a csatolóazonosító, az adatot tartalmazó puffer neve és a beírandó hossz:

iS-...J3YÖii JS3
fi ajifíl -7S A v

```
$data = "The cow jumped over the moon"
    . " while the dish and spoon learnt
HTML"; $result = write($sock, $data,
strlen($data)); print("<br>Write result: " .
$result);
```

A socket_get_status()-függvény a connect()-tel használt csatoló státusát adja eredményül, de az accept_connect()-tel létrehozott csatolókra megjósolhatatlan eredményt ad. A függvény paramétere a csatolóazonosító és egy "timed_out"- (időtúllépés-), "blocked"- (blokkolva-), "eof"- (fájl vége-) és "unreadjbytes"- (beolvasatlan bájtok-) elemet tartalmazó asszociatív tömböt ad vissza. Az első három mező igaz/hamis, a negyedik egész típusú. A következő kód egy connect()-tel használt csatoló állapotát teszteli, és a teszt eredményét egy tömbként megjeleníti. További kódok hozzáadásával tesztelheted az időtúllépést, a blokkolt portokat, és megfelelő üzeneteket készíthetsz, illetve ciklussal futtathatsz még egy readQ- vagy writeQ-parancsot:

```
$status = socket_get_status($sock)
; while (üst ($k, $v) = each
($status ) )
    print("<br>" $k $v) ;
```

A close() bezárja az azonosítóval megadott csatolót. Az azonosító a socket()- vagy az accept_connect()-függvényekből származik. A close() igazat eredményez, ha sikeresen lezárta a csatolót, és hamisat ad vissza, ha az azonosító érvénytelen:

```
if ( ! close($sock)) {print("<br>Socket close
    failed!");
```

SNMP

A www.ibr.cs.tu-bs.de/ietf/snmpv3/-oldalon rendkívüli mennyiségben találsz az SNMP-ről szóló dokumentumokat és az SNMP-vel foglalkozó vállalatokat (Simple Network Management Protocol). Legegyszerűbb módon kifejezve az SNMP lehetővé teszi, hogy a hálózatod bármely intelligens eszközével kommunikálj, akár hardverről, akár szoftverről van szó. Megkérdezheted a routert, hogy van-e elég memóriája, és betervezhetsz egy bővítést, ha éppen kifogyóban van.

A hálózatodon sok intelligens eszköz képes lehet arra, hogy figyelmeztetést küldjön, ha valamelyik erőforrás elapadóban van, de a tűréshatárok beállítása nem mindig megfelelő. N:~ sok haszna van egy hibaüzenetnek a rendszer összeomlása előtt 30 perccel, ha a megrendelés, szállítás és telepítés 30 órát vesz igénybe.

Miért használnál PHP-t az SNMP böngészésére, mikor oly sok szoftver elérhető a hálózat állapotának gazdag grafikus felületen történő megjelenítésére? Képzeld el, hogy megérkezel a Maldív-szigetekre vagy Ausztráliában a Great Barrier-zátonyhoz egy néhány napos könnyűbúvár vakációra. Jól tudod, hogy ha teli palackkal maximális mélységbe merülsz, akkor ezt követően legalább egy napig nem pattanhatsz repülőre, hogy visszarepülj az ir

-dába. Nagy magasságban az alacsony légnyomás miatt úgy pezsegne a véred, mint egy cc -

boz meleg kóla. Nem sok könnyűbúvár boltnak van közvetlen kapcsolata a vállalati hálóza todhoz - maximum egy szimpla Internet-kapcsolatra számíthatsz. Más nem is szükséges ahhoz, hogy elérd a PHP-t a szervereden, és ellenőrizd a hálózatod állapotát, mielőtt még 24 órára alámerülnél a paradicsomban. 1

Mielőtt hosszabb időre eltávoznál, írj néhány PHP alapú oldalt az SNMP-eszközök böngészésére, és adj hozzá SSL-t, valamint egy rendelkezésre álló bejelentkezést. Ezek után a világ bármelyik Internet-kávézójából karbantarthatod a hálózatodat.

Ötlet: Ha valóban eljutsz Ausztráliába a Great Barrier-zátonyhoz könnyűbúvárkodni, akkor megérkezés után az első dolgod az legyen, hogy küldesz egy e-mailt a főnöködnek, amelyben megmagyarázod, hogy Ausztráliába még nem ért el az Internet.

Az SNMP elérhető Unixon, Windows NT-n, Windows 2000-en, de Windows 98-on nem. Követned kell az SNMP telepítési útmutatóját, és szükséged lesz a hálózatod SNMP-kompatíbihs hálózati eszközeinek dokumentációjára. Győződj meg róla, hogy a dokumentációkat egy böngészővel is eléred, így az SNMP-információkkal párhuzamosan használhatod őket. A PHP-függvények a korábban ucd-snmp-nek nevezett net-snmp-t, használják, amely a http://net-snmp.sourceforge.net/-oldalrol tölthető le. A PHP-függvényekhez a net-snmp NO_ZEROLENGTH_COMMUNITY opcióját l-re állítsd, mielőtt fordítod.

Ha vannak nem SNMP-kompatíbilis eszközök a hálózatodon, akkor előfordulhat, hogy nem tudod elejétől végéig követni a problémát. Egy jól kialakított hálózatban az olcsóbb, nemintelligens, nemkarbantartható eszközök nagyon kilógnak, és a kulcsfontosságú gerinchálózatokat teljes mértékben SNMP-vel tartják karban; tehát a legdrágább részeket karbantarthatod a hálózaton keresztül.

Az snmpwalk()-függvény az SNMP használatának kiindulópontja. A függvény paraméterei egy host-név vagy IP-cím, egy csoportnév, egy objektumazonosító, egy opcionális időkorlát-paraméter és egy opcionális újrapróbálkozások száma. A függvény azokat az SNMP-objektumokat adja vissza egy tömbben, amelyek elérhetőek a megadott objektumon keresztül. A következő kód végigsétál a net-snmp fejlesztői által biztosított tesztszerver objektumain. A közös név az, amelyet a nyilvános hozzáférés számára meghatároztak. Az objektumazonosítót egy nulla hosszúságú sztringnek hagytuk meg, hogy az objektum információ-struktúrájának legfelső szintjén kezdje a sétát:

```
$host = "ucd-snmp.ucdavis.edu"; $objects =
snmpwalk($host, "demopublic", if ($objects)
{
   while (üst ($k, $v) = each ($objects) ) {
       print("<br>" . $v );
```

ü
!o
íébom
A

```
else

    print ("<br>snmpwalk error." );
```

A következőkben a az output első három sora látható. A demo-rendszerből származó output nagy része nem jelent semmit számomra, mert nem egy Cisco-routerről vagy valami hasonló eszközről szól. Az output több sora sokszor ismétlődik. Ismerned kell a lekérdezett eszközt és a hozzá tartozó normális értékeket, hogy lásd a változást:

```
"HP-UX ucd-snmp B.10.20 A 9000/715"
OID:.iso.3.6.1.4.1.2021.250.6
Timeticks: (1700055755)19 days,16:22:37.55
```

Az snmpget()-függvény paraméterei egy host-név vagy IP-cím, egy csoportnév, egy objektumazonosító, egy opcionális időkorlát-paraméter és egy opcionális újrapróbálkozások száma. A függvény egy SNMP-objektumot eredményez. Az azonosítót az snmpwalk() által visszaadott listáról is megállapíthatod. Az objektum tartalma az objektum információit szolgáltató eszköztől függ. Az snmpwalkQ-függvénnyel mindent láthatsz az eszközről, míg az **snmpget()-tel** gyorsan hozzáférhetsz egy bizonyos információhoz, vagyis a vizsgálataid jellegének megfelelően alkalmazhatod őket. A 17. fejezet leírja az objektumokat és azt, hogy miképp böngészheted őket.

Az **snmpset()**-függvény paraméterei egy host-név vagy IP-cím, egy csoportnév, egy objektum-azonosító, az érték típusa, az érték, egy opcionális időkorlát-paraméter és egy opcionális újrapróbálkozások száma. A függvény értéket ad meg az SNMP-objektumnak. A típus paraméter lehet **i** az egész számokhoz, **s** a sztringekhez és egyéb opciók is lehetségesek az objektumtól függően.

Az **snmpwalkoidQ** az **snmwalkQ** egy régebbi verziója, de semmilyen hasznos különbséget nem látok.

Az **snmp_get_quick_print()** a net-snmp quick_pnnt aktuális beállítását adja meg (Unix alatt), és az snmp_set_quick_print{) megváltoztatja ezt az értéket. A látható különbség az **snmpgetQ-** és az **snmpwalk()**-függvények eredményének formázásában jelentkezik; az alapbeállítás a verbose (beszédes), ahol az értékek magyarázatokba illesztve jelennek meg, míg **a** quick print- (gyors) megjelenítés hatására a függvények nyers adatokat adnak vissza.

Hogyan tudod igazán hasznát venni az SNMP-nek a Cairns Internet Caféból (a Great Barriers-zátony szélén)? Felállítasz egy adatbázist. Az első táblában felsorolod az IP-címeket, az eszköztípusokat és az összes eszköz SNMP-kompatíbilitását, amelynek bármi köze lehet a hálózatodhoz (az archiváló és a tartalék eszközöket is beleértve). A második táblában felsorolod az össze lehetséges tulajdonság és érték típusleírását, eszközök és azon belül objektumazonosító szerint indexelve. A harmadik táblázat az értékeket tartalmazza és eszközök, objektumazonosítók, dátum és idő szerint vannak indexelve. Az eszközöket gyártó, modell, leltári vagy sorozatszám és IP-cím alapján kell azonosítanod, vagyis követhetsz egy elemet, ha változik az IP-címe, vagy összehasonlíthatsz eszközöket, ha felváltva ugyanarra a hálózati helyre csatlakoztatták őket. Valamint írhatsz olyan oldalakat, amelyek egy adott eszköz jelen pillanatbeli értékeit hasonlítják össze mondjuk az egy héttel ezelőttivel.

Rendszeresen le kell kérdezned az eszközöket, hogy megállapíthasd a normális működési értékeiket. Az eszköz képernyőjét az eszközhöz igazíthatod, és az egyes értékekhez ma-

gyarázatokat fűzhetsz. Minden eszközhöz készíthetsz egy listát, amely a hozzá kapcsolódó eszközöket tartalmazza, vagyis láthatod egy switch-hez kapcsolódó összes hub-ot.

Ez a projekt igazi móka, és valószínűleg a főnököd még fizetni is fog érte, ha naprakész információkkal látja el a szervezet eszközeiről. Éppen ezért szükséged lesz egy olyan oldalra, amely felsorolja a használaton kívüli eszközöket és az okot, amely miatt használaton kívül vannak (elkezdett füstölni/kigyulladt/még mindig ég). Ha a főnök képeket szeretne, van egy aranyos kis nyílt forráskódú diagram-alkalmazás, a Dia (www.lysator.liu.se/~alla/dia), amellyel PNG-formátumban mentheted el a diagramokat és a PHP-val megjelenítheted őket az eszközök tulajdonságait felsoroló oldalon. (A Dia Unixon működik, de a Windows 32 bites bináris verzió is elérhető a http://hans.breuer.org/dia/-oldalon. Én Gimpet, [http://gimp.org] majd Diát telepítettem a Windows NT-munkaállomásomra, és mindkettő gyönyörűen dolgozott.)

CUrl ..Ar$_S$-.^..c,,-,-:c.„..>.

A Curl mindent megtesz, csak éppen cappuccinót nem készít. A Curl weblapját idézve:

> *A Curl egy olyan eszköz, amely alkalmas fájlok átvitelére URL-szintaxissal, támogatja az FTP-t, az FTPS-t, a HTTP-t, a HTTPS-t, a GOPHER-t, a TELNET-et, a DICT-et, a FILE-t és az LDAP-t. A Curl támogatja a HTTPS-tanúsítványokat, a HTTP POST-ot, a HTTP PUT-ot, az FTP-feltöltést, a kerberos-t, a HTTP űrlap alapú feltöltést, a proxy-kat, a cookie-kat, a felhasználó +jelszó hitelesítést, a fájlátvitel-folytatást, a http proxy tunnelinget és még egy rakás hasznos trükköt.*

A Curl alapját képező könyvtár a http://curl.haxx.se/-oldalon elérhető. Ahhoz, hogy Unix alatt használd a Curl-t, a PHP-t a —with-curl[= DIR]-paraméterrel fordítsd be *(zDIR* arra a könyvtárra mutasson, ahová letöltötted a Curl-könyvtárat). Windows NT vagy Windows 2000 alatt másold be a php\dlls\Libeay32.dll-, a php\dlls\Ssleay32.dll-, és a php\extensions\php_curl.dll-fájlokat a c:\winnt\system32-konyvtarba. Windows 98 alatt ugyanezt kell tenned, annyi különbséggel, hogy a c:\winnt\system32-konyvtarat a c:\windows\system-konyvtarra cseréled. 1

A curl_init() egy Curl-sessiont kezdeményez, a curl_setopt() beállítja az aktuális session-opcióit, a curl_exec() végrehajt egy műveletet, a curl_close() pedig lezárja a session-t. A legújabb funkciók használatához egy friss verzióra lesz szükséged, és a verziószámot a curl_version()-függvénnyel vizsgálhatod meg.

FTP

Valószínűleg te is használsz FTP-t a szervereken lévő fájlok elérésére, vagyis tudod, hogy ez egy egyszerű módja az alapvető fájlátvitelnek. Az FTP formális leírása a www.normos.org/en/summaries/ietf/rfc/rfc959.html-oldalon található meg. Az FTP bizonyos részei, különösen a biztonsági megoldás és a site-parancs olyan kiterjesztések, amelyek a távoli szerver operációs rendszerétől és FTP-szoftverétől függnek. ; .:-"S?

Az FTP-programok rendszerint nem igazán kifinomultak, vagyis egy FTP-program felvál tása egy PHP-szkripttel nagyobb ellenőrzést és több lehetőséget biztosít a fájlok saját logi kád szerinti kiválasztásában. Gyakran továbbítok fájlokat Windows NT és Unix-szerverek között, ezért szeretnék egy olyan FTP-programot, amelynek megadhatom a kizárandó fáj lok listáját és a fájlok átnevezésének szabályait. j

Mielőtt a PHP FTP-függvényeibe vetnéd magad, vedd fontolóra rmás függvények használa-tát, amelyek elfogadnak FTP-lekéréseket, például az fopen(). Az fopen() elfogadja az ftp://-előtaggal ellátott fájlneveket, ami lehetővé teszi, hogy fájlokat olvass és írj távoli gépekről távoli gépekre. Az fopen() különösen értékes lehet, ha szeretnéd ellenőrizni vagy átalakítani a fájlok tartalmát az átvitelközben. Például a \r\n Windows stílusú sorvégeket vagy a \n\r Macintosh stílusú sorvégeket lecserélheted \n Unix stílusú sorvégekre. Az FTP-nek kézenfekvő könyvtárkezelő függvényei vannak, amelyek révén az FTP-vel könnyedén áttekintheted a távoli szerverek könyvtárszerkezetét.

Gyors megoldások

Base64 kódolás

Ha szeretnél saját levelezőrendszert írni, akkor a csatolt fájlokhoz szükséged lesz a base64 kódolásra, hasonlóan a hírcsoporthoz csatolt állományokhoz. A base64-et arra is használhatod, hogy bináris adatokat kódolj olyan adatbázisban történő eltárolás céljából, amely nem rendelkezik bináris sztring mezővel **(blobs** vagy Binary Large Object). A base64 be- és kikódolást szemléltetendő, gépeld be a következő adatokat a szkriptedbe:

```
$data    []    =="Hello   Jennifer";
$data    []    =="Here   are   funny   characters:? + &'\"|[]{} -<>"}
$data    []    =="Here   are   binary   characters:".chr(5).chr(250);
```

A Sdata-tömbben lévő sztringek a $encoded-tömbbe történő kódolásához add hozzá a következő kódot, és vizuális ellenőrzés céljából jelenítsd meg a kódolt sztringeket:

```
reset($data);
while (üst ($k,   $v) = each($data))

    $encoded[$k] = base64_encode($v);
     print("<br>encoded: " . $encoded[$k]);
```

Itt a vizuális ellenőrzés:

```
encoded:SGVsbG8gSmVubmlmZXI=
encoded:SGVyZSBhcmUgZnVubnkgY2hhcmFjdGVyczogPysmJyJ8W117fX4 8Pjs=
encoded:SGVyZSBhcmUgYmluYXJ5lGNoYXJhY3RlcnM6lAX6
```

Kódold ki a $encoded-ban lévő sztringet a base64_decode() segítségével, és jelenítsd meg az eredményt, hogy ellenőrizd, egyeznek-e az adatok az eredetivel:

```
reset(Sencoded);
while (üst ($k, $v) = each ($encoded) )

    print ("<br>decoded:      base64 decode($v));
```

Itt vannak a kikódolt adatok:

```
decoded:Hello Jennifer
decoded:Here are funny characters:?+&'"   []{}-<>;
decoded:Here are binary characters:ú
```

A base64-függvények bármilyen sztringet elfogadnak inputként, vagyis egy fájlfüggvényt használva beolvashatod egy fájl tartalmát a memóriába, és aztán kódolhatod a fájlt. A kódolt sztringeket nem tudod összefűzni, hacsak az eredeti sztring hossza nem három egész számú többszöröse. Ha egyszerre csak 99 bájtot olvasol be a fájlból, minden 99 bájtos

sztringet kódolsz, és összefűzöd az eredményt, akkor ez működik. Ugyanez a folyamat 100 bájtos olvasási hosszal nem működne. A következő kód megjeleníti az egy mozdulattal kódolt base64 kódolású szöveget, majd a három bájtonként kódolt szöveget, valamint ellenőrzés céljából a visszakódolt változatot:

```
$poem = "Roses are red, violets are blue,"
    . " PHP is sweeter than those
two"; print("<br>" . base64
encode($poem) ); $c = "";
while(strlen($poem) )

    $c .= base64_encode(substr($poem, 0,
    3)); Spoem = substr($poem, 3);

print("<br>" $c ) ;
print( base64decode($c)
```

A következőkben látható eredmény azt mutatja, hogy a két kódolás megegyezik (vedd észre, hogy a könyv tördelése miatt a kódolt sztringek nem férnek el egy sorban; a *B* a második sor elején mindkét esetben közvetlenül az első sor végén álló *i* után következik):

```
Um9zZXMgYXJlIHJlZCwgdmlvbGV0cyBhcmUgYmxlZSwgUEhQIGlzIHN3ZWV0ZXIgdGhhbi_
    B0aG9zZSB0d28=
Um9zZXMgYXJlIHJlZCwgdmlvbGV0cyBhcmUgYmxlZSwgUEhQIGlzIHN3ZWV0ZXIgdGhhbi_
    B0aG9zZSB0d28= Roses are red, violets are blue, PHP
is sweeter than those two
```

Ha veszed az *előző* kódot, és a 3-at kicseréled 11-re, akkor a következő eredményt kapod. A kikódoló függvény az első = jelig (vagy == jelig) kikódol, majd megáll:

```
Um9zZXMgYXJlIHI=ZWQsIHZpb2xldHM=IGFyZSBibHVlLCA=UEhQIGlzIHN3ZWU=dGVyIH_
    RoYW4gdGg=b3NlIHR3bw==
Roses are r
```

Aweb böngészése PHP-val

A PHP képes más honlapok böngészésére, és az információ közvetítésére a látogatóknak a te honlapodon keresztül. Számos példát találhatunk arra, hogy ezzel a technikával szerez nek információkat más oldalakról. I

Ott vannak például a mera-keresőoldalak, amelyek más keresőoldalak felé közvetítik a keresést, összegyűjtik és egy oldalon mutatják be az eredményeket. Ezt a technikát használják azok is, akik a legjobb elérhető árak, napi hírek vagy egyéb értékes információ után kutatva böngésznek más oldalakat (ha nem volna értékes az információ, akkor az emberek nem akarnák leszedni mások oldalairól). Végül is azok az emberek, akik pénzt költöttek az eredeti tartalom létrehozására, beperelték a tartalmat lemásoló kalózokat, és legtöbbjüket eltántorították ettől a tevékenységtől. Néhány kalóz átváltozott tisztességes üzletemberré, elkezdett licenc díjakatfizetni, és tovább folytatta a honlap közvetítő/összegyűjtő tevékenységét.

Miért írtam ezt a megoldást, ha illegális, immorális és szükségtelenül megnöveli a honlapod méretét? Ez a megközelítés azoknak az embereknek a leghasznosabb, akik intraneteket tar tanak karban. Ha az intraneted a pornográfia és az időpocsékotó játékok miatt blokkolja a hozzáférést a legtöbb weblaphoz, akkor a PHP-ban írhatsz egy szerver alapú közvetítő programot, amely a weblapok bizonyos részeit megjeleníti, más részeket letilt. Közvetít hetsz egy keresőmotort, de kitörölheted azokat a találatokat, amelyek nem megfelelő sza vakat tartalmaznak. Mielőtt közvetítenéd az oldalt, kérd a kereső üzemeltetőjének engedé lyét, és győződj meg arról, hogy rendelkezik olyan tartalomszűrővel, amely kielégítheti az igényeidet. i

Ha az intraneted egy általános iskolában működik, akkor minden'oldalon kiválaszthatod a hosszú és bonyolult szavakat, és egy linket rendelhetsz hozzájuk, amely egy szótárra mutat, hogy az elmagyarázza a szó jelentését. A 12. fejezetben bemutatott szótár-függvények mindent lefednek, amivel bonyolult szavakhoz egyszerűbbeket rendelhetsz, sőt bonyolult kifejezéseket egyszerűbb kifejezésekké alakíthatsz.

Ha az intranetedet gyengén látók használják, és nem akarsz sok pénzt költeni arra, hogy speciális böngészőket telepíts minden számítógépre, akkor a bejövő oldalakat közvetítheted egy szkripttel, amely megnöveli a betűméretet és a kontrasztot a háttérszínek eltávolításá val. Ha a gyengén látó felhasználóid Internet-kávézókból böngészik a hálót és nem változ tathatják a böngészőt az igényeik szerint, akkor létrehozhatsz számukra egy közvetítő ol dalt, amely HTML-szinten dolgozza fel a közvetített oldalakat, megnövelve a betűméretet, hanggá konvertálva a szövegeket, és a hangfájlokra mutató linkeket elhelyezve a szövegek körül. i

Gondolj a harmadik világbeli hallgatókra, akik gyenge gépeket és korlátozott képességű böngészőket használnak. Segítheted őket, ha egy olyan közvetítőt írsz, amely csökkenti a grafikák bonyolultságát, a nyakatekert animációkat egyszerű képekké alakítja, és mindent kitöröl, ami Java-t használ. (Hej, a Java kitörlése de sok oldalt hasznosabbá tenne!)

Mielőtt megpróbálnál a hálózaton keresztül hozzáférni bármihez, növeld meg a PHP időkorlátját, hogy engedélyezd a hálózati késéseket. A kód itt látható, és az időkorlátot egy későbbi kódban is használjuk:

```
$time_limit = 30;
set_time_limit($time_limit) ;
```

Most következik a kód. Az fsockopen()-függvény megnyit egy weblapot, mintha ez egy fájl lenne, az fputsQ egy HTTP lekérést kezdeményez, a kód többi része pedig pontosan olyan, mint amit azért írnál, hogy egy fájlt megjeleníts a képernyőn. A kód a weblapot egy tömbbe olvassa be, vagyis kiegészítheted a kódot, hogy módosítsa az oldalt. Az oldalt a módosítási igényeidtől függően egy tömbbe vagy egy sztnngbe is beolvashatod:

```
$page = "www.yahoo.com";
$file = fsockopen($page, if 80, $errno, $error, $time limit)
($file)

    fputs($file, /
    "GET HTTP/1.0\r\n\r\n")
    while(!feof($file)
    í
```

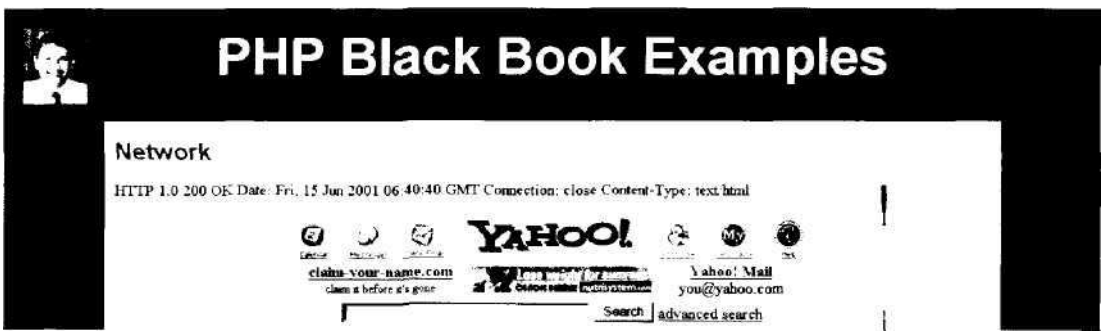
```
    $data[] = fgets($file, 1000); }
    fclose($file) ;

else
    {
    print ("<br>Error:        $errno        $error);
```

Az fsockopenQ-függvény paraméterei egy honlap neve, egy port szám, két változó neve, amelyek egy hibakódot és egy hibaüzenetet fognak eltárolni, és egy időkorlátérték másodpercben. Visszatérési értéke a fájlkezelő, amelyet bármelyik fájlfüggvény felhasználhat. Az **fputs()** a fájlhoz ír (vagyis a weblaphoz), és elküld egy HTTP GET-lekérést, amelyet a lekérés végét jelző üres sor követ.

A következő kód egyszerűen megjeleníti az adatokat. Egy hagyományos weblaphoz nem szükségesek különleges trükkök. Ha megjelenítés előtt módosítani akarod az oldalt, akkor a módosító kódot egy önálló részként a megjelenítő ciklus elé kell beilleszteni. Ilyen módon több szűrőt is hozzáadhatsz az oldal módosítására, és mindegyik egy tömbből (vagy sztringből) kapja az adatokat, a módosított adatokat pedig ugyanabba a változóba teszi vissza. A szűrőket egymástól függetlenül tetszőleges sorrendben halmozhatod.

```
reset($data);
while(list($k, $v) = each($data)
    {
    print($v);
```

A Yahoo! több okból is nagyszerű oldal. Az árfolyam-kalkulátorukat találtam a leghaszno-sabbnak, de próbálj csak megtervezni egy üzleti utat, és minden szükséges dolgot megtalálni egy olyan oldal nélkül, mint a Yahoo!. A 13.2 ábrán a yahoo.com a tesztoldalamon keresztül közvetítve látható. Azért választottam a Yahoo!-t, mert mint honlap nagyon hasznos, és mint weboldal a HTML és más technológiák olyan egyszerű implementálása, hogy az egyik legkönnyebben volna módosítható a gyengénlátók számára. A honlap nem tartalmaz átirányításokat, Java-t vagy bármi egyebet, ami hátrányosan befolyásolja a hozzáférést. Könnyen végigmehetnél az oldalakon HTML-szinten, megnövelhetnéd a betűméretet, a világos színeket sötétebbre változtathatnád, hanggá konvertálhatnád a szövegeket és a hangfájlokra mutató linkeket helyezhetnél a szövegek köré. Talán a Yahoo! támogatna egy gyengén látók számára készített közvetítő oldalt.



13.2 ábra Szörfözés egy másik honlapon

A Yahoo! után egy ASP-oídallal próbálkoztam, és a következő hibába ütköztem. Az oldal az eredeti szerver helyett a közvetítő szerverről próbál betölteni egy JavaScript-fájlt. Ha ilyen jellegű oldalakat akarsz közvetíteni, akkor közvetítened kell a hivatkozott fájlokat, a relatív hivatkozásokat abszolút hivatkozásokra kell változtatnod, vagy egyszerűen el kell hagynod a hivatkozásokat. Néhány oldal összeesik, ha eltávolítod az összes JavaScrpitet, más oldalak megbízhatóbbá válnak, és a jól megírt lapok ugyanúgy működnek:

```
Not Found
The requested URL /jscript/fpi-init.js was not found on this
server.
Additionally, a 404 Not Found error was encountered while trying
to use
    an ErrorDocument to handle the request.
(A keresett lap URL /jscript/fpi-init.js nem található a
szerveren. Továbbá a 404 fájl nem található, hiba történt kérés
teljesítése közben.)
```

Ha még száz oldal a rendelkezésemre állna, akkor felsorolhatnám az összes JavaScript programozási hibát, amit a drága, „professzionális" weblapokon találtam, de akkor további 300 oldalra lenne szükségem, hogy bemutassam a helyes JavaScript-programozást. Nos, itt van a teljes titok egy sorban:

- *Mielőtt egy objektumot használnál, győződj meg az objektum létezéséről.*

Ez a megoldás használta a HTTP-protokollt, aminek a definícióját az RFC2068-ban találod meg a www.w3.org/Protocols/rfc2068/rfc2068 oldalon. A HTTP-nek csak két verziója van, a HTTP 1.1.

Linkek ellenőrzése i

Az fsockopen()-függvénnyel ellenőrizheted egy link érvényességét. Ez az utasítás olyan lapok számára tökéletes, amelyek más oldalakra mutató linkeken alapulnak. Ha a lapod egy linkgyűjtemény, amely a világ összes kisállat-kereskedését felsorolja, akkor írhatsz egy szkriptet, amely naponta minden kisállat-kereskedés linkjét ellenőrzi. A portállapok gyorsan elveszítik a fogyasztóikat, ha a linkek halott oldalakra mutatnak. Egy kis szkript felhívhatja a figyelmedet a problémás linkekre mielőtt a fogyasztók kiábrándulnának.

A következő kód egy darab linket ellenőriz, de egy 1000 linket tartalmazó listát is kiolvashatna egy adatbázisból. Az fsockopenQ-függvény megnyit egy honlapra mutató linket, az fputs() elküldi a fejrész-információk lekérését (egy link ellenőrzéséhez elegendő a fejrész) és az fgetsQ elolvassa a fejrészt. A példában a fejrészt megjelenítjük, vagyis láthatod az eredményt, de könnyen kiegészítheted a kódot, hogy ellenőrizze a tartalmat, és csak akkor jelenítsen meg egy üzenetet, ha az eredmény azt jelzi, hogy egy oldal hiányzik:

```
$site = "www.phpbuilder.com";                                    j
$page = "/columns/peter20000629.php3";                           '
$file = fsockopen($site, 80, $errno, $errstr, $time
limit);
if($file)                                                        j

    fputs($file, "HEAD " . $page . "
    HTTP/1.0\r\n\r\n"); while( i f e o f ($file))

        $data[] = fgets($file, 1000)
```

```
    fclose($file);
    }
else
    {
    print("<br>Error: " . $errno . " " . $errstr );
    }
reset($data); while (üst ($k, $v)
= each($data))
    {
    print("<br>" . $v );
    }
```

Az alábbiakban látható az eredmény, ami megerősíti, hogy az oldal létezik:

```
HTTP/1.1 200 OK
Date:Sun,17 Jun 2001 04:42:42 GMT
Server:Apache/l.3.17 (Unix)PHP/4.0.4pll AuthMySQL/2.20
X-Powered-By:PHP/4.0.4pll
Connection:close
Content-Type:text/html
```

Ha ugyanezt a kódot arra használod, hogy egy nem létező oldalt kérj le, akkor a következő eredményt kapod. Figyeld meg, hogy az eredménykód 200-ról 404-re változott:

```
HTTP/1.0 404
Date:Sun,17 Jun 2001 09:11:15 GMT
Connection:close
Content-Type:text/html
```

Néhány weblap webszerverébe beépítették „az oldal nem található" funkciót, amit az Apache konfigurációs fájljában **Error Document-nek** hívnak. Ez az eszköz egy általános hibaüzenet-oldalra vagy egy oldalkereső eszközhöz irányítja át a böngészőt, mint a következőkben látható. A 302 átirányítást jelent, és a **Location:** mező adja meg az új címet. Ha átirányítást találsz, akkor kézzel ellenőrizheted, hogy valóban a megfelelő oldalra mutat, majd frissítheted a linkedet.

```
HTTP/1.1 302 Found
Date:Sun,17 Jun 2001 04:46:24 GMT
Server:Apache/1.3.17 (Unix)PHP/4.0.4pll AuthMySQL/2.20
X-Powered-By:PHP/4.0.4pll
Location:/search/?feedback=Page+Not+Found
Connection:close
Content-Type:text/html
```

Milyen régi egy oldal? Egy oldal kora fontos kérdés, ha aktuális információkat akarsz szolgáltatni, mint például az időjárás-jelentés, vagy ha szeretnéd ismerni a források frissességét. Cseréld ki az előző kódban az fputs()-sort a következő sorral, hogy ellenőrizd, változott-e az oldal 2001. június 17-e óta. Illeszd be a saját idődet, ha szükséges. Ha egy hétnél fiatalabb oldalra van szükséged, akkor illeszd be a 7 nappal ezelőtti dátumot. Csak az aktuális oldalakat fogod visszakapni, és a visszatérési kód *200* lesz. Ha az oldal idősebb, mint a megadott dátum, akkor egy "page not found" (az oldal nem található) hibaüzenetet kapsz:

```
fputs($file,"HEAD " . $page ."HTTP/1.0 \r \n"
    . "If-Modified-Since:Sunday, 17 Jun 2001 01:00:00 \r \n \r \n
```

Ezekből a feladatokból annyit automatizálhatsz, amennyit csak akarsz. Általában az ismétlődő ellenőrzést szokták automatizálni, és az olyan kivételek kezelését oldják meg kézzel, mint az átirányítások ellenőrzése. A kivételekről kaphatsz egy e-mailt, vagyis az ellenőrzést futtathatod éjszaka, és reggel megnézheted az eredményt. Az e-mailekkel a 15. fejezet foglalkozik. ; ,jt I^.^T:

FTP-függvények használata

Ebben a megoldásban az összes FTP-függvény használatát bemutatjuk, vagyis mindent megtehetsz, amit az FTP lehetővé tesz. Néhány függvény függ az operációs rendszertől, mivel az operációs rendszerek sajátos módon kezelik a fájlokat. Az egyik ilyen függvény az ftp_site(), amely teljes mértékben az FTP-szerver szoftverétől függ.

Az ftp_connect() paramétere egy host-név és egy opcionális port-szám, és egy erőforrás-azonosítót ad eredményül, amely pont olyan, mint egy fájlkezelő, vagy az fsockopen() által visszaadott azonosító. A port-szám alapbeállítás szerint 21:

```
$host    =    "ftp.gnu.org";
if{!$ftp    =    ftp_connect($host))

    print ("<br>Error:                    $errstr)                    seís
    }                        $errno          ;
```

Miután megnyitottál egy kapcsolatot az FTP-szerverrel, be kell jelentkezned egy felhasználói azonosítóval és egy jelszóval. Megpróbálkoztam névtelenül bejelentkezni, de a *Wrong paraméter count for ftp Jogin* (Hibás paraméter-szám az ftploginhez) hibaüzenetet kaptam:

```
ftp_login($ftp);                                                    i
```

Megpróbáltam a névtelen bejelentkezést üres sztringekkel, mint itt:

```
ftp_login($ftp,"","");                                                    39f
                                                                          xrtw
```

Ezt az eredményt kaptam:

```
Warning:ftp    login:'USER':command    not    understood
(Figyelem:    Az    ftp_login:'USER'    parancs    nem    értelmezhető.) ■      n
```

Mikor megpróbáltam jelszó nélküli felhasználói nevet létrehozni, akkor ugyanezt az üzenetet kaptam, csak a USER helyén PASS szerepelt. A működő kód a következő példában látható, egy hibakereséssel kiegészítve. A saját szkriptedben ennek a kódnak minden részét az előző kód megfelelő zárójeleibe helyezheted, így a bejelentkezés csak akkor fut le, ha működik a kapcsolat, illetve beszúrhatsz a hibaüzenet után egy exit()-utasítást, aminek hatására a szkript leáll:

```
if(!ftp    login($ftp,    "anonymous",    "x@y.com"))

    print("<br>Login                    $errno          $errstr
    error:                                              );
```

Ha szeretnéd tudni, milyen rendszer áll az FTP-szerver mögött, akkor használd a következő kódot. A FTP legnagyobb részéhez nem szükséges a rendszer ismerete, de se-

gíthet eldönteni, hogy a fájlnévben szereplő szóközöket átfordítsd-e aláhúzásokra, illetve választ adhat az egyéb operációs rendszertől függő kérdésekre:

```
if($type = ftp_systype($ftp))

    print("<br>systype: $type);

else
    {
                                                    Serrstr);
    print("<br>Systype error: " . $errno . " "
```

A kód eredménye az ftp.gnu.org oldalon:

```
systype:UNIX
```

Ha az FTP-t egy tűzfal mögül használod, akkor a sikeres FTP-zéshez szükséged lehet a következő parancsra:

```
if(ftp_pasv($ftp, true)) í
    print("<br>passive mode
    on.");

else

    print {"<br>Systype error: " . $errno . " " . $errstr);
```

A PHP4 új függvénye, az **ftp_nlist()** felsorolja egy könyvtár tartalmát az FTP-szerveren. A következő kód az ftp_nlist()-függvényt, a megadott kulcsokat és értékeket, valamint egy tömb elemeinek felsorolására használatos általános kódot használja:

```
if($array = ftp_nlist($ftp, $directory))

    reset($array);
    while (üst ($k, $v) = each ($array) )

        print("<br>k: " . $k . ", v: " . $v);


else { print("<br>ftp_nlist() failed.");
    }
```

Itt látható az eredmény, ami ránézésre a Unix Is-utasításának eredményére emlékeztet, leszámítva a dupla perjeleket a könyvtárak és fájlnevek elején (a perjeleket el kell távolítani, mielőtt más függvényekben használnád a neveket):

```
k 0, v | //pub
k 1, v | //welcome .mse
k 2, v | //bin
k 3, v | //lib
k 4, v | //gnu
k 5, v | //ls-lrR. txt
```

```
k: 6,   v: 6, v:
k: 7,   v: //ls-lrR.txt.gz
k: 8,   v: //non-gnu      --------
k: 9,   v: //README
k: 10,  v |//lpf.README
k: 11,  v |//old-gnu
k: 12,  v |//find-ls.txt
k: 13,  v |//find-ls.txt.gz
k: 14,  v |//md5sums.txt
k: 15,  v |//md5sums.txt.gz
k: 16,  v |//var //third-party
          c
```

.sart-TVA

t

Ötlet: Ez az egyik leginkább frusztráló elgépelési hiba: "while(list($k, $v) = each($array));". A pontosvessző a sor végén nem okoz szintaktikai hibát, és a kód látszólag működik, de a tömb örökké üres marad. Fúúú! Két pohár kávéba és 185 új ősz hajszálba kerül, mire megfejtek egy ilyen buta elírást.

Az ftp_rawlist() új a PHP4-ben, és az FTP-szerver egy könyvtárában található alkönyvtárak és fájlok módosítatlan, szüretien felsorolását adja. A következő kód az ftp_rawlist()-utasítást alkalmazza az előző kódban szereplő könyvtárra, és megjeleníti az eredményként kapott tömb adatait az ftp_nlist()-ben:

```
if($raw = ftp rawlist($ftp, $directory))

    reset($raw);
    whiledist ($k,    $v)   =   each($raw>)

        print("<br>"    .    $v);

else

    print("<br>ftp_rawlist failed.");
```

Az itt látható eredmény a Unix Is -l-utasítására emlékeztet:

```
-rw-r—r—  1 ftp ftp 1516 Aug 18 2000 README
drwxr-xr-x  2 ftp ftp 4096 Jan 1 1999 bin
-rw-r-r-  1 ftp ftp 2131267 Jul 6 13:28 find-ls.txt[1]
-rw-r-r-  1 ftp ftp 208066 Jul 6 13:28 find-ls. txt. gz
drwxr-xr-x193 ftp ftp 8192 Jul 3 03:03 gnu
drwxr-xr-x  2 ftp ftp 4096 Jan 1 1999 lib
-rw-r—r—  1 ftp ftp 90 Feb 16 1993 lpf.README
-rw-r—r—  1 ftp ftp 1342751 Jul 6 13:28 ls-lrR.txt
-rw-r—r—  1 ftp ftp 137580 Jul 6 13:28 ls-lrR.txt.gz
-rw-r —r—  1 ftp ftp 997585 Jul 6 13:33 md5sums.txt
-rw-r—r—  1 ftp ftp 293249 Jul 6 13:33 md5sums.txt.gz
lrw-r~r~  1 ftp ftp 11 Feb 23 03:55 non-gnu -> gnu/non-gnu
drwxr-xr-x10 ftp ftp 4096 May 29 14:50 old-gnu
lrw-r—r—  1 ftp ftp 1 Feb 23 03:19 pub -> .
drwxr-xr-x  2 ftp ftp 4096 Mar 5 20:28 third-party
-rw-r-r—  1 ftp ftp 128800 Mar 25 03:03 var
-rw-r-r-  1 ftp ftp 980 Aug 18 2000 welcome.msg
```

Ötlet: Ha szereted az Is- és hasonló Unix-parancsok használatát, de NT-n dolgozol, akkor telepítsd a Cygwint. A Cygwin számos POSIX-kompatíbilis Unix-parancsot tartalmaz az NT-hez.

A könyvtár megváltoztatásához a következőben bemutatott ftp_chdir()-függvényt használhatod. Ha nem létezik a könyvtár, akkor az ftp_chdir() egy üzenetet küld a böngészőablakba, vagyis írj egy @ szimbólumot a függvény neve elé, hogy letiltsd a hibaüzenetet, és végezd el a saját hibakezelésedet (ami a példában csak egy hibaüzenet):

```
if(@ftp_chdir($ftp, "pub"))

    print("<br>chdir    worked");
else

    print("<br>chdir    failed");
```

Ha egy szinttel feljebb akarsz ugrani a könyvtárszerkezetben, akkor a cd..-paranccsal egyenértékű ftp_cdup()-függvényt használhatod, ami a következő példában látható (a következő néhány példához add hozzá a megfelelő hibaüzeneteket, amelyeket az előző példákban is használtunk):

```
if (@ftp_cdup($ftp) )
```

Egy könyvtárat a következőben bemutatott ftp_mkdir()-függvénnyel hozhatsz létre az FTP-szerveren. A függvény a létrehozott könyvtár nevét adja vissza:

```
if($new_dir    =ftp_mkdir($ftp,"new") )
```

Egy könyvtár (vagy fájl) átnevezéséhez az FTP-szerveren használd az itt látható ftp_raname()-függvényt (a kód többi részét másold át az ftp_chdir()-példából):

```
if(ftp_rename($ftp,"new","test"))
```

Egy könyvtárt az ftp_rmdir()-függvénnyel törölhetsz az FTP-szerverről. Az eredmény siker esetén igaz, kudarc esetén hamis (a kód többi részét másold át az itp_chdir()-példa-ból):

```
if(ftp_rmdir($ftp,"new"))
```

A következő kód egy könyvtár vagy fájl kitörlésére alkalmas ftp_delete()-függvényt mutatja. Az eredmény siker esetén igaz, kudarc esetén hamis (a kód többi részét másold át az ftp_chdir()-példából):

```
if(ftp_delete($ftp,"new.txt") )
```

A letöltés előtt ellenőrizned kell a fájlok méretét arra az esetre, ha a letöltetni kívánt rövid szöveges fájlok között van egy 650 MB-os CD image-fájl. A következőben bemutatott ftp_size() a fájl méretét adja vissza. Mivel a nulla érvényes fájlhossz lehet, és mivel a nulla összekeverhető a hamissal, ha nem a megfelelő típusú összehasonlítást végzed, ezért az ftp_size() -1-et eredményez kudarc esetén. Mikor az első alkalommal kipróbáltam, -1-et

kaptam, mert *Welcome* helyett *welcome-ox.* gépeltem be. Hoppá! Az ftp_systype UNIX-ot eredményezett:

```
$size = ftp_size($ftp, "Welcome");

if{?size < 0)                                                    ^

   print("<br>ftp_size error.");

else                                                                                -

   print("<br>size: " . $size);
```

A Welcome-ra ezt az eredményt kapjtuk:

```
size:364
```

Aktuális egy fájl? Az **ftp_mdtm()** egy Unix időbélyegzőt ad eredményül, amely a fájl utolsó módosítását mutatja. Mivel a nulla érvényes dátum/idő lehet, és mivel ha nem a megfelelő típusú összehasonlítást végzed, a nulla összekeverhető a hamissal, ezért az **ftp_mdtm()** -1-et eredményez kudarc esetén. A következő kód az **ftp_size()** kódjának módosítása - a méret helyett a módosítás dátumát/idejét jeleníti meg:

```
$time = ftp_mdtm($ftp, "Welcome")
if{$time < 0)
    {
   print("<br>ftp_mdtm error.");

else

   print("<br>date/time:          date("Y-m-d H:i:s",
                                $time)
```

A Welcome-ra ezt az eredményt kapjtuk:

```
time: 2001-08-09 03:14:51
```

Az **ftp_get()**-függvénnyel fájlokat kapsz a távoli szervertől. Az első paraméter az FTP-kapcsolat azonosítója, a második a helyi fájlnév, a harmadik a távoli fájlnév, az utolsó pedig azt közli az FTP-vel, hogy ASCII szöveges módban vagy bináris módban dolgozzon. Az alapbeállítás szerint a fájl abba a könyvtárba kerül, ahol az **ftp_get()**-függvényt tartalmazó szkript található, de ha a helyi fájlnevet teljes elérési úttal látod el, akkor a fájl bárhová kerülhet a szkriptet futtató gépen (vagyis bárhová, ahová van írási jogod). Az **ftp_get()**-függvénnyel nem irányíthatod a fájlt arra a számítógépre, amelyik a szkript futtatását kérte (amelyiken a böngésző fut). A következő példában a fájl a tesztpartíciómra, a t:/-re kerül:

```
if(ftp_get($ftp, "t:/welcome", "Welcome", FTP_BINARY))                    )rí
    í
   print("<br>Get worked.");

else

                                                                          í Jwlé.
   print("<br>Get
   failed.");
```

Az **ftp_put()** függvénnyel felmásolhatsz egy fájlt a távoli szerverre. Az első paraméter az FTP-kapcsolat azonosítója, a második a távoli fájlnév, a harmadik a helyi fájlnév, az utolsó pedig azt közli az FTP-vel, hogy ASCII szöveges módban vagy bináris módban dolgozzon. A következő példában egy helyi fájlt a Netscape FTP-szerverére töltünk fel, amelyet már korábban megnyitottunk. Ez a szerver nem engedélyezi a feltöltéseket, úgyhogy másik tesztszervert kell találnod magadnak:

```
if(ftp_put($ftp,    "Welcome",    "t:/welcome",    FTP_BINARY))
    {
    print ( "<br>Put worked.");                                    ";Jr.i
    }
else                            -   .;:.
    {                           ' -   ti
    print("<br>Put failed.");
    }
```

A távoli fájlokat az ftp_fget()-függvénnyel irányíthatod a böngészőt futtató gépre. Az első paraméter az FTP-kapcsolat azonosítója, a második egy fájlkezelő a helyi fájl számára, a harmadik a távoli fájlnév, az utolsó pedig azt közli az FTP-vel, hogy az ASCII szöveges módban vagy bináris módban dolgozzon. A helyi fájl lehet egy ideiglenes fájl, és az ideiglenes fájl tartalmát az fpassthru()-függvénnyel irányíthatod a böngészőhöz. A **tempnam()** egy ideiglenes fájlt oszt ki, az **fopen()** megnyitja a fájlt írás/olvasás módban, az ftp_fget() beírja a távoli fájlt a helyi fájlba, a **rewindQ** visszaállítja a fájl pointerét a fájl elejére, az **fpassthru()** pedig kiírja a böngészőnek a fájl tartalmát. A fájlfüggvények részletes leírása a 8. fejezetben található:

```
$temp_file = tempnam("", "");
if($file = fopen($temp_file, "w+"))
    {
    if(ftp_fget($ftp, $file, "Welcome", FTP_BINARY))
        {                                                          : %
       print("<br>Fget worked." );
       rewind($file);
       fpassthru($file);

    else

       print ("<br>Fget failed."
       fclose($file);


else
    {
    print("<br>Fopen   failed.");
    }
```

Az ftp_fget()-példa módosítható úgy, hogy egy állandó másolatot hagyjon a lemezen, vagy hogy módosítsa a fájl tartalmát a távoli szerverről történő kiolvasás és a böngésző felé történő továbbítás között. Ha az fpassthru()-t egy magasabb szintű kódra cserélnéd, akkor a fájl tartalmát formázva is megjeleníthetnéd a képernyőn. „A web böngészése PHP-val" című Gyors megoldásban leírom, hogyan segíthetjük a gyengén látókat a böngészésben azáltal, hogy egy olyan szerveren keresztül közvetítjük a böngészett lapokat, amely javítja a

lapok vizuális formázását. Ugyanezt az elvet alkalmazhatod az FTP-re, vagyis javíthatod az FTP-n keresztül elérhető dokumentumok formázását.

Az **ftp_fput()**-függvény feltölt egy fájlt egy nyitott fájlkezelőből egy távoli szerverre. Azt leszámítva, hogy a helyi fájl fájlnév helyett fájlkezelőként van megadva, a paraméterek megegyeznek az **ftp_put()** paramétereivel. Egy nyilvánvaló használati lehetőség az **ftp_fput() és** az **ftp_fget()** közös használatára, ha több szervert tartasz karban. Ekkor az egyik szerverről egy másik szerverre irányíthatod a fájlokat (lásd a következő példát). Az egyetlen különbség az előző példakódhoz képest az fpassthruQ-függvény lecserélése az **ftp_fput{)**-függvényre. Egy üzleti alkalmazásban a korábban szereplő **ftp_put()**-példához hasonlóan tesztelnéd az **ftp_fput()** eredményét:

```
$temp file = tempnam("", "");
if($file = fopen($temp_file,
"w+"))

    if(ftp_fget($ftp, $file, "Welcome", FTP_BINARY))

        print ("<br>Fget worked." );
        rewind($file);
        ftp_fput($ftp2, "Welcome", $file, FTP_BINARY))

    else
        {
        print("<br>Fget failed.");
        fclose($file) ;

else
    {
    print ("<br>Fopen failed.");
```

Egy fájl hozzáférésének vagy csoportos jogainak megváltoztatásához az FTP-szerveren a lent látható **ftp_site()**-függvény használható. Az eredmény siker esetén igaz, kudarc esetén hamis. (A kód többi részét másold át az **ftp_chdir()**-példából.) Egyedül egy probléma van a site-paranccsal: a formátum és a tartalom a parancsot fogadó rendszer típusától függ. Meg kell határoznod a rendszer típusát az **ftp_systype()**-függvénnyel, és meg kell nézned a rendszer honlapját a parancs formátumának megállapításához. Néhány rendszer a site-pa-rancs után elfogadja a Unix-parancsokat, abban a formában, ahogyan a parancssorba begépelnéd, például a **site chmod** *777:*

```
if(ftp_site($ftp,$command))
```

Ha befejezted a host-tal létesített kapcsolat használatát, akkor az **ftp_quit()** függvénnyel zárd be azt. Ezután tetszőleges FTP-szervert elérhetsz a számodra megfelelő módon.

```
if ($ftp)

    ftp^quit($ftp);
```

Curl használata

Használd a Curlt a PHP FTP- és csatolófüggvényei helyett, és próbálj ki minden lehetséges megközelítést, mielőtt eldöntenéd, melyik a legjobb számodra. Ha néhány jellegzetessége miatt szükséges számodra a Curl használata, akkor megkönnyítheti az életed, ha mindent Curl-ben csinálsz.

A következő példa egy kapcsolatot kezdeményez egy weblappal a curl_init()-függvénnyel, megnyit egy fájlt az fopen()-nal és a curl_setopt() függvénnyel közli a Curllel a fájl azonosítóját. A Curl képes elolvasni az oldalt a távoli szerverről, és képes egy fájlba írni az oldalt. Ha meg akarod gátolni a Curlt abban, hogy a HTTP-fejrészeket is elhelyezze a fájlban, akkor kapcsold ki a CURLOPT_HEADER-opciót. A curl_exec()-függvény végrehajt egy lekérést, majd a curl_close() lezárja a kapcsolatot. Az fclose() bezárja az output-fájlt:

```
if($curled = curl init ("http: //petermoulding.

    cotn/") ) if($file = fopen("t:/testpage.txt",

    "w+"))

        curl_setopt($curled, CURLOPT_FILE,
        $file);
        curl_setopt($curled, CURLOPT_HEADER, 0);
        curl^exec($curled) ;
        curl_close($curled) ;        r
        fclose($file);

    else
        {                                                    - {
        print("<br>fopen()  failed."  );                      st

    }                                !-.::«
else {
    print("<br>curl_init() failed." );
```

A példakód kiterjesztésével elérhető, hogy a megkapott oldalt megjelenítsük egy böngészőben, információkat nyerjünk ki belőle, és mindenféle egyéb tevékenységet végezzünk, amelyeket az fsockopen() példában már körvonalaztam. Egy ilyen egyszerű példa esetében látszólag nem sok különbség van az fsockopen() és a curl() között.

A Curl az SSL-t és a HTTPS-oldalakat is kezeli, ami különösen fontos, ha szerverek között osztasz meg információkat. Ha tartalmat árulsz más lapok számára, akkor szeretnéd biztonságosan megoldani az információáramlást, hogy ne tudják lemásolni az információidat, és meggyőződhess arról, hogy az ügyfeleid valóban megkapták a tartalmat. Ugyanez érvényes, ha te veszel tartalmat; szeretnél bizonyos lenni afelől, hogy a tartalom a megfelelő forrásból származik, és hogy a versenytársaid nem tudják ingyen megszerezni az információkat. Valószínűleg nekik is megvan a könyvem, vagyis ők is használhatnak Curl-t az oldalak lemásolására, hacsak nem teszed biztonságossá az oldalakat.

A Curl segítségével biztonságosan másolhatsz a szerverek között. A másolás előtt kapcsold be a CURLOPT_SSLCERT- és a CURLOPT_SSLCERTPASSWD-opciókat, és használj https-protokollt.

Néhányan arról számolnak be, hogy a Curl gyorsabb. Nem teszteltem őket párhuzamosan, mivel a Curl-ben eltérő opciók vannak, és én inkább hajlamos vagyok utasításkészletet választani az egyszerű használat és a rugalmasság, mint a gyorsaság alapján. A hálózati idő nagy részét a kapcsolat megteremtése emészti fel, tehát ha újra fel tudod használni a kapcsolatodat egy második lekéréshez, akkor időt takarítasz meg. Egy másik módja az idő megtakarításának, ha csak a fejrészeket kéred le, mikor nincs szükséged az oldal tartalmára. A teljesítményeredményeid változóak lesznek. Olvasd át az összes Curl-opciót, hogy el-dönthesd, segítenek-e optimalizálni a Curl teljesítményét.

Curl-opciók

A Curl által végzett műveletek a curl_setopt()-függvénnyel megadott opcióktól függenek, valamint az URL formátumától, különösen az első részétől, a sémától vagy a protokolltól. A Curl a séma alapján dönti el, hogy HTTP- vagy FTP-műveletet végezzen. Az alábbiakban felsoroljuk az opciókat, de minden újabb verzió a meglévő opciók finomítását és újabb opciók hozzáadását vonja maga után.

CURLOPTJCOOKIE
A CURLOPT_COOKIE-opció értékének egy sztringet adhatsz meg, amely a következő HTTP-fejrésszel elküldött cookie-t tartalmazza.

CURL OPT_COOKIEFILE
A CURLOPT_COOKIEFILE-opció értékének a cookie-adatokat tartalmazó fájl nevét kell beállítani. A cookie-fájl használhat Netscape-formátumot vagy HTTP stílusú fejrészeket:

```
curl_setopt($curled,CURLOPT_COOKIEFILE,"test.txt");
```

iA
3

CURLOPT_CUSTOMREQUEST
A CURLOPT_CUSTOMREQUEST beállításával a GET- és a HEAD-opciókat egy másikra, például DELETE-re cserélheted, mikor egy HTTP-lekérést hajtasz végre. Ez csak akkor működik, ha a távoli szerver elfogadja a parancsot:

```
curl^setopt($curled,CURLOPT_CUSTOMREQUEST, "DELETE    _string    value_");
```

CURL OPT_FA IL ONERROR
Állíts be a CURLOPT_FAILONERROR-opciónak nullától eltérő értéket, ha azt szeretnéd, hogy a PHP leálljon, ha a Curl egy 300-nál nagyobb HTTP-kódot ad vissza. Az alapbeállítás szerint a HTTP-kódtól függetlenül minden oldalt megkapsz:

```
curl setopt($curled,CURLOPT_FAILONERROR, 1) ;
```

CURL OPT_FOLLOWLOCATION
Állíts be egy nullától különböző értéket a CURLOPTFOLLOWLOCATlON-opciónak, és a Curl követni fogja a Location:-fejrészeket (átirányításokat), amelyeket a többi fej-

résszel együtt kap a szervertől. Fennáll a lehetősége az átirányítási hurok kialakulásának, mert a Curl minden átirányítást követni fog, és egy hiányzó oldalt, ugyanarra a hiányzó oldalra is átirányíthat:

```
curl_setopt(?curled,CURLOPT_F0LL0WLOCATION,1) ;
```

CURLOPT_FTPAPPEND

Állíts be egy nem nulla értéket a CURLOPT_FTPAPPEND-opcÍónak, és az FTP ahelyett, hogy felülírná a távoli fájlt, a másolt fájlt hozzáfűzi a távoli szerveren lévő fájlhoz:

```
curl_setopt(Scurled,CURLOPT_FTPAPPEND, 1);
```

CURLOPT_FTPLISTONLY

Állíts be egy nem nulla értéket a CURLOPT_FTPLISTONLY-nak, és az FTP egy könyvtár listázásakor semmi mást nem fog megjeleníteni a neveken kívül:

```
curl_setopt (Scurled, CURLOPT__FTPLISTONLY, 1) ;
```

CURLOPT_FTPPORT

Adj meg egy sztringet a CURLOPT_FTPPORT-nak, amely egy IP-címet tartalmaz egy FTP POST-hoz. Az érték lehet egy IP-cím, egy host-név, egy hálózati interfész név (Unix alatt) vagy egy kötőjel (-), amelynek hatására a rendszer alapértelmezett IP-címét használja:

```
curl_setopt($curled,CURLOPT_FTPPORT,"192.168.0.1");
```

CURLOPT_HEADER

Állíts be a CURLOPT_HEADER-nek egy nem nulla értéket, ha azt szeretnéd, hogy a HTTP-fejrészek szerepeljenek az outputban:

```
curl    setopt($curled,CURLOPT    HEADER,1);
```

CURLOPTJNFILESIZE

Mielőtt féltőkénél egy fájlt egy távoli szerverre, állítsd be a fájl méretét a CURLOPT_INFILESIZE-ban:

```
$size    =filesize("test.txt");
curl    setopt(Scurled,CURLOPT    INFILESIZE,$size);
```

CURLOPTJ-OW_SPEED_UMIT

A Curl a CURLOPT_LOW_SPEED_LIMlT-ben beállított, bájt per másodpercben adott átviteli sebesség alatt megszakítja az átvitelt:

```
curl_setopt($curled,CURLQPTLOW_SPEED LIMIT,300);
```

CURLOPT_LOW_SPEED_TIME

Állítsd be a CURLOPT_LOW_SPEED_TIME-ban, hogy hány másodpercet várakozzon a Curl, mielőtt megszakítja az átvitelt, ha az átviteli sebesség a CURLOPT_LOW_SPEED_LIMIT-ban megadott érték alá esik:

```
curl__setopt ($curled, CURL0PT_LOW_SPEED_TIME, 20) ;
```

CURLOPT_MUTE

Ha a CURLOPT_MUTE-nak egy nem nulla értéket állítasz be, „elnémítja" a PHP-t, amikor az a Curl-függvényeket hajtja végre:

```
curlsetopt($curled,CURLOPT_MUTE,1);
```

CURLOPT_NETRC

Ha egy távoli honlaphoz kapcsolódsz, akkor állíts be egy nem nulla értéket a CURLOPT_NETRC-nek, és a Curl a —./netrc-fájlodban fog felhasználói nevet és jelszót keresni a távoli oldalhoz:

```
curl    setopt($curled,CURLOPT    NETRC,1);
```

CURLOPT_NOBODY

Ha a CURLOPT_NOBODY értékét nullától eltérőre állítod, akkor az oldal törzsrésze nem fog szerepelni az outputban. Ne feledd el megváltoztatni a CURLOPT_HEADER-t, ha csak a fejrészt szeretnéd megkapni:

```
curl_setopt($curled,CURLOPT_NOBODY,1);
```

CURLOPT_NOPROGRESS

Állíts be nulla értéket a CURLOPT_NOPROGRESS-nek, ha azt szeretnéd, hogy a PHP megjelenítse, mekkora hányad van készen az átvitelből. Egy nem nulla érték beállításával kikapcsolhatod az átvitel állapotának megjelenítését:

```
curl_setopt ($curled, CURLOPT__NOPROGRESS, 0) ;
```

CURLOPT_POST

Állíts be nem nulla értéket a CURLOPTPOST-nak, ha a HTML-űrlapok által használt POST-metódussal akarsz egy lekérést elküldeni:

```
curl    setopt($curled,CURLOPT_POST,1);
```

CURLOPT_POSTFIELDS

A CURLOPT_POSTFIELDS-opció értékének beállíthatod a HTTP POST-metódusával elküldendő adatokat tartalmazó sztringet.

CURLOPT_PROXYUSERPWD
Egy proxy-szervérhez való kapcsolódáshoz állíts be egy *[username]:\password]* formájú sztringet a CURLOPT_PROXYUSERPWD-opció értékének:

```
curl    setopt(Scurled,CURLOPT    PROXYUSERPWD,"freda:ero4509");
```

CURLOPT_PUT
Állíts be egy nem nulla értéket a CURLOPT_PUT-nak, ha egy HTTP PUT-metódussal történő lekérést szeretnél végrehajtani. Szintén állítsd be a CURLOPT_INFILE-t és a CURLOPT_INFILESIZE-t:

```
curl_setopt($curled,CURLOPT_PUT, 1) ;
```

CURLOPT_RANGE
A CURLOPTJRANGE-opcióban beállíthatod, milyen tartományban szeretnéd a HTTP adatátvitelt használni. Használhatod az X-Y-formátumot, vagy több különböző tartományt is megadhatsz az X-Y,N-M-formátumban.

CURL OPT_REFERER
A CURLOPT_REFERER-ben megadhatsz egy sztringet, amely a HTTP-Iekérés hivatkozási fejrészét tartalmazza. A hivatkozási fejrész a szerverről származó oldallekérő információkat tartalmazza, és a PHP-szkríptekben a $HTTP_REFERER-változóban elérhető. Ha meg akarod mondani a távoli szervernek az igazat, akkor a hivatkozó mezőt az alábbiakban bemutatott módon szerkeszd meg; de az is lehet, hogy inkább a kapcsolódó oldal nevét vagy egyszerűen a domainnevedet akarod megadni:

```
curl_setopt($curled,CURLOPT_REFERER,
    "http://".$HTTP_HOST . $PHP_SELF) ;
```

CURLOPT_RESUME_FROM
A CURLOPT_RESUME_FROM-ban azt az értéket adhatod meg bájtokban, ahonnan a Curl újrakezdi az *előző* sikertelen átvitelt. Mielőtt újrakezded, nem árt ellenőrizni, hogy az outputfájl rendben van-e:

```
curl_setopt ($curled,CURLOPT_RESUME_FROM,500) ;
```

CURLOPT_SSLCERT
A CURLOPT_SSLCERT-ban egy PEM (privacy-enhanced mail, javított titkosítású levél) formátumú tanúsítvány fájlnevét adhatod meg (a PEM megkísérli beépíteni a rejtjelezést, például a PGP-t, a levél csatolt állományának MIME-szabványába):

```
curl_setopt($curled,CURLOPT_SSLCERT,"test.pera");
```

CURLOPT_SSLCERTPASSWD
A CURLOPT_SSLCERTPASSWD-ban megadhatod a CURLOPT_SSLCERT-tanúsítványhoz szükséges jelszót:

```
curl^setopt($curled,CURLOPT_SSLCERTPASSWD,"testingpw");
```

CURLOPT_SSLVERSION

A CURLOPT_SSLVERSION-ben megadhatod az általad használt SSL verziójának számát (2 vagy 3). A PHP magától is képes meghatározni a verziót, de bizonyos estekben neked kell beállítani az értéket:

```
curl    setopt($curled,CURLOPT^SSLVERSION,3);
```


CURLOPT_TIMECONDITION

A CURLOPT_TIMECONDITION-opció beállításával a CURLOPT TIMEVALUE-ér telmezését határozhatod meg. A CURLOPTJTIMECONDITION egy HTTP-sajátosság, és e két értéket veheti fel: **TIMECOND_IFMODSINCE** vagy TIMECONDJSUNMODSINCE:

```
($curled,CURLOPT TIMECONDITION,TIMECOND IFMODSINCE);
```

CURLOPTJTIMEOUT

A CURLOPT_TIMEOUT-opcióban a Curl-függvények időkorlátját állíthatod be másod percben, a Curl ennyi ideig fog várni a függvények befejezésére:

```
curl_setopt ($curled, CURLOPT_TIMEOUT, 30) ;
```

CURLOPT_ TIMEVAL UE

Adj át egy **long** típusú változót paraméterként a CURLOPT_TIMEVALUE-opciónak. A változó tartalmazza az 1970. január 1-je óta eltelt időt. A figyelembe vett időt a CURLOPT_TIMEVALUE-opciónak megadott érték vagy az alapbeállítás szerinti **TIMECONDJFMODSINCE** határozza meg.

CURLOPT_UPLOAD

Állítsd a CURLOPTJJPLOAD értékét nullától eltérőre, ha fel akarsz tölteni egy fájlt:

```
curl_setopt<$curled,CURLOPT_UPLOAD,1);
```

CURLOPT_URL

Állítsd be az olvasni kívánt URL-t a **CURLOPT__URL-be.** Ez az opció a **curl_init()** részeként is beállítható és gyakran ezt a megoldást szokták alkalmazni:

```
curl_setopt($curled,CURLOPT_URL,"http://petermoulding.com/");
```

CURLOPT_USERAGENT

A CURLOPT_USERAGENT-opcióban azt a felhasználói programot határozhatod meg, amelyet emulálni szeretnél, miközben más weblapokat böngészel. Néhány weblap a böngésző feltételezett képességei alapján változtatja az oldalak tartalmát, vagyis ha nem

szeretnéd, hogy a bejövő oldalak haszontalan Javával legyenek teletömve, akkor a felhasználói program mezőben valamilyen régebbi, Java előtti böngészőt jelölj meg. A PHP a **$HTTP_USER_AGENT**-változóba teszi a felhasználói programot. A következő példában Netscape 4.77 English-verziót állítottam be, de bármilyen nyelvet és böngészőtípust beilleszthetsz. Ha csak szöveget szeretnél kapni képek nélkül, akkor olyan felhasználói program sztringet használj, amely a Lynx böngészőnevet tartalmazza:


```
curl_setopt($curled,CURLOPT_USERAGENT,"Mozilla/4.77 [en ] (
(WinNT;U)");
```

CURLOPT^USERPWD

A szerverhez való kapcsolódáshoz állíts be egy *[username\;\password* formájú sztringet a CURLOPTJJSERPWD-opció értékének:

```
curl_setopt(Scurled,CURLOPT_USERPWD,"j oe:dgO 9xy") ;
```

CURLOPT_VERBOSE

Állíts be egy nem nulla értéket a **CURLOPT_VERBOSE-opciónak,** ha azt szeretnéd, hogy a Curl mindenről jelentést tegyen:

```
curl_setopt($curled,CURLOPT_VERBOSE,1);
```

a J

c

14. fejezet

LDAP

Gyors megoldások

Áttekintés

Látogass el a yahoo.com oldalra, kattints a Árts & Humanities-re (Művészetek és Humántárgyak), az Artists-ra (Művészek), majd az Animation@-ra, és utána a Cartoonists@-ra. Épp egy könyvtárat használtál arra, hogy karikaturistát keress. A 14.1 ábra azt a struktúrát mutatja, amelyen keresztülrágtad magad. Az LDAP segítségével pontosan ilyen típusú könyvtárkereséseket végezhetsz el.

Egy LDAP-adatbázis beállítható Yahool-stílusú könyvtárként vagy olyan könyvtárként, mint a telefonkönyv. Az LDAP az olyan adatokra megfelelő, amelyeket sokkal többször olvasnak be, mint írnak, a nagyon kis adatdarabokra és a hierarchikus szerkezetű adatokra.

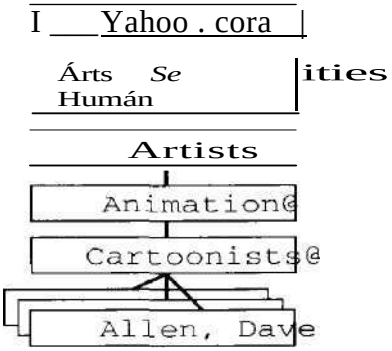
A LDAP a Lightweight Directory Access Protocol (Könnyűsúlyú Könyvtár Elérési Protokoll) rövidítése, és az X.500 Directory Access Protocol (DAP) könnyített verziója. Mennyire könnyű a könnyített verzió? A DAP az OSI hálózati modellt használja, míg a LDAP az egyszerű TCP/IP-protokollt. A TCP/IP sokkal kevesebb ráfordítással működik az Internettel, mivel a TCP/IP az Internet alapvető protokollja. A DAP mindenféle összetett adatszerkezetet, míg az LDAP majdnem minden mezó're egyszerű sztringeket használ. Az egyszerű adatszerkezetek használata következményeként a LDAP mögött álló adatbázis majdnem bármilyen lehet. Az OpenLDAP a kicsiny és hatékony Berkeley DB-t használja, amely a SleepyCat Software-től szerezhető be (www.sleepycat.com).

A hagyományos relációs adatbázisokban az adatokat *táblázatoknak* nevezett szervezett listákban tárolják, az adatbázis szoftvere különböző indexeket épít fel, és az egész úgy van kialakítva, hogy alkalmazkodjon a nagyon gyors frissítésekhez. Az LDAP egy régebbi stílusú adatbázis-szerkezetet használ, amelyet *hierarchikusnak* neveznek, ebben az adat az index. A hierarchikus adatbázisok gyors olvasási elérést tesznek lehetővé, ha ismered az indexstruktúrát, vagyis azt, hogy például milyen kategóriákon ugrálsz keresztül a Yahoo!-n. A hierarchikus adatbázisok általában lassabbak frissítésnél és új adatok bevitelénél, mert valószínűbb, hogy az adatbázis megváltoztatása új oldalak és a régi oldalakról mutató linkek létrehozását igényli.

Az XML-adatbázisok, mint a Software AG Tamino-ja, ugyanilyen hierarchikus megközelítést használnak a strukturált dokumentumok tárolására. Elvileg az LDAP használhat XML-adatbázist az adattárolásra, és strukturált dokumentumok tárolhatók LDAP-ben. Az LDAP-adatbázisok beállíthatók úgy, hogy alkalmazkodjanak a Document Object Model-hez (dokumentum objektum modell - DOM), és az egész honlapod tartalmát kiszolgálhatják. A gyakorlatban azonban az LDAP-adatbázisok túl lassúnak bizonyultak arra, hogy az egyszerű könyvtárakon túl bármire használhatóak legyenek.

A Unix és az Apache is beállítható, hogy hitelesítse egy LDAP-könyvtár használóit. Nagy a kísértés, hogy a felhasználói azonosítót és jelszót egy LDAP-könyvtárban tárold, és hozzáadd a kontakt információkat és a felhasználói profil információit, például a felhasználó által választott betűméretet. Amint a gyakori profilfrissítéseket LDAP-adatbázisba teszed, a válaszok lelassulnak, a felhasználók feszültek lesznek, és a honlapod veszít forgalmából.

Minden honlap számára jobb megközelítést jelent az LDAP csak felhasználóhitelesítésre való használata, minden mást viszont hagyományos adatbázisban tárolj. Ha MySQL-t használsz oldalaidon, használd minden olyan adat tárolására, amit a felhasználó saját maga frissíteni tud. Ha a felhasználói profil információkat mégis LDAP-ben tárolod, és az információt minden egyes weboldal formázására felhasználod, kerüld el a profil állandó LDAP-ből való beolvasását azzal, hogy a profilt a felhasználó session-rekordjába másolod. Csak akkor térj vissza az LDAP-hez, amikor a felhasználó felfrissít valamit a profiljában.



14.1 ábra A Yahool-címtár könyvtárszerkezete

Telepítés

Van egy Gyors megoldás a fejezetben, amely az LDAP Windows NT alatti telepítését taglalja, beleértve az OpenLDAP-szerver telepítését is. A Windows 2000 alatti telepítés megegyezik a Windows NT alattival. A Windows 95 és 98 felhasználóinak nem találtam jó és átfogó megoldást. A Unix felhasználóknak egy LDAP kliens könyvtárat kell a www.openldap.org/- vagy **http://developer.netscape.com/tech/directory/-oldalakrol** telepíteni, majd a -with-ldap-opcióval be kell a PHP-t fordítani (az LDAP rajta lehet a Unix-dsztribúció CD-jén).

Terminológia

Az LDAP terminológia egyedi jelentéseket használ, amelyeket ismerned kell, mielőtt az LDAP részletes leírását elkezded olvasni. Az LDAP-könyvtáron keresztüli navigáció az olyan kifejezések LDAP-beli használatának megértésétől függ, mint az *attribútum* és *objektum.*

DN - megkülönböztetett nevek

Az LDAP megkülönböztetett neveket (distinguished names - DN) használ az azonosítás alapegységeként. A DN olyan azonosítók gyűjteménye, amely az általad kívánt elérési utat építi fel. Itt van a Yahoo! karikaturistás példa legfelső szintű" DN-je. A de a > *domainkomponens* (domain component) rövidítése. A yahoo.com-nak két domain komponense van, a **yahoo** és a **com:**

dc=yahoo, dc=com

■

Az és tesztoldalam a test.petermoulding.com, így ezt a következő DN-nel érem el, három domainkomponenst használva:

dc=test, dc=petermoulding, dc=com

Ha a honlapodnak több komponense van, több dc = -t kell használnod. A mail.test.petermoulding.com.au-hoz 5 domainkomponens szükséges.

A DN szabványos komponenseinek előzetesen meghatározott azonosítói vannak. Egy ország nevét c = -vel azonosítják. Egy cég vagy szervezet nevét o=-vel. A felhasználói neveket cn=-vel (common name). Megadhatod a saját mezőazonosítóidat, de ne írd felül a gyakran használt azonosítókat, mert ez összeomláshoz vezet, amennyiben az LDAP-könyvtáradat más LDAP-könyvtárakhoz csatlakoztatod.

A Yahoo!-s példánál maradva nevezd el a különböző szinteket így: levels ca (category - kategória), se (subcategory - alkategória), ss (subsubcategory - alalkategória), ni (next levél -következő szint) és cn (common name - köznév). A kategóriaszintre vontakózó kérés a következő DN-t használja:

ca=Arts & Humanities, dc=yahoo, dc=com

Vedd észre a különbséget a Yahoo! kategóriamegjelenítés és egyéb strukturált könyvtárinformáció, például egy fájl elérési útja között. Az LDAP DN-t jobbról balra kell olvasni, így a legfelső szintű könyvtár van a jobb oldalon, és az alacsonyabb szintek balról adódnak hozzá. Az alkategória szintre vonatkozó kérés a következő DN-t használja:

sc=Artists, ca=Arts & Humanities, dc=yahoo, dc=com

A gyakori név szintre vonatkozó kérés az alábbi DN-t használja:

cn="Allen, Dave", nl=Cartoonists@, ss=Animation@, sc=Artists, ca=Arts & _ Humanities, dc=yahoo, dc=com

A Yahoo .'-könyvtárban a végső elem egy link a honlapra annak leírásával, így végső soron egy kicsit eltér az általános LDAP könyvtárhasználattól, ahol a végső elem egy olyan kontakt információ, mint a név és telefonszám. Vedd észre a speciális karaktereket, például vesszőket tartalmazó mezőértékek körüli kétszeres idézőjelet,". Van még néhány dolog, amit nem szabad elfelejteni a megkülönböztetett nevek használatakor. Bármelyik mező tartalmazhat értékek listáját; egy "Allén, Dave"-re vonatkozó kérés számtalan "Allén, Dave"-nek nevezett ember bejegyzését adhatja vissza. Az adatbázis beállítható, úgy hogy az "Allén, Dave"-et fn = Dave, sn = Allen-ként tárolja (first name - keresztnév, surname - vezetéknév).

RDN - relatív megkülönböztetett nevek

A DN-ek bői RDN-ek (relatíve distinguished names - relatív megkülönböztetett nevek) lesznek, amikor egy LDAP-könwtár struktúráján belül használják őket. Talán már hozzászoktál, hogy egy oldalra ./index.html-ként hivatkozol, amikor egy weboldal alkönyvtárából kéred az oldalt (ha /test/-könyvtárban vagy, és a ./index.html-t kéred, a /test/index.html-t kapod). Egy RDN ugyanígy működik. Ha egyszer kiválasztod az sc=Artists, ca=Arts & Humanities, dc=yahoo, dc=com-ot, utána kérheted a **cn="Allen,** Dave", nl=Cartoonists@, ss = Animation@-t.

Szintek

Ahogyan a yahoo.com kategóriái között vagy egy lemezpartíció könyvtáraiban is egyre lejebb lépkedhetsz, ugyanúgy megteheted ezt az LDAP-könyvtár *szintjei* között is. Minden szintnél az azonosítók egy listáját kéred meg, és egy azonosítót választasz, hogy a következő szintre vigyen.

Van különbség aközött, ha LDAP-könyvtárban vagy fájlkönyvtárakban lépkedsz. Amikor fájlkönyvtárakon lépkedsz keresztül, minden szint ugyanolyan típusú elemeket tartalmaz: könyvtárakat és fájlokat. Az LDAP-ben minden szint teljesen másmilyen lehet. Egy lemez fájlstruktúrájában egy könyvtár fájlok és könyvtárak listáját tartalmazhatja, de egy fájlnév nem vonatkozhat egynél több fájlra. Ha egy lemez fájlstruktúráját LDAP-re cserélnénk, a fájlok is tartalmazhatnának listát, s ez duplikált fájlokhoz vezethetne, ami lehetetlenné tenné a pontos fájlvisszakeresést.

Elemek

Egy LDAP-könyvtár minden egyes hivatkozási pontját elemnek, csomópontnak vagy *megkülönböztetett szolgáltatási elemnek* (distinguished service entry - DSE) nevezik. Hasonló egy relációs adatbázis rekordjaihoz, mert egyedi azonossága van és mezőket tartalmaz. Viszont egy LDAP-elemben minden mező tartalmazhatja értékek listáját, és lehet alacsonyabb szintű elem.

A Yahoo!-s példában az sc=Artists, ca=Arts & **Humanities, dc=yahoo,** dc=com-ként azonosított csomópont tartalmazhat mezőket és alacsonyabb szintű elemeket, például az ss = **Animation@-t.** Amikor végignézel a Yahoo! kategóriáin, minden oldalon leíró szöveget és hirdetési bannereket, alkategóriák listáját és a szintnek megfelelő elemek listáját találod. Az **LDAP** képes ilyen sokféle információ tárolására, de az információn keresztüli navigáció problémás is lehet, mert minden alkalommal, amikor egy elemet keresel vissza, minden megkapott mezőt ellenőrizned kell, hogy nem lista-e vagy nem tartalmaz-e alacsonyabb szintű elemeket.

Attribútumok

Ahogy a relációs adatbázis rekordjainak vannak mezői, az egyes elemeknek vannak *attribútumai.* A Yahoo!-s példában "Allén, Dave" karikaturistának az **Allén, Dave** köznévattribútuma van. Lehetne e-mailcím vagy honlapattribútuma, vagy keresztnév- és vezetéknév-attribútuma a köznév helyett. Tulajdonképpen semmi akadálya nincs, hogy legyen keresztneve, vezetékneve, közneve, illetve más névformák, mint például megszólítás (salutation — sl). Az eleme lehetne egy a következőhöz hasonló sor:

```
fn=David,  sn=Allen,  cn=Dave Allén,  sl="Mr D. Allén"
```

Attribútumot bármikor hozzáadhatsz, az egyéni elemek attribútumait kihagyhatod, és variálhatod, hogy az egyes elemekhez milyen attribútumokat használsz. Ha létrehoznád az egész világot tartalmazó telefonkönyvet, abban 6,5 milliárd elem lenne, és csak néhány százmilliónak lenne e-mailcíme, és nem mindenkinek lenne irányítószáma, az irányítószámok formátuma pedig országonként változó lenne.

Objektum

Az LDAP-könyvtár egy *objektuma* definiálja az attribútumokat egy elem révén az összes olyan elemre, amely ugyanahhoz az objektumosztályhoz tartozik. A Yahoo!-s példában az összes karikaturistának ugyanaz az objektuma, ami meghatározza, hogy mi lehet egy karikaturista elemében. Természetesen a legtöbb *mező* opcionális, és néhány mezőben lehet azonos vagy keverhető érték, így egy objektum definícióját rendkívül rugalmasra állíthatod be, akár a határozatlanságig is eljutva. Az objektumdefiníciónak tervezettnek kell lenniük, kicsit mintha normáznád az adatbázisrekordokat, biztosítva, hogy ne legyen adat-duplikáció.

DIT

Az összes objektumot tartalmazó LDAP-könyvtár szerkezetét *Könyvtár Információs Fának* (Directory Information Tree - DIT) nevezik.

Séma

A *séma* egy LDAP **DIT** és a DIT-ben levő objektumok formális definíciója. Következzék egy LDAP-séma, az OpenLDAP core.scheme-jének gyors áttekintése. Ha többet akarsz tudni az LDAP-sémákról, rágd át magad a **http://openldap.org** honlap összes linkjén.

A következő kód egy attribútumtípus alapvető definíciója az **attributetype-utasítással.** A **2.5.4.41** egy a sémán belül egyedi azonosító. A **NAME** ennek a típusnak a nevét jelöli. Az **EQUALITY** egy rutin nevét adja, amelyet az ilyen típusú mezőben levő érték más értékkel való összehasonlítására használnak annak megállapítására, hogy egyeznek-e. A **caselgnoreMatch** azt jelenti, hogy az összehasonlító rutin nem tesz különbséget a kis- és nagybetűk között. A **SUBSTR** az értékek alsztringjén alapuló összehasonlításra használt rutin. A **SYNTAX** nagy része értelmetlennek és az öszes mezőben azonosnak tűnik. A SYNTAX utolsó része, a **15{32768}** 15-ös hosszúságot jelent, 32768-ig terjedő változó hosszúsággal:

```
attributetype ( 2.5.4.41 NAME 'name'
    EQUALITY caselgnoreMatch SUBSTR
    caselgnoreSubstringsMatch SYNTAX
    1.3.6.1.4.1.1466.115.121.1.15(32768} )
```

Az attribútumtípusok újrahasználhatok és kiterjeszthetők. A **NAME** újra használtuk a következő két attributetype-definícióban. Vedd észre, hogy mindkettőnek van egy rövid és egy hosszú neve:

```
attributetype ( 2.5.4.3 NAME ( 'cn' 'commonNarae' )
SUP name ) attributetype ( 2.5.4.4 NAME ( 'sn' 'surname'
) SUP name )
```

A következő **attributetype SINGLE-VALUE-t** ad hozzá, hogy megakadályozza azt, hogy a mezők értékek listáját kapják:

```
attributetype ( 2.5.4.6 NAME ( 'c' 'countryName' ) SUP name
SINGLE-VALUE )
```

A séma **objectclass**-definíciókat is tartalmaz, de remélem, ezeket nem kell megtanulnod. Elég, ha ki tudod terjeszteni az attributetype-okat, hogy megkapd a szükséges mezőket.

LDIF

Amikor egy LDAP-könyvtárba elemet küldesz, vagy visszakeresel belőle, az elemet LDAP Data Interchange Formát (LDIF), egy szöveg alapú formátumban formázod. Ha egy LDAP-elem bináris információt tartalmazó attribútumot foglal magába, a bináris információnak base64 kódolásúnak kell lennie.

Az LDIF egy lehetséges alternatívája az XML alapú Directory Services Markup Langüage (DSML), amelynek leírását a http://dsml.org-oldalon találod. Az LDAP-szerverek az LDAP-protokoll használatával kommunikálnak egymással és a kliensekkel. Amikor az **ldap_bind()-et** használod (ahogy azt a „Csalakozás az LDAP-hez" című Gyors megoldásban elmagyarázom), és egy URL-lel mutatsz egy LDAP-szerverre, a protokollt így határozod meg: **ldap://.**

Szerverjellemzők

Van előnye az LDAP-nek az alternatív adatbázisokkal szemben? Kiterjeszthetsz egy LDAP-könyvtárat egy szerveren túlra is? Az LDAP-szerverek következő jellemzői lehetővé teszik, hogy LDAP-szervered egy LDP-szerverekből álló világhálózatra terjeszd ki. Sőt, csak kevés adatbázis nyújt az LDAP küldési tulajdonságával egyenértékű lehetőséget.

Küldés

Egy LDAP-szerver a küldés segítségével oszthatja meg egy másikkal a munkát. Ha a yahoo.com LDAP-t használna, és a **yahoo.com.an** kezelné az összes Holland Antillákkal kapcsolatos elemet, a yahoo.com minden c=an-t (country = Netherlands Antilles - Holland Antillák) tartalmazó kérést a **dc=yahoo, dc=com, dc=an (yahoo.com.an)**-ra küldene.

A küldés segítségével számtalan szerver tudja a munkát megosztani, az egyes szerverekre érkezett kérések mennyisége vagy egy a DlT-n belüli csomópont alapján. Ha a Yahoo! Árts & Humanities részlege teljesen más szerkezetű, mint a Yahoo! többi része, az ide érkező kérések egy teljesen külön, a Yahoo! többi részét kezelő szervertől eltérő konfigurációval rendelkező szerverre mennek.

Másolás

Ha több mint egy LDAP-szerver van, hogy megosszák a munkát, az LDAP-szerverek mindegyikére le kell másolnod az adatokat. Ha húsz aktív szervered van, vegyél egy huszonegyediket a frissítésekhez. Bárki bármilyen frissítést ad a könyvtárhoz, a változtatást ezen a huszonegyedik szerveren végezd el. Majd innen másold át az összes többi szerverre. Mivel a másolás csak 3 percig tarthat, óránként minden szerverre felmásolhatod a változtatásokat.

Biztonság -.insisr-

Az LDAP alapprotokollja az LDAP, de ha az LDAP-szervereden telepítve van a Secure Sockets Layer (SSL) vagy az ennél újabb Transport Layer Security (TLS), a biztonság érdekében az LDAP-t felcserélheted LDAPS-re. Ha már dolgoztál HTTP-vel és HTTPS-sel, akkor mindent tudsz az LDAPS-ről. Ha még nem dolgoztál a HTTPS-sel, jobb, ha az LDAPS előtt azzal próbálkozol.

Mind a HTTPS, mind az LDAPS az SSL biztonsági igazoló telepítését igényli, amellyel ellenőrizheted, hogy milyen protokollokat használsz a honlapodon. Ha valaki HTTP-vel éri el a honlapod, akkor az LDAP-t LDAP-vel érd el. Ha egy felhasználó bejelentkezik és elkezd HTTPS-sel kapcsolódni a honlapodhoz, mérlegelned kell az LDAPS használatát. Lehet, hogy a felhasználó a bejelentkezés után csak nyilvános adatokat keres, így nincs szüksége az LDAP-hez való biztonságos hozzáférésre, de az is lehet, hogy az LDAP-könyvtár egy csak tagoknak szóló részéhez akar hozzáférni, és ekkor igényli a biztonságot.

LDAP-függvények

Ez a rész a Gyors megoldásokban nem említett LDAP-függvényeket mutatja be.

Az **ldap_compare()** csatlakozási azonosítót, a könyvtár egy elemére mutató megkülönböztetett nevet, egy attribútum nevét és egy értéket fogad el, amelyet összehasonlít a könyvtárban talált attribútummal. A függvény igazat ad vissza, ha a megadott érték megegyezik a könyvtárban levő értékkel, hamisat, ha nem egyezik, és -1-et, ha hiba történik.

Az **ldap_delete()** csatlakozási azonosítót és egy elem megkülönböztetett nevét fogadja el. Az **ldap_delete()** törli az elemet, igazat ad vissza, ha sikerült a törlés, hamisat, ha nem.

Az **Idap_dn2ufn()** egy megkülönböztetett nevet fogad el, és az értékeket formázott sztringben adja vissza, a típusnevek eltávolításával. Az **Idap_dn2ufn()-t** a következő kóddal teszteltem:

```
print{ldap dn2ufn("dc=test, dc=petermoulding, dc=com"));
```

Az eredmény az alábbi sor lett, ami számomra nem tűnik túlzottan felhasználóbarátnak. Talán egy későbbi változatban a kód képes lesz a többszörös komponensekből létrehozni a honlapok címét:

```
test, dc=petermoulding, dc=com
```

Az **ldap_explode_dn()** egy megkülönböztetett nevet és egy formázási opciót fogad el. A megkülönböztetett nevet komponensekre bontva adja vissza, minden komponenst egy tömb egy elemeként tárolva. Ha a második paramétert 0-ra állítjuk, meghagyja a komponenseknek az attribútum nevét, például **c=FR,** ha pedig l-re, akkor levágja az attribútum nevét, és csak az értéket hagyja, például FR.

A következő kód — 0-ra állított második paraméterrel — kibontja a $dn-t a $array-tömbbe, és megjeleníti a kapott tömböt:

```
$dn = "dc=test, dc=petermoulding, dc=com";
Sarray = ldap explode dn($dn, 0); while (üst ($k,
$v) = each ( $array) )

    print("<br>k:       $k       v:       $v ) ;
```

íme az eredmény, amely azt mutatja, hogy a tömb ilyen formában akkor a leghasznosabb, amikor a megkülönböztetett névkomponensekből egyéb megkülönböztetett neveket akarsz létrehozni:

```
k: count,    v:    3
k: 0,    v:    dc=test
k: 1,    v:    dc=petermoulding
k: 2,    v:    dc=com
```

A következő kód ugyanazt a $dn-t bontja ki a $array-tömbbe, de a második paraméter l-re van állítva, és így jeleníti meg a tömböt:

```
$array    =    ldap_explode_dn($dn,
1); while (üst ($k,    $v)    =    each (
Sarray) )

    print("<br>k:          $k                    $v
                                                );
```

íme az eredmény, amely azt mutatja, hogy a tömb ilyen formában akkor a leghasznosabb, amikor az értékeket a felhasználónak akarod megmutatni:

```
k: count,    v:    3
k: 0,    v:    test          ,-nit
k: 1,    v:    petermoulding
k: 2,    v:    com
```

Az ldap_search()-függvényt használom a „Gyors megoldásokban", de itt mutatom be, mert az általa visszaadott értékre szükség van az Idap_first_entry()-függvényben Az ldap_search() csatlakozási azonosítót, a keresés kezdőpontját jelölő megkülönböztetett nevet és egy keresési sztringet fogad el. Keresésieredmény-azonosítót ad vissza az ldap_first_entry()-nek, vagy hamisat, ha a keresés nem vezet eredményre. A következő kód munka közben mutatja meg az ldap_search()-öt, amikor a c=* keresési sztringgel keres rá az országra:

```
$ldap["dn"]    =    "dc=test,    dc=petermoulding,
dc=com"; $ldap["search"]    =    "c=*";
if($search    =    ldap_search($ldap["connection"],
    $ldap["dn"], $ldap["search"]))
                                                (f {11 rt vJ/sft qr,bí
                                                íi.r'.913.>J TJobjCÍ," é  ■ /
    //    Insert    code    here:

else

    print ("<br>LDAP    search    failed    for    "    .    $LDAP["search"]) ;
```

Az ldap_search() keresési sztringjének rugalmas a szintaktikája, de sajnos - ellentétben az SQL where megkötéseivel, amit programozók milliói megértenek - nem szabványos for-

mátumú. Jó néhányszor el kell olvasnod az LDAP dokumentációját ahhoz, hogy akár a legegyszerűbb keresésed működjön. A legújabb PHP-ban van néhány opcionális paraméter, amelyekkel korlátozhatod a visszaadott elemek számát, kiválaszthatod, hogy melyik attribútumot adja vissza, illetve egyéb szabályozókat kezelhetsz. Azt gondolom, egyszerűbb lett volna, ha az SQL-struktúrát használták volna, olyan keresési kifejezésekkel, mint az SQL **start** és **limit.** Ehelyett egy másik szintaktikát is meg kell tanulnod. A keresési sztringgel való problémák azt jelentik, hogy egy hagyományos relációs adatbázis használata jobb megoldás, hacsak nem kompatibilitási okok miatt vagy kénytelen LDAP-t használni.

Amikor keresési sztringeket tesztelsz, azt találhatod, hogy bizonyos elemek és attribútumok hiányoznak, mert az LDAP-könyvtárnak beépített korlátai vannak arra vonatkozóan, hogy miket kereshetsz vissza. Némely LDAP-szoftverek alapértelmezésben mindent visszaadnak, ha csak az opcionális paraméterekkel nem korlátozod a kérést. Más adatbázisokban beépített korlátozások vannak, amit az opcionális paraméterekkel kell felülírni, és lehet, hogy némelyikük úgy van beállítva, hogy nem tudod felülírni őket.

Az ldap_count_entries() a keresés által visszaadott elemeket számolja, és a keresés kezelésére kialakítandó stratégia tervezéshez használható. Ha a keresési input egy online űrlapból való, jelenítsd meg az első 10 találatot és egy üzenetet, amely az összes megtalált elem számát jelzi, így a felhasználó dönthet, hogy pontosítja vagy tágítja a keresést, a találatok számának függvényében. A **ldap_count_entries()** egy csatlakozási azonosítót és az **ldap_search()** keresésieredmény-azonosítóját fogadja el, és a számolás eredményét adja vissza, vagy hamisat, ha hiba történt.

Az **ldap_first_entry()-, ldap_next_entry()-, ldap_first_attribute()-** és az **ldap_next_attribute()-függvényekkel** nagy LDAP-könyvtárakon tudsz elemenként és attribútumonként végigmenni anélkül, hogy az **ldap_get()**-függvénysorozat használatánál előforduló memória-többletráfordításokkal számolnod kellene. Az ldap_get-függvények egyszerre olvasnak be mindent egy nagy tömbbe, így, hogy egy nagy terjedelmű keresés lehet az összes rendelkezésre álló memóriát felhasználná. Az **ldap_first-** és az **ldap_next-**függvényekkel az egyes elemekhez csak néhány attribútumot választva tudsz a keresési eredményeken végigfutni, és az egyes elemeknél a választást megváltoztathatod.

Az **ldap_first_entry()** egy csatlakozási azonosítót és az **ldap_search()** keresésieredmény-azonosítóját fogadja el, és az eredmény első elemének elemazonosítóját adja vissza, vagy hamisat, ha a függvény nem tud elemazonosítót visszaadni. A **ldap_next_attribute()** egy csatlakozási azonosítót és az **ldap_first_entry()** által visszaadott elemazonosítót fogad el, és a következő elem azonosítóját adja vissza. A következő kód a **ldap_first_entry()** és az **ldap_next_entry()** működését mutatja meg, ahogy végiglépkednek a **ldap_search()** által visszaadott keresési eredményeken, a **$search-ön.** A **$entry** adja át az eredményeket az **ldap_next_entry()** egyik előfordulásából a másikba, amíg az **ldap_next_entry()** hamisat nem ad vissza. A **display**_entry()-függvényt az egyes elemek megjelenítésére hoztam létre. Bármilyen kódot hozzáadhatsz ezen a ponton, akár a későbbi, **ldap_first_attribute()-ot** és **ldap_next_attribute()-ot** tartalmazó kódot is:

```
$entry = ldap_first_entry($ldap["connection" ], $search);
while($entry !== falsé)

    print("<br>ldap_first_entry search: " . $search );
    display_entry($ldap["connection"] , $entry);
    $entry = ldap_next_entry($ldap["connection"j , $entry)
```

Az **ldap_first_attribute()-** és az **ldap_next_attribute()**-függvényekkel az **ldap_first_entry()**-és ldap_next_entry()-függvények által visszaadott elemek attribútumain lépkedhetsz **végig. A ldap_first_attribute()** egy csatlakozási azonosítót és az **ldap_first_entry()** vagy az **ldap_next_entry()** által visszaadott elemazonosítót, illetve egy a példában **$next>nek nevezen** segédmezőt fogad el. Egy attribútumot tartalmazó sztringet ad vissza, vagy pedig hamisat, amikor nincsen több attribútum. Az **ldap_next_attríbute()** csatlakozási azonosítót, az **ldap_first_entry()** vagy az **ldap_next_entry()** által visszaadott elemazonosítót és egy **Snext** nevű segédmezőt fogad el. Az attribútumot tartalmazó sztringet ad vissza, vagy pedig **hamisat,** amikor nincs több attribútum. A $next biztosítja az **ldap_first_attribute()** és az **ldap_next_attribute(),** illetve az **ldap_next_attribute()** előfordulásai közötti **kommunikációt:**

```
$next   =   "";
§attribute = ldap_first_attribute($ldap["connection"], $entry, $next);
while(isset($attribute) and $attribute !== falsé and isset ($next))
    {
    print ("<br>ldap_first_entry search: " . $attribute);
    $attribute = ldap_next_attribute($ldap["connection"], $entry, $next);
```

A **Snext** nevű segédmezőre az **LDAP dokumentációjában ber_identifier-ként hivatkoznak,** és egy belső memórialokációs pointerként írják le. Mindenféle weboldalt kipróbáltam, hogy a ber_identifier pontos definícióját megtaláljam, de nem sikerült. Amikor kipróbáltam **a** kódot, az Apache összeomlott. Az **ldap_first_attribute()** értelmetlen valamit adott vissza az ldap_next_attribute()-nak. Kerüld ezeket a függvényeket, és használd a **ldap_get_attributes()-t.**

Az **ldap_free_result()** felszabadítja a keresési eredményekre kiosztott memóriát. **Nem kell** az eredményhalmazt felszabadítani, ha csak egyet használsz a szkriptedben, **mivel a szkript** végén az eredmények automatikusan felszabadulnak. Az **ldap_free_result()** használata akkor jó, ha számtalan nagy eredményhalmaz van egy szkriptben, és nincs szükséged **arra,** hogy mind egyszerre legyen nyitva.

Gyors megoldások

LDAP Windows NT alatti telepítése

Az LDAP Windows NT alatti telepítéséhez egy LDAP-szerverre és a PHP LDAP-kiterjesztésére van szükséged. Kezdj a PHP LDAP-kiterjesztéssel, majd próbálj ki egy opciót az LDAP-szerverre.

PHP-kiterjesztés

Az én Windows NT munkaállomásomon a PHP 4.0.7dev a c:/Program Files/php/-könyv-tárba van telepítve, így a PHP alkönyvtárai ehhez a könyvárhoz viszonyítottak. Az első lépés a PHP LDAP-kiterjesztésnek a PHP-kiterjesztések alkönyvtárból a c:/winnt/system32/-be való másolása. A PHP LDAP egy extra DLL-t igényel, így másold a PHP dlls-alkönyvtárból a Libsasl.dll-t a c:/winnt/system32/-be.

Ha a PHP-t Apache-modulként futtatod, állítsd le az Apache-ot. Szerkeszd át a php.ini-t, eltávolítva a pontosvesszőt (;) a következő sorból, majd mentsd el a php.ini változását. (A php.ini-ben egy sor előtt levő pontosvessző megjegyzéssé változtatja a sort.) Ha le kellett az Apache-ot állítanod, most indítsd újra:

```
extension=php    ldap.dll                                    "'    "'
```

Ha IlS-t vagy másmilyen webszervert használsz, akkor kövesd a PHP- vagy a webszerve-red telepítéséhez mellékelt utasításokat.

Open LD AP-szerver

Az NT-szerverek némely disztribúciói a levelező szerver részeként tartalmazzák az LDAP-t. A Microsoft Small Business Szervere gyors és költséghatékony megoldást jelentett az olyan egyetlen szervert használó vállalkozásoknak, amelyek mindent arra telepítettek fel. A Microsoft forgalmazza a Microsoft ActiveServer Directory-t, amely hasonló az LDAP-hez, de a PHP LDAP-függvényeivel nem érhető el. Ha nem akarod a Microsoft levelezőcsomagját, vagy hozzám hasonlóan olyan megoldásra van szükséged, amely illeszkedik az ügyfeleid Unix-szerverén telepített LDAP-hez, az **openldap.org-ról** beszerezhető OpenLDAP-re van szükséged.

Olvasd el a **php/tools/ldap/** PHP-könyvtárban lévő readme.txt-fájlt. Van ott egy link a www.fivesight.com/downloads/openldap.asp-oldalra, amely egy letölthető Windows NT/Windows 2000 bináris állományt tartalmaz, illetve információt ad arról, hogyan lett a bináris állomány lefordítva. A letöltés egy tar.gz-fájlba van tömörítve, amely WinZippel kitömöríthető. A falj tartalmát c:/Program Files/OpenLDAP/-könyvtárba tömörítsd ki.

Módosítsd a slapd.conf-ot úgy, hogy a könyvtárnevek illeszkedjenek a rendszeredhez. Az alábbi négy sort kell megváltoztatnod (és ezeket kellene a későbbi verziókban megváltoz-

tatni). A **%SYSCONFDIR%** az OpenLDAP-t tartalmazó könyvtár egy alkönyvtárára mutat. A **%LOCALSTATEDIR%** az OpenLDAP-t tartalmazó könyvtár egy alkönyvtárára mutathat, vagy tesztelés esetén a tesztlemez egy új könyvtárára:

```
include         %SYSCONFDIR%/schema/core.schema
pidfile         %LOCALSTATEDIR%/slapd.pid
argsfile        %LOCALSTATEDIR%/slapd.args
directory       %LOCALSTATEDIR%/openLDAP-ldbm
```

A következő sorok az általam használt konfigurációt mutatják, a **%LOCALSTATEDIR%** egy új könyvtárra mutat azon a lemezen, amit az Internethez kapcsolódó szoftverek és weboldalak tesztelésére használok:

```
include         "c:/program Files/OpenLDAP/schema/core.schema"
pidfile         i:/OpenLDAP/slapd.pid argsfile
i:/OpenLDAP/slapd.args directory    i : /OpenLDAP/openLDAP-ldbm
```

A konfigurációs fájl következő változtatása a honlapod címének hozzáadása a következő sorral:

```
suffix    "dc=my-domain,    dc=com"
```

Az én élesben működő oldalam a **petermoulding.com,** így én a slapd.conf-ot így állítom be:

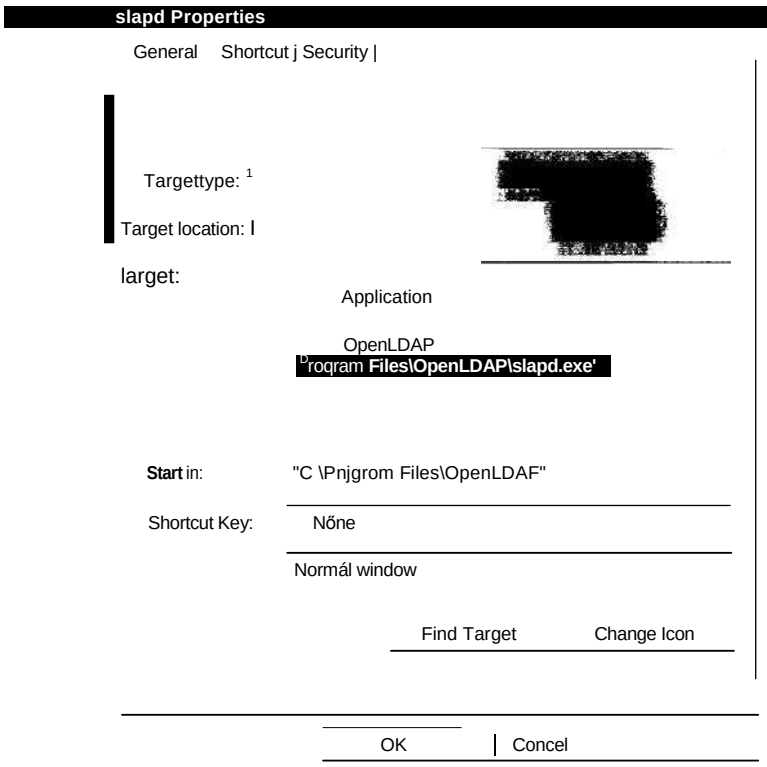
```
suffix    "dc=petermoulding,    dc=com"
```

A tesztoldalam **test.petermoulding.com,** így a tesztmunkaállomást a következő sorral állítom be, három **dc** = -t használva. Ha az oldalam címének több komponense lenne, több dc = -t kellene használnom. Például a **mail.test.petermoulding.com.au-hoz** öt **dc=** szükséges:

```
suffix    "dc=test,    dc=petermoulding,    dc=com"
```

Az LDAP-szerver indításához hozz létre egy a slapd.exe-re mutató parancsikont. A parancsikon tulajdonságait a 14.2 ábra mutatja. A parancsikon az LDAP-könyvtárból indítja a szénért, így az LDAP-könyvtárban levő bármilyen speciális DLL-ek felülírják a rendszerkönyvtárakban levő ugyanolyan nevű DLL-eket.

Indítsd el a szervert a parancsikonra való kattintással. A slapd egy parancs prompt ablakban fog felbukkanni, ahogy azt a 14.3 ábra mutatja. Ha végeztél a slapd-del, csak zárd be a parancs prompt ablakot.



14.2 ábra LDAP-szerver indító parancsikonjának beállításai

14.3 ábra Parancssorablakban induló LDAP-szerver

A szerver tesztelése

Az LDAP-szervert a következő kóddal teszteld. Az ldap_connect() csatlakozási azonosítót ad vissza, illetve hamisat, ha a csatlakozás nem sikerül:

```
if($connection=ldap_connect("localhost"))
   print("<br>LDAP connection: " . $connection);
else
   print("<br>LDAP connection failed.");
```

Az eredmény:

```
LDAP connection: Resource id #2
```

Az LDAP bináris állományt saját magad is lefordíthatod, ha van Visual C + + -od. A Windows NT-hez való nyílt forráskódú szoftverek fele nyílt forráskódú C fordítót használ, má-

sik fele pedig Visual C+ +-t. Én igyekszem elkerülni az olyan szoftver fordítását, amely Visual C + +-t használ, mert a szoftver hajlamos a DLL-ek speciális verzióit használni, amelyek már a rendszerkönyvtárakban vannak. A **http://fivesight.com** weboldalon van egy figyelmeztetés a msvcrt.dll-ről, illetve instrukció arra vonatkozólag, hogy hogyan kerüld el a problémát az LDAP slapd más LDAP-könyvtárból való indításával.

A következő' megoldás az LDAP-szerverekhez való csatlakozást bővíti ki.

Csatlakozás az LDAP-hez

Az LDAP használatához csatlakoznod kell egy LDAP-szerverhez. Az előző Gyors megoldásban a csatlakozás a localhost-hoz történt, és az LDAP-szerver úgy volt konfigurálva, hogy a **test.petermoulding.com-oldalra** mutasson. Ebben a megoldásban ugyanehhez a szerverhez lesz a csatlakozás, de a **test.petermoulding.com** nevű oldalt fogom használni. A csatlakozási segédprogramok használatával bármilyen, a PHP-szkriptet futtató szerverről látható szervert elérhetsz, és az LDAP-szerverek konfigurálhatóak úgy, hogy a kéréseket más szervereknek továbbítsák, így egy csatlakozással szerverek teljes hálózatán kereshetsz.

Egy LDAP-csatlakozás az **ldap_connect()-tel** kezdődik a szerver elérésére, ezt az ldap_bind()-parancs követi, ami nem más, mint az LDAP-szerverre való bejelentkezés. Elvégzed a szükséges LDAP-műveleteket, majd az **ldap_close()-** és **ldap_unbind()**-utasításokkal lezárod az LDAP-szerverhez való csatlakozást. Ez a megoldás a csatlakozási, össze-kapcsolódási és bezárási folyamatokat foglalja magában. A későbbi megoldások mutatják be a műveleteket, amelyeket a szerverhez való csatlakozás és összekapcsolódás után végezhetsz el.

Az első lépés a csatlakozás. A következő kód a **ldap_connect()** és az oldal nevének használatával egy egyszerű csatlakozást hajt végre:

```
$ldap["site"]     =     "test.petermoulding.com";
if($ldap["connection"]     =     ldap_connect($ldap["site"]))

    print("<br>LDAP    connected              $ldap["site"]);
    to

else {
    print("<br>LDAP    connection    failed
    for                                                $ldap["site"]);
```

Az **ldap_connect()** egyetlen opciója a port számát tartalmazó második paraméter, amely alapértelmezésben 389. A következő kód az előző kód első két sora, kibővítve a port számával. Erre akkor lehet szükséged, ha több LDAP-szerver van ugyanazon a gépen:

```
$ldap["port"]     =     138 9;
$ldap["site"]     =     "test.petermoulding.com";
if($ldap["connection"]     =                                    $ldap["port"]))
ldap_connect($ldap["site"
```

A PHP 4.0.4-től felfelé a **ldap_connect()** elfogad URL-t a honlap neveként, így az LDAP-szervereket az Internet bármely részéről elérheted. A következő az első kód URL-re való megváltoztatása: ,,iItinifv

```
$ldap["site"]     =     "http://test.petermoulding.com";
if($ldap["connection"]     =     ldap_connect($ldap["site"] ) )
```

A **http://test.petermoulding.com** nem érvényes URL egy LDAP-szerverhez, így a következő hibát kapod, ugyanazt, amit akkor kapsz, ha az oldal neve vagy az URL hibás. A hiba akkor is előfordulhat, ha a hálózaton belül a válasz túl lassú:

```
Warning: Could not create LDAP session handle (3): Time limit
exceeded Figyelem: Nem sikerült az LDAP session-kezelö
létrehozása(3): Időkorlát túllépése
```

A **ldap://test.petermoulding.com** a megfelelő URL egy LDAP-szerverhez. Az URL-sé-**mának** (más néven protokollnak) ldap-nek kell lennie, ahogy a következő kód is mutatja:

```
Sldap["site"]     =     "ldap://test.petermoulding.com";
```

Ha a biztonság érdekében az LDAP-szerveredre az SSL telepítve van, cseréld a **ldap-t** ldap$-ra, mint itt:

```
$ldap["site"]     =     "ldaps://test.petermoulding.com";
```

Ha már van érvényes csatlakozásod, az **ldap_bind()** használatával kell az LDAP-szerverrel összekapcsolódnod. A következő kód a minimum, ami az **ldap_bind()-hoz** elegendő, és csak akkor, ha az LDAP-szerver konfigurálva van a névtelen összekapcsolódásra. A nyilvános LDAP-szervereket, amelyek szaknévsor vagy telefonkönyv stílusú könyvtárakként vannak beállítva, olvasásra általában el lehet érni névtelen összekapcsolódással, de a változtatások végrehajtásához jelszó szükséges:

```
if ($ldap ["connection"] )

    if(ldap_bind($ldap["connection"]))

        print ( "<br>LDAP    anonymous    bind    worked."); "í. i '. ■=■*" ! q?í\

    else

                                                          " í>"} íft
        print("<br>LDAP anonymous bind failed.");          ■
```

```
                                                    . »fU!OO
Az eredmény:                .      „    ■..■.,..,■■■-.
LDAP    anonymous    bind    worked.        . . . _                ^    _'
```

Amikor felhasználóként kell bejelentkezned, akkor a következő kódot használd. A második paraméterre relatív megkülönböztetett névként (RDN) hivatkoznak, a harmadik paraméter pedig a jelszó. Az RDN általában a **cn=username** formában tartalmazza a felhasználói nevet, és tartalmazhat további azonosítókat, például a könyvtár azon részét, amelyhez jogosult vagy hozzáférni. Az **ldap_bind()** dokumentációja azt mondja, hogy adhatsz jelszó nélküli felhasználói nevet, de amikor ezt a PHP 4.0.7-tet kipróbáltam, nem stimmelt a paraméterek száma. Az OpenLDAP-ben van egy alapértelmezett Manager nevű felhasználó, aki mindent megcsinálhat, és van egy alapértelmezett jelszó, a secret. Használd a Managert a

kezdeti adatbázis-beállításokhoz és -tesztelésekhez, de mielőtt az LDAP-adatbázist nyilvános szerverre helyeznéd, változtasd meg a felhasználói nevet és a jelszót:

```
Sldap["user"] = "Manager";
$ldap["rdn"] = "cn=" . $ldap["user"] . ", dc=test, dc=petermoulding,
    dc=com";
$ldap["password"] = "secret";
if ($ldap["connection"] ) {
    if(ldap_bind{$ldap["connection" ] , $ldap["rdn"] , $ldap["password"])) {
    print("<br>LDAP bind worked for " . $ldap["user"]);
    }
    el se                    I .  .  .  .  .  ;
        print("<br>LDAP bind failed for " . $ldap["user"]);
```

Ha a felhasználói név vagy a jelszó hibás, a következő üzenetet kapod. Az üzenet akkor jelenik meg, ha az RDN egyéb azonosítókat hiányol. Lehet, hogy csak az adatbázis egy részéhez vagy jogosult hozzáférni, és ebben az esetben az RDN részeként meg kell határoznod az adatbázis adott részét:

```
Warning: LDAP: Unable to bind to server: Invalid
credentials Figyelem: LDAP: Képtelen összekapcsolódni a
szerverrel: Érvénytelen azonosítók
```

Ha rosszul adod az RDN formátumát, a következő üzenetet kapod:

```
Warning: LDAP: Unable to bind to server: Invalid DN syntax
Figyelem: LDAP: Képtelen összekapcsolódni a szerverrel:
Érvénytelen DN
szintaktika
```

Ha az LDAP-adatbázis számtalan országra vonatkozó információkat tartalmaz, és a különböző emberek adott országok információit frissíthetik, és például te vagy jogosult Skócia adatainak karbantartására, akkor a következő kódnak megfelelően hozzá kell adnod a c = Scotland-et az RDN-hez. A kód az előző kód első része, illetve az extra paraméterként hozzáadott ország:

```
$ldap["user"]    =    "Manager";
$ldap["country"]    =    "Scotland";
!fí $ldap["rdn"]    =    "cn="    .    $ldap["user"]    dc=test,    dc=petermoulding.
. dc=com,    c="    .    $ldap["country" ] ;
```

A legtöbb LDAP-szerver úgy van beállítva, hogy az országmező az ország neve helyett a kétkarakteres országkódot tartalmazza. Az országkódokat a fejezet későbbi, „Országkódok hozzáadása" című részében magyarázom meg. Skóciának nincsen országkódja, a GB-ben van benne (Nagy-Britannia vagy Egyesült Királyság).

Ha befejezted a csatlakozást, a következőben bemutatott ldap_close()-zal zárhatod be azt. A függvénynek csupán a csatlakozási azonosítóra van szüksége. A függvény igazat ad vissza sikeres művelet, hamisat pedig hiba esetén:

```
if($ldap["connection"]) { if(ldap
    close($ldap["connection"]))
        {
        print("<br>LDAP close worked.");
        }
    else

        print("<br>LDAP    close    failed.");
```

Az eredmény:

LDAP close worked. ▮ ƒy-B .. •.: . ▮▮. .

Az ldap_close{) alternatívája az **ldap_unbind().** Mindkét függvény ugyanazt a paramétert fogadja el, ugyanazt a műveletet végzi el, és ugyanazt az eredményt adja vissza. Én az **ldap__close()-t** szoktam használni, mert könnyebb megjegyezni, hiszen sok olyan PHP-függvénycsoportnak, amelyiknek van nyitó (open) függvénye, van záró (close) függvénye is. Az **ldap_unbind()** így működik:

```
if($ldap["connection"])

    if(ldap_unbind($ldap["connection"]))

        print ("<br>LDAP    unbind    worked.");                    ___ _:.  .,

    else                    jj-jítasü öxs

        print("<br>LDAP    unbind    failed.");
```

Az összes többi Gyors megoldás ezt a csatlakozási kódot használja, és egy **$ldap["connection"]**-teszttel kezdődik. A frissítések azt feltételezik, hogy az OpenLDAP által alapértelmezésben beállított Manager-azonosítót használod, illetve azt, hogy az összes többi kódra az alapértelmezett globális olvasási jogosultságot. Ha másmilyen LDAP-szervert használsz, változtasd meg a frissítési kódot, hogy a megfelelő jogosultsága felhasználót használja.

Országkódok hozzáadása

Az LDAP-könytárak tipikusan legmagasabb szintjei a domainek és az országkódok. Ez a megoldás egy mozdulattal az összes országkódot hozzáadja, így egy teljesen nemzetközi oldalt hozhatsz létre, amelyben az országkódok listáját például űrlapok bejelölő dobozaiban használhatod. Az ehhez alkalmazott függvények az **ldap_add()** és az **ldap_mod_add().**

Jóllehet a HTML-oldalak az országkódokat nem közvetlenül az LDAP-könyvtárból olvassák be, használhatod azt egy bárhova beágyazható fájl létrehozására. Ha az oldalad vagy

•ari ,}3fjvrjm í.DVJ

vállalkozásod egy LDAP-könyvtár köré épül, egyszerű az országkódokat az LDAP-könyvtárban naprakészen tartani, a könyvtárat pedig az oldalad forrásaként használni.

Csatlakozás

Ez a kód feltételezi, hogy az előző, „Csatlakozás az LDAP-hez" című megoldásban mutatott kóddal csatlakoztál az LDAP-hez, a Manager felhasználói nevet használva. A következő kód a **ldap_bind()** és a **ldap_close()** közé van beszúrva. A megoldás többi része az if() kapcsos zárójelei közé van beírva, így a kód csak akkor futtatható, ha van érvényes csatlakozás:

```
if($ldap["connection"])
    {
    //      The rest of the
    }       code
```

Országkódok megszerzése

Az országkódok listáját a könyv CD-ROM mellékletéből szerezheted meg (/examples/countrycodes.txt), és tedd a fájlt olyan helyre, ahol PHP-ből beolvashatod. Ha naprakész listát vagy még több információt szeretnél az országkódokról, látogasd meg a www.din.de/gremien/nas/nabd/iso3166ma/-oldalt, és keresd meg az ISO 3166 listát.

A következő példa a 239 elemű lista első néhány sorát mutatja az adott fájlból. Minden sornak szóköz és **newline** van a végén, így a **trimQ**-függvénnyel meg kell őket csonkolni. A nevet és a kétbetűs országkódot az explode()-dal tudod szétválasztani, illetve tömbbe helyezni a későbbi formázáshoz és használathoz. A kétbetűs országkódok nagybetűvel vannak írva, hogy megkülönböztethessük őket a kétbetűs nyelvi kódoktól. A nyelvi kódok kisbetűsek, és a kettőt néha kombinálni kell, ha a böngésző egy weboldalt meghatározott nyelven kér le. Egy amerikai az angol nyelvű oldalakat en-US-ként kéri, jelezve, hogy a *catalog* szó az amerikai írásmód szerint legyen *catalog*, míg egy brit inkább a en-GB-t kéri, hogy a *catalog catalogue-ként* legyen írva:

```
AFGHANISTAN;AF
ALBÁNIA;AL
ALGÉRIA;DZ
AMERICAN SAM0A;AS
```

Országkódok formázása

Az első lépés az input-fájl megtisztítása, illetve az eredmények tömbbe helyezése, hogy azok bárhol, így adatbázisok és HTML-készítő programok inputjaiként is használhatók legyenek. Helyezd át a countrycodes.txt-t ugyanabba a könyvtárba, ahol a szkripted is van, és tedd a fájl nevét a **$file-be,** mint itt, vagy mutasson a **$file** arra a helyre, amit a fájlnak választottál:

```
$file    =    "./countrycodes.txt";
```

A következő kódban a **file()** beolvassa a szöveges fájlt egy tömbbe úgy, hogy a tömb minden eleme egy sornyi inputot tartalmaz. (A fájlfüggvényeket a 8. fejezetben magyarázom el.) A while()-ciklussal végigfut a **$text** elemein. A kód többi része beolvas egy elemet, formázza azt, majd a formázott tartalmat új elemként berakja a **$codes** nevű tömbbe. Az összes elem az **explode()-dal** van két részre bontva: az ország neve a $name-be, az országkód pedig a $code-ba kerül. A **$name** nagybetűs, így kisbetűkre konvertálom. Valójában az ucwordsQ az összes szó első betűjét nagybetűsre állítja, de ez a függvény csak üres közökkel határolt szavakkal működik, és ebben a fájlban a nyitó zárójel és a pont után nagybetű kell. Az első két str_replace()-függvény a (és . jeleket üres közökre cseréli, a következő két **str_replace()**-függvény pedig az eredeti értékre konvertálja vissza az üres közöket. A maradék **str_replace()**-függvények manuális igazítást végeznek az egyedi és utolsó szóként of-ot tartalmazó neveken. Az eredményt a vizuális ellenőrzés kedvéért megjelenítem, és a **$codes-ba** tárolom:

```
if(Stext   =   file(Sfile))

   print("<br>File worked for " . Sfile);
   while (üst (Sk, Sv) = each (Stext) )

      $v = trim($v);
                                    ", $v);
            strtolower(Sname);
            str_replace ("(", "\n",  Sname);
            str_replace(".", "\r",  Sname);
            ucwords(Sname);
            str_replace("\r", ".",  Sname);
            str_replace("\n", "(",  Sname);
            str_replace(" And ", "  and ", Sname);
            str_replace("D'ivoire",  "d'Ivoire", Sname)
            Sname . " " ; str__replace
            (" Of str replac
            trim(Sname); . S                        Sname);

                              "Snbsp; Snbsp;" . Sname);
      list($name, $code) =
      explode
      Sname = Sname = Sname =
      Sname = Sname = Sname =
      Sname = Sname = Sname =
      Sname = Sname = Sname =
      print{" Scodes[Scode] =
      Sname;


else í
   print("<br>File failed for " .     $file)
                               The
```

Hivatkozás:	oldal:
Bármilyen típusú adat megjelenítése	**278**

Az első országkód hozzáadása

Ez a kód a **$codes** első elemét olvassa be, és az **ldap_add()** használatával hozzáadja az

LDAP-könyvtárhoz. Az első sor a $codes-tömb elemmutatóját a tömb elejére állítja, amit

érdemes megtenni, amikor van kód a tömb létrehozása és használata között (a tömbökről részletesen a 3. fejezetben esik szó):

```
reset (Scodes) ;                    -,    .                    ;   . if(  i-- ; j ^*. ; i
```

Az LDAP-könyvtár megfelelő helyének eléréséhez szükséged van egy megkülönböztetett névre, mely pontosan oda mutat, ahova az új elemet be akarod szúrni. Mivel az ország a könyvtár legfelső szintjén van, csak a domainnévre van szükség, ahogy itt is láthatod:

```
$ldap["dn"]  =  "dc=test,  dc=petermoulding,  dc=com";
```

A **ldap_add()** hozzáadja az első elemet az elemek egy osztályára, illetve megadja az elem objectClass-definícióját. Mivel az elemet egy tömbből adod hozzá, kezdd a tömböt az objectClass-definíciónak - ami jelen esetben a **country** - az elem tömbjéhez való hozzáadásával:

```
$entry["objectClass"] = "country";
```

Ezután szükséged van az elem tömbjében az országkódra, és ezt a kód következő sora csinálja meg, ahol az országkód a Sentry["c"]-be kerül. Vedd észre, hogy könyvtárba történő beszúráskor az elem kulcsa a mező neve. Az OpenLDAP által biztosított alapértelmezett LDAP-könyvtársémában nincs kiosztva attribútum az országnévhez, így ez kimaradt ebből a példából: `list($entry["c"],  $v)  =  each(Scodes);`

Add hozzá az elemet az objectClass-szal együtt a **ldap_add()** használatával, ahogy a következő kód mutatja. Az első paraméter az LDAP csatlakozási azonosító, a második a megkülönböztetett név, ahova az elemet hozzáadod, a harmadik pedig az elemet tartalmazó tömb. Az **ldap_add()** igazat ad vissza sikeres művelet, hamisat hiba esetén:

```
if(ldap_add($ldap["connection"],  $ldap["dn"],  Sentry))                    w

    print("<br>LDAP  add  worked
    for                                  $entry["c"]);

else

    print("<br>LDAP  add  failed          $entry["c"] ) ;                    nöJ iM
    for
```

További országkódok hozzáadása * Ja';^?

Amikor már van egy országkód a könyvtárban, nincs többé szükség az objektumosztályra, így a következő kód használatával azt távolítsd el az elem tömbjéből:

```
unset($entry["objectClass"]);
```

A következő kód ciklussal végigfut a Scodes-tömbön, és a **ldap_mod_add()** használatával az egyes elemeket hozzáadja az LDAP-könyvtárhoz. Mivel az *előző* kód az **each()** használatával érte el a tömb elemeit, a tömbmutató a második elemen áll, a ciklus pedig a máso-

diktól az utolsó elemig fog futni. Az ldap_mod_add() paraméterei megegyeznek a ldap_add()-éivel:

```
while (üst ($entry ["c"] ,     $v)     =     each ($codes) )
        {                                                    *: & ü
        if(ldapmodadd(Sldap["connection"] , $ldap["dn"], $entry))
            {
            print("<br>LDAP add worked for " . $entry["c"] );
            }
        else
            {
            print("<br>LDAP    add    failed    for    "     .     $entry["c"]     );
```

Ha a country-t más, például capital (főváros) vagy population (népesség) attribútumokkal definiálták volna, az ldap_add() vagy az ldap_mod_add() használata előtt ezeket az attribútumokat adnád az elem tömbjéhez, ahogy azt a következő kód mutatja. A példában Kína fővárosa és népessége szerepel:

```
Sentry["capital"]     =     "Beijing";
$entry["population"]     =
"1300000000";
```

Az LDAP többszörös értékeket is engedélyez az egyes elemek attribútumaihoz, kivéve, ha egy attribútumban a többszörös értékek egy sémán belül ki vannak kapcsolva. Hogyan adhatsz egy elem tömbjében többszörös értéket egy attribútumnak? Használd egyszerűen az itt látható hagyományos PHP-tömbökhöz való elemhozzáadási technikát. A példa azt feltételezi, hogy az airport-ot a country attribútumaként adjuk hozzá, és az airport-ban a többszörös értékek nincsenek kikapcsolva. Az itt felsorolt repülőterek Kína nemzetközi repülőterei:

```
$entry["airport"][]   =   "Shanghai    Pudong";
$entry [ "airport" ] [ ]   =   "Guangzhou    Baiyun";
$entry ["airport"] []   =   "Xinzheng";
$entry["airport"][]   =   "Hong    Kong";
```

Mi történik, ha elszúrod az elemtömböt? A következő üzenetet kapod. Ezt az általános üzenetet tucatnyi hiba esetén megkaphatod, például az attribútumnevek félregépelésekor vagy az ldap_mod_add() helyett használt ldap_add() esetében:

```
Warning: LDAP: add operation could not be completed.
Figyelem: LDAP hozzáadás művelete nem hajtható végre.
```

Felhasználó hozzáadása

Amikor az alapértelmezett LDAP-könyvtár konfigurációhoz új felhasználót adsz hozzá, akkor először az országot, majd a szervezetet adod hozzá, és csak a végén a személyt. Az előző megoldásban az összes országkódot hozzáadtam, így ez a példa egy szervezetet és egy személyt ad hozzá. Egy LDAP-könyvtárt bárhogyan konfiguráltatsz a séma megváltoz-

tatásával, de tekintve a LDAP-séma kialakítások korlátozott dokumentációit, javaslom, hogy a saját tervezés előtt gyakorold a PHP LDAP-függvények használatát az alapértelmezett sémán.

Ez a megoldás a folyamat bemutatására egy közbülső szintet (organization - szervezet) és egy végső szintet (person - személy) ad hozzá. Akárhány közbülső szintet hozzáadhatsz, és egy kicsiny kóddal az input-fájlon úgy is végigfuthatsz, hogy többszörös elemeket adjon

-,>■■>-ír.

Közbülső szintek hozzáadása

Egy elem hozzáadásakor szükséged van egy kiindulópontra, egy megkülönböztetett névre, amely arra az elemre mutat, amely az új elem szülője lesz. Ez a lépés a country-hoz adja az organization-t, így a DN a megfelelő országra mutat, ahogy itt láthatod:

```
$ldap["dn"] = "c=AU, dc=test, dc=petermoulding, dc=com";
```

Az elemet egy tömbön keresztül adom hozzá, amelyet az ldap_add()-be betáplálok. A tömb neve Sentry, és a következő példában látható. A $entry-tömb objectClass-eleme az organization-t nevezi az elemhez használt objektumosztálynak. Az organízation-objek-tumosztálynak egy kötelező attribútuma van, az o, és van néhány tetszőleges. Most csak az o-attribútumra van szükség. A cégemet neveztem el szervezetként, mert <reklám> egy nagyszerű weboldalhoz csak erre az egy szervezetre van szükséged </reklám>:

```
$entry["objectClass"] =
"organization"; $entry["o"] =
"petermoulding";
```

Az ldap_add() a csatlakozási azonosító a DN és a tömb használatával hozzáadja az elemet:

```
if(ldap_add($ldap["connection"] , $ldap["dn"], $entry))
    {
    print("<br>LDAP add worked for " . $LDAP["dn"]);
    }
else

    print("<br>LDAP add failed for       $LDAP["dn"])
                                        ;
```

Az eredmény a következő:

```
LDAP add worked for c=AU, dc=test, dc=petermoulding, dc=com
```

A végső szint hozzáadása

Ez a lépés a személy elem végső szintjét, a person objektumosztályú szintet, illetve a person-objektumban definiált attribútumokat adja hozzá. Az attribútumok a keresztnév (cn) és a vezetéknév (sn). Amikor egy elemet hozzáadsz, az összes kötelező attribútumot is hozzá kell adnod (a sémában a MUST alatt felsorolt attribútumok), viszont kihagyhatod az opcionális, a sémában a MAY alatt felsorolt attribútumokat: ;„

```
$ldap["dn"] = "o=petermoulding, c=AU, dc=test, dc=petermoulding,
    dc=com";
unset ($entry) ;      ■ ' '."'                                          >'
$entry["cn"] = "Pixie";      (: ■ ""                              "   nferrre«
$entry["sn"] = "Moulding";   " "                 ..' ;',,, ., \
$entry["objectClass"] = "person";
if($ldap["connection"])

    if(ldap_add($ldap["connection"] , $ldap["dn"], $entry))

        print("<br>LDAP add worked for " . $idap["dn"]) ;

    else

        print("<br>LDAP    add    failed    for    "    .    $ldap["dn"]);

    }
```

Az eredmény:

```
LDAP    add    worked    for    o=petermoulding,    c=AU,    dc=test,
     dc=petermoulding, dc=com
```

Hibakezelés

Olyan kódot akarsz, amely kezeli a hibákat, így a weblapod látogatóinak nem kell a szépen megtervezett oldalakon hibaüzeneteket olvasniuk. Három függvény segít neked ebben: **ldap_errno(), Idap_err2str()** és **ldap_error().** A @-t is használnod kell.

A @ az a PHP-függvény előtag, amely elrejti a beépített hibaüzeneteket. A példa egyik lépésében az ldap_bind()-ot hiba létrehozására használjuk, így láthatod azt a hibaüzenetet, ami el lesz rejtve, majd láthatod a hibakezelést.

A következő kód úgy csatlakozik az LDAP-hez, ahogy azt a „Csatlakozás az LDAP-hez" című Gyors megoldásban mutattam. Mivel a LDAP-hibafüggvényekhez előtte csatlakozni kell az LDAP-hez, nem használhatod ezeket az LDAP csatlakozási hibáinak a kijavítására:

```
$ldap["site"]    =    "test.petermoulding.com";
if($ldap["connection"]    =    ldap_connect($ldap["site"]))

    print ("<br>LDAP connected to       Sldap [ "site" ] ) ;
    }
else

    print("<br>LDAP    connection    failed    for    "    .    $ldap["site"]);
```

A következő sor a szokásos ldap_bind()-kód az LDAP elérésére, egy apró változtatással. A második paraméter, a megkülönböztetett név érvénytelen információt tartalmaz, amely nem alkot megkülönböztetett nevet:

```
if($ldap["connection"])                                          ,
    if(ldapjoind($ldap["connection"],    "aa",    "bb"))                  s-
        print("<br>LDAP   bind   worked.");
                                                                  »▮▮  ..-in-
    else
        print ("<br>LDAP   bind   failed.");                    .   ""'"
    }
```

A hiba a következő üzenetet generálja, ami egyáltalán nem segít a látogatóidnak:

```
Warning: LDAP: Unable to bind to server: Invalid DN syntax
Figyelem: LDAP: Képtelen összekapcsolódni a szerverrel:
Érvénytelen DN-szintaktika.
```

A következő kódban az **Idap_bind()**-függvény elé @-t tettem, hogy a beépített hibaüzeneteket elrejtse. Próbáld ki az *előző* kódot a @ itt látható beszúrásával:

```
if(@ldapjoind($ldap["connection" ],    "aa",    "bb"))
```

Amint a beépített hibaüzeneteket elrejted, észlelheted a szkripteddel a hibákat, és sokkal kevésbé zavaró módon kezelheted. A fontos hibákat naplózhatod, így később áttekintheted és megelőzheted őket.

Az **ldap_errno()** csatlakozási azonosítót fogad el, és a csatlakozás alatt végzett utolsó művelet hibaszámát adja vissza. A hibaszám alapján elvégezhetsz műveleteket, vagy a számot megjeleníthető üzenetre konvertálhatod a **Idap_err2str()** használatával. A következő kód visszakeresi a hiba számát az **ldap_errno()** segítségével, és a switch()-csel a megfelelő műveletet választja ki:

```
$error = ldap_errno($ldap["connection" ]);
switch($error)

    case 0 :
       print("<br>No error.");
       break;
    case 34 :
       print("<br>There is a syntax error in the Distinguished Name.");
       break;
    default:
       print ("<br>Error number: " . $error);                    ii^Ji™_»«—,
    }                                                             "    *
```

Honnan tudtam, hogy a szintaktikai hiba száma a 34? Az Idap_err2str()-rel kilistáztam az összes előforduló hibát. A hibaszámok az LDAP-szoftverek és -könyvtárak teljes körében szabványosak, de az LDAP összes implementációjának megvan a saját hibaüzenet-készlete. Ha azt akarod eldönteni, hogy a szkript milyen műveletet hajtson végre, a döntést a hibaszám és ne a hibaüzenet szövege alapján hozd meg. A művelet lehet egy input-űrlap visszaküldése a felhasználónak, hogy újra kitöltse azt, vagy egy e-mail küldése saját magadnak, hogy figyelmeztessen egy súlyosabb hiba esetén.

Az Idap_err2str() egy LDAP-hibaszámot fogad el, és a hibaüzenetet adja vissza. A következő kód az első 100 lehetséges hibaüzenetet adja vissza, így megnézheted, hogy az egyes üzenetek mit jelentenek (és felkészülhetsz a kezelésükre is):

```
for($i = 0; $i < 100; $i++)                A^í Z?ú>") .vnxrq ■

   print("<br>Error number:    $i   ,     Idap_err2str
                                  message:
```

A következő lista a hibaüzenetek listáján első része, így elkezdheted a tervezést. A 3. és 4. üzenetben levő időbeli és méretbeli korlátok nagy könyvtárakban történő átfogó keresés esetén fordulhatnak elő:

```
Error number: 0, message: Success
Error number: 1, message: Operations error
Error number: 2, message: Protocol error Time
Error number: 3, message: limit exceeded Size
Error number: 4, message: limit exceeded
Error number: 5, message: Compare falsé Compare
Error number: 6, message: true
```

Amikor ez a kód elég nagy számot ér el, olyan üzeneteket fogsz látni, mint a következő kettő. Ez azt jelzi, hogy túlfutottál a lehetséges hibaüzeneteken. Vannak hézagok a lista közepén, így az LDAP minden egyes új változatának megjelenésekor a listát az utolsó ismert hibaüzenetet jócskán meghaladó számig kell kiíratni, hogy meggyőződj arról, hogy nincsenek új üzenetek megbújva a hézagokban:

```
Error number: 99, message: Unknown
error Error number: 100, message:
Unknown error
```

Az **ldap_error()** csatlakozási azonosítót fogad el, és a linken keresztül elvégzett utolsó művelet hibaüzenetét adja vissza, ami megegyezik az **Idap_err2str(ldap_errno())** használatával- A következő kód az **ldap_bind()-hiba** miatti üzenetet jeleníti meg:

```
print ("<br>Error: " . ldap_error($ldap["connection"] ) ) ;
```

Az eredmény a következő. Ne felejtsd el, hogy ez az üzenet az OpenLDAP-tól jött, és nem biztos, hogy ugyanezt a szöveget kapod a Netscape-könyvtár termékétől vagy más LDAP-szerverektől:

```
Error: Invalid DN syntax *
```

Az összes elem listázása

Előfordulhat, hogy egy tesztkönyvtár összes elemét és attribútumát ki akarod listázni, például amikor az első elemeket adod hozzá egy teljesen új könyvtárhoz. Megeshet, hogy a segélyhívó információkat akarod az **emergency objectClass** használatával az egyes helyekhez hozzáadni, és meg akarsz győződni arról, hogy az elemek 100 százalékig helyesek. íme az ehhez való megoldás.

A kódnak három változata van. Az első változat az **ldap_read()** és az **ldap_get_entries()** használatával ragadja meg az elemeket, hogy bármilyen általad választott megkülönbözte tett névre az elemek egy szintjét adja eredményül. Az adatok egy táblázatban vannak kilis tázva, így láthatod a formátumot és megértheted a következő részeket. A második változat minden szint minden elemét beolvassa (annyi song, *amennyit* meg akarsz jeleníteni), és az adatokat kissé strukturálva jeleníti meg, így láthatod, hogyan kezelhetők a többszörös szin tek. A harmadik változat egy kis csinosítást kísérel meg az outputon, hogy jobban olvasha tó legyen. . ., .j ,

Ha tovább akarod vinni a formázást, akkor a Windows Intézőnek vagy más kifinomult strukturált adatbeolvasónak a megfelelőjét alakíthatod ki. Amikor a megjelenítést ilyen szintre viszed, a finom részletek attól függnek, hogy hogyan strukturálod a könyvtárat, milyen adatokat használsz, és mit akarsz megmutatni a böngészőnek. Akarod, hogy bizonyos emberek gyorsan eljussanak az e-mailcímekhez? Akarod, hogy a megjelenítés megőrizze, hogy hol volt az illető legutoljára, amikor a könyvtárat böngészte? Hány opciót akarsz nekik adni **a** kereséshez? Függetlenül attól, hogy milyen bonyolultra csinálod **a** könyvtárstruktúrát, a tartalmat és a felületet, amikor a hibákat próbálod megjavítani, mindig szükséged lesz a mindent megjelenít opcióra.

Ennek a megoldásnak a kódja „Csatlakozás az LDAP-hez" című megoldás csatlakozását és összeköttetését használja. A kód **a** csatlakozási azonosító tesztelésével kezdődik.

Az egy szinten levő összes elem listázása

A legmagasabb szintű elem az OpenLDAP-könyvtár alapértelmezett beállításában a **country,** így az első kód az országokat *listázza*, ki. A country-elemnek egy kötelező attribútuma van, az országnév, amelyet c-nek rövidítenek. Az összes nyilvános LDAP-könyvtárban az ISO-szabványos kétbetűs országkódnak kell lennie az országnévben.

Az LDAP olyan sorrendben tárolja az elemeket, amilyen sorrendben hozzáadtad őket, és **a** normál visszakereső függvényekben nincsen rendezési lehetőség. Ha az értékeket sorrendben akarod, **a** legegyszerűbb megoldás őket PHP-tömbbe rakni, és a tömböt rendezni. Ha a visszakeresés után mindig rendezni akarod az elemeket, gondold végig, nem célszerűbb-e az adatokat hagyományos relációs adatbázisban tárolni, és SQL-nézeteket használni arra, hogy az adatokat az általad kívánt módban adja vissza.

Adatok olvasása

A következő kód visszakeresi az adatokat, az **ldap_read()** használatával kiválasztja egy könyvtár összes elemét, az **ldap_get_entries()** pedig kinyeri az elemeket egy **$entries** nevű tömbbe. A kód többi része státusüzenetekek jelenít meg, hogy lásd, mit érnek el a függvények. A kezdő pont a könyvtárban a **$ldap["dn"]-ben** megadott megkülönböztetett név. Ha az **ldap_read()-nek** sikerül eredményt létrehoznia, az **ldap_free_result()** lefut, hogy felszabadítsa az eredménykészletet arra az esetre, ha a szkripted a számtalan eredménykészlet létrehozásával felzabálná a memóriát:

```
$ldap["dn"] = "dc=test, dc=petermoulding, dc=com";           ''
if($ldap["connection"]) {
    if (Sresult = @ldap__read($ldap["connection"], $ldap["dn"],
        "objectClass=*"}) {
        if(Sentries = ldap_get_entries($ldap["connection" ], $result)) {
           print ("<br>ldap_get_entries returned " . count($entries)
               . " for " . $ldap["dn"]) ;              ,  kni ■;

        else
            {
        print ("<br>ldap_get_entries failed for " . $ldap["dn"]); }
        ldap_free_result($result);
        }
   else { print("<br>LDAP read failed for " . $ldap["dn"]);
```

Adatok megjelenítése

A Sentries-tömböt az előző kód hozza létre, a következő kód pedig ennek teljes tartalmát megjeleníti egy a lényeggel törődő HTML-táblázatban. A tömb elemek keverékét tartalmazza: van, ami maga is tömb, és van, ami nem. A tömbök három szint mélységig mehetnek, így hasonló kód van három szint mélységig beágyazva. A **while()**-utasítás egy szinten fut végig a tömbön. Az if()-utasítások észlelik a mezőket, és outputot csinálnak belőlük, vagy ha tömböt találnak, a következő szintű whileQ-utasításhoz továbbítják őket:

```
if (isset (Sentries)) {
    print ("<table    border=\"3\">\n"     ) ;
    while (üst ($kl,     $vl)     =    each ($entries )
    ) { if(is_array($vl))
            {   —
           while (üst ($k2, $v2) = each{$vl))

               if(is_array($v2))
                    (                          ' '  '.-:,,.
                  while (üst ($k3, $v3) = each($v2)) {
                     print ("<tr><td>" . $kl . "</tdxtd>" . $k2 . "</td>" .
                         "<td>" . $k3 . "</td><td>" . $v3 . " . "</tr>\n");


               else

                   print ("<trxtd>" . $kl . "</td><td>" . $k2 . "</td>"     ■
```

```
                                    "    .    $v2    .    "</tdxtd> </td></tr>\n")


        else
            {
            print ("<tr><td>"    .    $kl    .    "</tdxtd>"    .    $vl
                .    " .    "<td>  </tdxtd> 

    print ("</table>\n");
```

A 14.4 ábra mutatja az eredményeket. Mivel 239 ország van a könyvtárban, kivágtam a középső 233-at az ábrából. A tömb legmagasabb szintje egyetlen elemet tartalmaz, a **count**-ot, amely a következő szinten található tömbök számát mutatja. Több tesztet is lefuttattam, és mind ugyanazt a legfelső szintet adták eredményül.

count	1	**i**	
0	objectclass	**count**	**l**
1	objectclass	**0**	**country**
0	0	**objectclass**	
0	**c**	**count**	**238**
1	**c**	**0**	
1	**c**	**1**	**AL**
0.	**c**	**2**	
0'	c	**235**	
1	**c**	**236**	
1	**c**	**237**	
0	1	**c**	
0	**count**	2	
0	**dn**	dc=test, dc=petermoulding, dc=com	

14.4 ábra Nyers adatok a ldap_get_entries()-ből

A 0 elem - a legfelső szinten - egy tömb, amely elemek összevisszaságának tűnő valamit tartalmaz. A második szint helyes olvasásához kezdd a dn nevű elemmel. A dn megkülönböztetett nevet tartalmaz, ami a tömbön belül az elemekre mutat. A count az azon a szinten levő tömbök számát tartalmazza, így a megszámolt elemeken egy for()-ciklussal végigfuthatsz. A 0-tól l-ig számozott elemek a következő beolvasandó elemek.

A 0 elem tartalmazza az objectclass nevet. Az **objectclass** ezen a szinten egy elem és egy tömb, amely az egy szinttel lejjebb olvasandó elemeket tartalmazza. Az 1 elem tartalmazza a **c** nevet, amely egy **c** nevű elemre mutat, ez egy szinttel lejjebb feldolgozandó tömb. A **country** az elem objectclass-a, míg a c és az objectclass a country attribútumai.

A c egy tömb, amely tartalmazza a c-tömbben lévő, megszámolt elemek számát, és a megszámolt elemeket. A **c** a többértékes attribútumok példája.

Az összes szint összes elemének listázása

Egy LDAP-könyvtár összes szintjének végigolvasásához a következő kód egy függvényt használ, amely feldolgoz egy szintet, majd a függvény behívja saját magát, hogy az alacsonyabb szinteket is feldolgozza. A függvény az adatokat megjelenítendő sorokként rakja be egy tömbbe, és egy megjelenítendő sorba beleveszi az alsóbb szintek összes sorát is. Minden szint különböző hosszúságú nemüres szóközöket -   - tesz a megjelenítendő elé, ezt a böngésző észreveszi, és behúzással jeleníti meg a sorokat.

Ha ezt a kódot a tesztkönyvtáron kívül bármi máson kipróbálod, győződj meg arról, hogy beleveszed a nyomtatásra besorolt sorok megszámlálását, illetve a visszakeresést azelőtt befejezed, hogy elérnéd a böngésződ tulajdonképpeni határait.

A függvény

A display_one_entry() az ldap_read()-et használja, hogy az egy szinten levő összes elemet beolvassa. Az ldap_read()-nek egy csatlakozási azonosítóra, a kiindulási ponthoz egy megkülönböztetett névre és egy keresési sztringre van szüksége. Az objectClass = *-sztring minden objectClass attribútumú elemet megkeres. Az elemeknek kell, hogy legyen egy objectClass-uk, mielőtt egy LDAP-könyvtárba kerülnek, így a keresési sztring minden elemet meg fog találni.

Az ldap_read() az Ldap_get_entries()-be táplált eredményazonosítót adja vissza. Az ldap_get_entries() ezután egy tömböt ad vissza, amelyet megjelenítendő sorokra dolgoz fel egy kóddal, amely különbözik a korábbi megoldásban használt kódtól. Ez a változat azon információk alapján dekódolja a tömböt, amit a tömb korábbi megjelenítésekor fedezett fel.

A tömb minden szintjének van egy számértéke, ezt a számértéket teszi a $cl-, $c2- stb. mezőkbe, és ezeket használja a for()-ciklus korlátjaként. A for()-ciklus végigfut a tömbökön és az altömbökön, és beolvassa azokat a számozott tömbelemeket, amelyek LDAP-elemeket (1. szint) jelképeznek, az elemek attribútumait (2. szint) és az attribútumok több értékét (3. szint).

A $s a sorok szabványos behúzását adja meg, a $sl, a $s2 stb. pedig a kumulált behúzásokat alkalmazzák a sorokra. A sorok végül a Sprint-tömbbe kerülnek, amelyet kiírat a kód. Az output minden sora egy leírást tartalmaz, amelyet az érték követ:

```
function display_one_entry($connection, $dn)

    if($result = @ldap_read($connection, $dn, "objectClass = *" ) )

        $entries = ldap_get_entries($connection,   $result
        $cl = Sentries ["count"]; $s =             )
        "Snbsp;Snbsp; "; for($il = 0; $il <
        $cl; $il++)

            $v2   =   $entries
            $print[]   =   "dn:   "   .
            $v2["dn"];
            $c2   =   $v2["count"];
```

```
        for($i2 = 0; $i2 < $c2;

            $a2 = $v2 [$Í2] ;
            $print[] = $s . "attribute: " . $a2; "v"
            $v3 = $v2[$a2];
            $c3 = $v3 ["count"] ;
            for($i3 = 0; $13 < $c3;

                $a3 = $v3[$i3];
                $print[] = $s . $s . "value: " . $a3; if($pr
                = display_one_entry($connection, $a2 . " =
                " . $a3 . ", " . $dn) ) {
                   while {üst ($k, $v) = each($pr)) {
                       $print[] = $s . $s . $s . $v );


                }


        ldap_free_result($result);
        }

    else
        {
        $print = falsé;
        }
    return($print);
```

Adatok visszakeresése és megjelenítése

A következő kód egyszerűen azt ellenőrzi, hogy az LDAP-hez való csatlakozás létezik-e, majd a megfelelő megkülönböztetett névvel behívja a display_one_entry()-t. A display_one_entry()-ből visszaadott $print-tömb elemenként kiíratódik, a sor elején HTML-töréssel (<br>), a végén pedig newline-nal (\n). Megjelenítéskor a törés új sort tetet a böngészővel az oldalra, a newline pedig új sort jelenít meg, ha az oldal forrását nézed meg a böngészőben. A forrás megtekintésével kijavíthatod a weboldalak hibáit, a newline-ok pedig megkímélnek az időigényes vízszintes görgetéstől:

```
$ldap["dn"] = "dc=test, dc=petermoulding,
dc=com"; if($ldap["connection"] ) {
    if($print = display_one_entry($ldap["connection"], $ldap["dn"])) {
        while {üst ($k, $v) = each ($print) )
            {
            print("<br>" . $v . "\n");


    else
```

```
print("<br>LDAP search failed for $ldap["search"]);
```

A következő eredmény mutatja, hogy a kód hogyan lépked végig az elemeken és szinteken. Minden egyes alkalommal, amikor a kód egy értéket talál egy attribútumhoz, a kód megpróbálja azt alsóbb szintű elemként beolvasni. Mivel több száz ország van a könyvtárban, az outputot néhány ország után levágtam. Vedd észre az országok sorrendjét; ha ettől eltérő sorrendet akarsz, a $entries-tömböt kell újrastrukturálnod, hogy rendezhesd az elemeket:

```
        dn: dc=petermoulding, dc=com
     dc=test, :   objectclass
    attribute  country :
       value: c
    attribute  AU
       value:  c=AU, dc=test, dc=petermoulding, dc=com
          dn: attribute: objectclass
                 value: organization
              attribute: o
                 value: petermoulding
                   dn: o=petermoulding, c=AU, dc=test, dc=petermoulding,
                       dc=com
                       attribute: cn
                          value: Péter
                attribute:  sn
                   value:  Moulding
                attribute:  objectclass
                   value:  person
value  AL
value  DZ
value  AS
value  AD
```

f

Az elemek értelmezése a listázáson belül

A következő kód továbblép az LDAP-könyvtárból visszakeresett adatok értelmezése felé. Az előző display_one_entry()-függvényt úgy módosítottam, hogy értelmezze az LDAP-könyvtárban minden elemre közös mezőket és értékeket. A kódhoz bármilyen, a könyvtáradra jellemző érték értelmezését hozzáadhatod.

Az első változtatás az objectclass nevek $o-ba való kigyűjtése, majd a kigyűjtött neveknek a megkülönböztetett névhez való hozzáadása. Ezzel elemenként kétsornyi outputot lehet megtakarítani, és könnyebb megnézni, hova vezet a megkülönböztetett név. Az gondoltam, hogy egy kék fonttag megkönnyíti annak eldöntését, hogy a képernyőn melyik mező melyik.

Mivel az objectclass a ciklusnak a számozott tömbelemeken való végigfutása előtt ki van emelve, a kód tartalmaz egy if($a2 != "objectclass")-utasítást, hogy a számozott elemek között átugorja az objectclass-t. Ezt a változtatást feltételessé teheted, így teszt üzemmód-

ban futtathatod úgy a kódot, hogy a tesztelést segítendő minden mező és számérték megjelenik, de például kisebb betűvel és másmilyen színben. A cél az ismétlődő elemek által felhasznált terület csökkentése, hogy a változó értékre koncentrálhass.

Kevesebb print-utasítás van, így a megmaradtak behúzását csökkentettem. Amikor egy mély könyvtár alsó részébe jutsz, lesz olyan információ, amely a jelen kódban használt egyszerű behúzásos rendszer mellett túlcsordul a képernyő jobb oldalán. Ekkor mérlegeld a felsőbb szintek elrejtését, de mindig jelezd a megkülönböztetett névvel, hogy melyik szinten van a felhasználó:

```
function display one_entry($connection, $dn) { if($result =
   @ldap_read($connectionf $dn, "objectClass=*"))
       {
       Sentries = ldap_get_entries($connection, $result);
       $cl = $entries["count"];
       $s = "&amp;nbsp;&amp;nbsp;Snbsp;";
       for($il = 0; $11 < $cl; $il++)

          $v2 = $entries
          $o = $v2["objectclass"][0];
          $print[] = "<font color=\"Blue\"                    "</font>: "
              . $o;                                  $v2["dn"]
          $c2 = $v2["count"];
          for($i2 = 0; $i2 < $c2;

             $a2 = $v2[$i2];
             if($a2 != "objectclass")
                í
             $v3 = $v2[$a2];
             $c3 = $v3["count"];
             for($i3 = 0; $13 < $c3;
                {
                $a3 = $v3[$i3];
                $print[] = $s . $a2          $a3;
                if ($pr = display_one_entry($connection,
                   $a2 . " = " . $a3 . ", " . $dn) )

                   while(list($k, $v) = each($pr))

                      $print[] = $s . $s . $v );




       ldap_free_result($result)
       }
   else  {  $print  =
       falsé;
```

```
return($print);
```

Az eredmény a következő. A használhatóság csökkentése nélkül a 22 sorból 12 lett:

```
dc=test, dc=petermoulding, dc=com:
    country c: AU
        c=AU, dc=test, dc=petermouldíng, dc=com:
            organization o: petermoulding
                o=petermoulding, c=AU, dc=test, dc=petermoulding,
                    dc=com: person cn: Péter sn: Moulding
    c: AL                                                    í<i- ■ *  3 l i > ; i-
    c: DZ                                                                  i
    c: AS                                                      *    ES«.í."í:Í?
    c: AD                          K                          -T.í-TS ■■   '  : a£
```

Ha úgy döntesz, hogy LDAP-t használsz, akkor a következő kihívás a 9. fejezetben leírt űrlapok használata, illetve a legfelső szintű megkülönböztetett nevek begépelésének lehetősége. Ehhez a rövid attribútumneveket egy kereső tömb használatával hosszabbra kell változtatnod, például sn-ről surname-re. Az alsóbb szintű mezőket linkre változtathatod, így csak egy kattintás egy alsóbb szintre, és az új megjelenítés az adott szinttől kezdődik. Az eredmény a könyvtár egy térképe, amellyel az egész könyvtárban mozoghatsz. Ne felejts el a felsőbb szintekre mutató linkeket is beszúrni, és legyen egy olyan inputmező, ahol a felhasználó meghatározhatja, hogy hány szintet akar egy oldalon megjeleníteni.

Hivatkozás:	oldal:
Űrlap létrehozása	307

:ir- '- (■, fu;3S
■ f

uhi! i
így :.

15. fejezet

Posta

Áttekintés

Ha valaha megőrültél már az e-mailkliensekről, itt az idő, hogy bosszút állj, és azokat a vacak alkalmazásokat a te penge, kifinomult PHP alapú kliensedre cseréld. Olvasd el a „Működési üzemmódok" című részt, utána pedig írd meg a saját kliensed, ami megfelel az igényeidnek.

Ha levelet akarsz küldeni, a PHP mail()-függvényei kínálják a legjobb megoldást. A fejezet Gyors megoldásai megmutatják, hogyan küldj egy vagy több levelet. Ha a bejövő levelek vannak a középpontban, a PHP LDAP-támogatása jelenti a legjobb választást, de ekkor is olvasd végig a mail()-rüggvényes példákat, hogy lásd, hogyan épül fel a levél fejrésze és üzenetrésze.

Ha az egyedi levelezési protokollokról további információt szeretnél, jó néhány napot el-tölthetsz a következő dokumentumok olvasásával, de az is elég, ha kiválasztod azokat, amelyekre szükséged van:

- *Comrnon Internet Message Headers (Általános Internet-üzenet fejrészek)* - **www.faqs.org/rfcs/rf c2076.html**
- *Internet Message Access Protocol (IMAP) (Internet-üzenet hozzáférési protokoll)* - **www.faqs.org/rfcs/rfc2060.html**
- *Internet Message Formát (Internet-üzenet formátumok)* - **www.faqs.org/rfcs/rfc2822.html**
- *Multtpurpose Internet Mail Extensions (MIMÉ) (Többcélú Internet levelezési kiterjesz tések)* - **www.faqs.org/rfcs/rfc2045.html; www.faqs.org/rfcs/rfc2046.html; www.faqs.org/rfcs/rfc2047.html; www.faqs.org/rfcs/rfc2Q48.html; www.faqs.org/rfcs/rfc2049.html;**
- *Network News Transfer Protocol (NNTP) (Hálózati hírtovábbító protokoll)* - **www.faqs.org/rfcs/rfc977.html**
- *Post Office Protocol Version 3 (POP3) (Postai protokoll 3. verzió)* - **www.faqs.org/rfcs/rfcl939.html**
- *Simple Mail Transfer Protocol (SMTP) (Egyszerű levelezéstovábbító protokoll)* - **www.faqs.org/rfcs/rfc2821.html**

A **PHP** levelezési támogatása a mail()-függvényeket, illetve a fejezet későbbi részében felsorolt IMAP-függvényeket tartalmazza. Van egy ezmlm_hash()-függvény is az EZMLM levelezőlista-kezelő felhasználói számára.

IMAP

A PHP IMAP-függvényei kezelik az IMAP-protokollt, továbbá a POP3-t, az SMTP-t, az NNTP-t és a hely postaládaelérését. Sok levelezőszerver elérésére az IMAP a szabványos protokoll, de sokkal több, mint elég, ha csak egy levelet akarsz küldeni. Az IMAP-függvények a következők:

Postaládafüggvények

A postaládafüggvények a postaládáidhoz való hozzáférést kezelik. A későbbi levelezési függvények kezelik az egyes levelekhez való hozzáférést. Először postaládát kell választanod, és csak utána levelet. A hírszervereken a hírcsoportok a postaládák megfelelői:

- **imap_open()** - A postaláda nevét, egy felhasználói nevet, egy jelszót és opcionális jelzőket fogad el. Egy erőforrás-azonosítót ad vissza az IMAP-szerveren levő postalá dához való csatlakozáshoz vagy egy POP3-csatlakozáshoz, vagy egy NNTP-szerverhez. A postaláda neve tartalmazza a szerver nevét, egy opcionális portnevet és egy postaláda nevet. A felhasználó alapértelmezett postaládája az INBOX, így a helyi szerveren levő postaláda a {localhost}INBOX. Ha szükség van a port számára, akkor ugyanez a {localhost:143}INBOX lenne. A POP3-hoz a {localhost:110/POP3}INBOX-szal férsz hozzá. A jelzők csak olvasási hozzáférést engednek, illetve névtelen hozzáférést a hírekhez (NNTP), és egy opciót arra vonatkozóan, hogy a kapcsolatot postaláda megnyitása nélkül hozza létre.
- **imap_close()** - **Az imap_open()** által megadott erőforrás-azonosítót fogadja el, és le zárja a kapcsolatot.
- **imap_getmailboxes()** - Ezt a függvényt kell használnod, amint megnyitod a kapcso latot. A függvény az **imap_open()-ből** származó csatlakozási azonosítót, egy hivatko zási sztringet és egy mintasztringet fogad el. Objektumok tömbjét adja vissza, az ob jektumok mindegyike a postaládára vonatkozó információt tartalmaz. A hivatkozási sztring a lista kiindulási pontja, így a nevek hierarchiájában kezdheted egy ponttal lej jebb a listát. A mintával a nevek egy alhalmazát választhatod ki, úgy működik, mint a fájlkeresés, így a * mindent visszaad, az s* pedig az s-sel kezdődő neveket. A vissza kapott tömböt használhatod legördülő kiválasztási listaként, hogy a felhasználó másik postaládát választhasson. Megkaphatod a postaládában levő üzenetek számát és - ami ennél fontosabb - az olvasatianként megjelölt üzenetek számát - vagyis a még nem olvasott üzeneteket.
- **imap_listmailbox()** - **Az imap_getmailboxes()** rövid változata. Ugyanazokat a para métereket fogadja el, de csak postaládák nevét tartalmazó tömböt ad vissza.
- **imap_scanmailbox()** - Csatlakozási azonosítót, hivatkozási sztringet, mintát és kere sési sztringet fogad el. Egy tömböt ad vissza, amely azokat a postaládákat tartalmaz za, amelyek adataiban megvan a keresési sztring. Az **imap_getmailboxes()-t** bemuta tó szakaszban olvashatsz a hivatkozási sztringről és a mintáról.
- **imap_search()** - Csatlakozási azonosítót, keresési kritériumot és egy jelzőt fogad el. Az aktuális postaládából egy tömböt ad vissza, amely tömb a keresési kritériumnak megfelelő üzeneteket tartalmazza. A kritérium sztring szóközzel elválasztott kulcs szavakat tartalmazhat. A szóközt tartalmazó kulcsszavakat idézőjel közé kell rakni. A jelző lehet **SE_UID,** amennyiben azt akarod, hogy a tömb az üzenetszámok helyett UID-ket tartalmazzon. A kulcsszavak az RFC2060-ban **(www.faqs.org/rfcs/rfc2060.html)** a 6.4.4 fejezetben vannak felsorolva, és többek kö-zött olyan egyszavas kulcsszavakat tartalmaznak, mint az **ANSWERED,** amellyel az olyan üzeneteket kapod meg, amire válaszoltak, vagy olyan sztringpárokat, mint a

CC "john", amely az olyan üzeneteket adja vissza, amelyeket a **john** sztringet tartalmazó címekre küldtek másolatként.

- **imap_reopen()** - Ha a felhasználó egy másik postaládát választ, az **imap_reopen()** egy csatlakozási azonosítót, a kiválasztott postafiók nevét és az opcionális jelzőket fogadja el. Ezután a kapcsolatot az új postaládára változtatja meg. Az opcionális jelzők ugyanazok, mint az **imap_open()** esetében.
- **imap_check()** - Csatlakozási azonosítót fogad el, és egy az aktuális postaládára vo natkozó információkat, így az üzenetek számát is tartalmazó objektumot ad vissza.
- **imap_status()** - A postaládára vonatkozó információkat tartalmazó objektumot ad vissza. A függvény csatlakozási azonosítót, a postaláda nevét és néhány opciót igé nyel, utóbbiak közül az **SA_ALL** a leghasznosabb. A postaláda nevének formátumát az **imap_open()-nél** írtam le.
- **imap_num_msg()** — Csatlakozási azonosítót fogad el, és az aktuális postaládában levő üzenetek számát adja vissza.
- **imap_num_recent()** - Csatlakozási azonosítót fogad el, az aktuális postaláda legfris sebb üzeneteinek számát adja vissza. Legfrissebb üzenetnek azok a levelek minősülnek, amelyek újak a postaládában vagy azért, mert újonnan érkeztek, vagy mert bár régebbiek, de most lettek a postaládába másolva.
- **imap_mailboxmsginfo()** - Hasonló az **imap_status()-hoz,** de az **imap_mailboxmsginio()** futása tovább tart, mert több információt gyűjt össze az **üzenetek beolvasásakor,** mint az **Ímap_status().** Az információt egy objektumban ad ja vissza, amely az utolsó módosítás dátumát, az üzenetek számát, a legfrissebb üze netek számát, az olvasatlan üzenetek számát, a törölt üzenetek számát és a postaláda méretét adja vissza. Ezt a függvényt akkor használd az **imap_status()** helyett, amikor régi leveleket akarsz törölni, hogy lemezterületet nyerj.
- **imap_expunge()** - Az aktuális postaláda összes törölt üzenetét eltávolítja. Az üzene tek egyenkénti törléséhez a **imap_delete()-et** használd.

;-- — **imap_createmailbox()** — Vadonatúj postaládát hoz létre, amikor meguntad a régieket. A függvény egy csatlakozási azonosítót és egy postaládanevet fogad el — utóbbit az **imap_open()-nél** meghatározott formátumban.

- **imap_deletemailbox()** — Törli a postaládát, amikor már nem akarod használni. A függvény egy csatlakozási azonosítót és egy postaládanevet fogad el - utóbbit az imap_open()-nél meghatározott formátumban.
- **imap_renamemailbox()** - Vadonatúj nevet ad a postaládádnak. A függvény egy csat lakozási azonosítót, a postaláda jelenlegi és új nevét fogadja el. Mindkét névnek az **imap_open()-nél** meghatározott formátumban kell szerepelnie.

Levelezési függvények t arj 3í*J3íís

A levelezési függvények magukat az egyedi leveleket vagy hírcsoportpostát olvassák, és feltételezik, hogy korábban kiválasztottad a szervert és a postaládát vagy a hírcsoportot:

imap_headers() - Csatlakozási azonosítót fogad el, amennyiben a felhasználónak már sikerült kiválasztania a postaládáját, és az abban a postaládában levő összes levél fejrészét adja vissza egy tömbben.

imap_headerínfo() - Csatlakozási azonosítót, az imap_headers()-bó'l származó üzenetszámot és néhány opcionális paramétert fogad el. Egy a levelek fejrészével teli objektumot ad vissza.

imap_header() - Az imapheaderinfoQ aliasa.

imap_fetch_overview() - Csatlakozási azonosítót, az üzenetszámok sorrendjét tartalmazó sztringet és opcionális jelzőket fogad el. Egy levelenként egy objektumot tartalmazó tömböt ad vissza. Az objektumok a tárgyat, a feladót, méretet és dátumot tartalmazzák, illetve azt, hogy a levelet olvasták-e már.

imap_sort() - A levél fejrészeit sorrendben tartalmazó tömböt ad vissza. A függvény csatlakozási azonosítót, kritérium-sztringet, egy fordított sorrend paramétert, és egy opciós paramétert tartalmaz. A kritérium-sztring olyan értéket tartalmazhat, mint a SORTDATE, hogy a fejrészeket a fejrészmezők valamelyike szerint rendezze. Ha a fordított sorrend paraméter l-re van állítva, akkor a rendezési sorrend fordított.

imap_body() - Csatlakozási azonosítót, egy üzenetszámot és opcionális jelzőket tartalmaz. Az üzenet törzsét adja vissza sztringként. Ha az üzenet többrészes MIME-üzenet, az üzenet szerkezetének feltérképezéséhez használd az imap_fetchstructure()-t, majd az üzenet egyes részeit az imap_fetchbody()-val kaphatod meg.

imap_fetchstructure() - A többrészes MIME-üzenetek szerkezetét mutatja meg. Az üzenet egyes részeit az imap_fetchbody()-val kaphatod meg.

imap_fetchbody() - A többrészes MIME-üzenet részeit adja meg.

imap_mail_compose() - Egy üzenetet hoz létre sztringben, az imap_append()-ben való használatra. Az input egy üzenetfejekből álló tömb és egy az üzenet törzsrészeit tartalmazó tömb. Bármelyik rész lehet fejrészből és szövegből álló tömb, ami csak egy kicsit teszi könnyebbé a többrészes üzenetek létrehozását. A függvényt nem-IMAP MIME-levelezéshez használhatod.

imap_append() — Üzenetet ad hozzá a postaládához. A függvény csatlakozási azonosítót, az imap_open()-ben használt postaládanévnek megfelelő nevet, az üzenetet tartalmazó sztringet és opcionális jelzőket fogad el. Igazat ad vissza, ha a hozzáadás sikeres, hamisat, ha nem. Az üzenetsztring formátuma hasonló a mail() használatát bemutató példában látható levelekéhez. Az imap_append()-ben a fejrészek, például a Címzett és a Feladó az üzenetsztringben vannak ahelyett, hogy külön paraméterként kerülnének átadásra.

imap_delete() - Csatlakozási azonosítót és üzenetszámot fogad el, és kitörli az üzenetet. A törölt üzenetek törökként vannak jelölve és addig a postaládában maradnak, amíg az imap_expunge()-ot nem használod a postaládán.

imap_undelete() - Az imap_delete() fordítottja. Csatlakozási azonosítót és üzenetszámot fogad el, és eltávolítja a törlési jelzőt az üzenetről. A törölt üzenetek törökként vannak jelölve, és addig a postaládában maradnak, amíg az imap_expunge()-ot

nem használod a postaládán. Ha már eltávolítottad a törölt elemeket, nem fogod visszakapni őket.

- imap_mail() - Olvasd el az „Egy levél küldése" című Gyors megoldások részben a mail()-ről leírtakat, majd próbáld ki az imap_mail()-t egy IMAP-szerverrel. Az imap_mail() majdnem ugyanazokat a paramétereket igényli, mint a mailQ, és ugyan azt az eredményt produkálja.
- imap_mail_copy() - Csatlakozási azonosítót, az üzenetszámokat tartalmazó sztringet, a célpostaláda nevét és néhány opcionális jelzőt fogad el. Az üzeneteket az aktuális postaládából a célpostaládába másolja.
- imap_mail_move() - Az imap_mail_copy()-val megegyező paramétereket fogad el, és az aktuális postaládából a célpostaládába helyezi át a leveleket.

Kiemelőfüggvények

Az IMAP-kiemelés eszköz a sok postaláda közül néhánynak a hangsúlyozására. Tegyük fel, hogy a munkahelyeden postaládák százaihoz van hozzáférésed, amelyekben a különböző termékek iránt érdeklődő ügyfelek levelei vannak, és valahol a sok között lapulnak a privát postaládáid. Egyszerűen kiemelheted a saját postaládáidat: amikor a magánleveleidet akarod, a hagyományos függvényeket helyettesítsd a kiemelőfüggvényekkel:

- imap_subscribe() — Csatlakozási azonosítót és postaládanevet fogad el, és a postaládát a kiemelési listádba teszi. A postaláda neve az imap_open()-nél leírt formátumban kell, hogy legyen.
- imap_getsubscribed() - Majdnem teljesen megegyezik az imap_getmailboxes()-zel. Míg az imap_getmailboxes() az összes postaládát visszaadja, az imap_getsubscribed() csak a kiemelési listán levő függvényeket adja vissza.
- imap_listsubscribed() - Majdnem teljesen megegyezik az imap_listmailbox()-szal. Míg az imap_listmailbox() az összes postaládát visszaadja, az imap_listsubscribed() csak a kiemelési listán levő függvényeket adja vissza.
- imap_unsubscribe() - Csatlakozási azonosítót és postaládanevet fogad el, és a postaládát eltávolítja a kiemelési listáról. A postaláda neve az imap_open()-nél leírt formátumban kell, hogy legyen.

Sztringkonvertáló függvények

Jelen szakasz függvényei a sztringeket különböző megjelenítésekre kódolják, amilyen például az UTF, és párosító kikódoló függvényeket tartalmaznak.

Az UTF UCS transzformációs formátumokat jelent. Az UCS az Unicode 16 bites karakterkészletet jelenti, amellyel a világ összes írott nyelve megjeleníthető, még az olyan, piktogramokat használó ideografikus nyelvek is, mint a mandarin-kínai, a koreai és a japán.

imap_8bit()

Az imap_8bit() sztringet fogad el, és idézőjeles nyomtatható sztringként formázva adja vissza, amely a levélbe az RFC 2045-nek megfelelően beilleszthető. A hosszabb sztringek a 76. karakternél vannak becsomagolva, így ezt a függvényt csak olyan sztringekre használd, amelyek a levél törzsébe kerülnek.

A következő kód az imap_8bit()-et teszteli, és az eredmény megmutatja, hogy hogy néznek ki az idézőjeles nyomtatható kódok. A példában a $before-tÖmböt egyszeres és kétszeres idézőjelet, kocsivissza jelet (carriage return) és újsort (newline) is tartalmazó szöveggel töltjük fel. Az RFC 2045 csak akkor ismeri fel az új sort, amikor azok kocsivissza-újsor párral vannak jelölve (ahogy a Windows és az NT használja), de amikor egyszerű newline-nal (újsorral, ahogy a Unix használja), akkor nem. Van egy sor a return-re, egy a newline-re és egy, amelyben mindkettő van. Az RFC 2045 a sortörést a 76. karakternél követeli meg, így vannak 76 karakteresnél hosszabb tesztsztringek. A második hosszú sor newline-okat, a harmadik pedig return-newline-okat tartalmaz.

A kód ciklussal végigfut a tömbön, konvertálja a sztringeket, és a későbbi tesztekhez tömbbe menti a konvertálásokat. A before- (előtte) és after- (utána) sztringek vannak kiíratva. Mivel a kocsivissza és az újsorok a before-sztringben nem lennének láthatók, a before-sztring URL kódolású. A szóközök + jelre változnak, a speciális karakterek pedig százalékjelre, amelyeket a hexadecimális megjelenítés követ:

```
$before [ ] =  'test of  quote, ', doublequote, \", and equal, =.
$before[] =  'test of  doublequote \""; quote carriage return \r";
$before[] =  'test of  quote newline \n";
$before[] =  'test of quote carriage return and newline \r\n";
$before[] =  'test oflong, long, long, long, long, long," long,
$before[] = " 'test oflong, long, long, long, long," lout newlines .
long, " long  long    " ;
$before[]    = text wi  long, long, long, long, long, long,\n"
      "       'test of                    long, long, long,\n"
      long,    long,
      long,    long,    long,
      "    long   text   with   newlines."; $before[]    =    "test
of   long,    long,    long,    long,    long,    long,\r\n
      "    long,    long,    long,    long,    long,    long,    long,
      long,\r\n"
      "    long   text   with   carriage   returns   and
newlines."; while(list($k,    $v)    =    each(Sbefore))

   $after[$k] = imap_8bit{ $v) ; print (
   "<brxf ont color = \ "blue\ " >" .          urlencode($v)     '</fontxbr>'
   $after[$k] .
```

Az eredményt a következőkben mutatom, minden konverziót megjegyzésekkel kísérve. Az első tesztben az egyszeres *(%27)* és kétszeres idézőjelek (%22) nem változtak, azonban az egyenlőségjel (=) =3D-re változott. Az egyenlőségjelet a kódolás részeként használja, így ha megtalálható az input-sztringben, kódolni kell. A kódolt karakterek egy egyenlőségjelre, és egy azt követő, hexadecimális karakterként (ahol a betűk nagybetűvel vannak írva) megjelenített karakterre változnak. A dekódolás alatt a dekódoló az egyenlőségjelet keresi, és az utána talált két karaktert kódolja ki. A dekódoló attól függetlenül el kell, hogy fogadja a karaktereket, hogy azok kis- vagy nagybetűsek:

```
test+of+quote%2C+%2 7%2c+doublequote%2C+%22%2C+and+equal%2C+%3D. test
of   quote,    ',    doublequote,    ",    and    equal,    =3D.
```

A következő teszt **=OD-re** konvertált kocsivissza jelet mutatja, ahogy azt RFC 2045 a nem megjeleníthető karakterekre megkívánja. Az azt megelőző szóköz is át van konvertálva =20-ra. Az RFC 2045-nek hosszú, bonyolult előírásai vannak a sor végén álló szóközök konvertálására, mert némely levelező programok kidobják a sorvégi szóközöket. Az RFC 2045 legalább egy oldallal rövidebb lenne, ha a kitalálói úgy döntöttek volna, hogy egyszerűen az összes szóközt kódolják:

```
test+of+quote+carriage+return+%OD test
of quote carriage return=20=0D
```

A következő sor az =0A-ra kódolt newlíne-t mutatja, de az előtte álló szóköz nincsen kódolva, jelezve, hogy a kód azt gondolja, hogy a szóköz nem a sor végén van, így azt nem kell kódolni. Miért különbözik ez a sor végén levő kocsivisszától? Az RFC 2045 olyan te-kervényes, hogy nem tudtam kitalálni, vajon a különbség az RFC 2045 előírásai vagy az **imap_8bít()** hibája miatt adódik-e:

```
test+of+quote+newline+%0A test
of quote newline =0A
```

A következő teszt egy kocsivissza-újsor párt tartalmaz. Az eredmény azt mutatja, hogy a kocsivissza-újsor párokat kódolás nélkül kapod vissza. Az előtte levő szóköz kódolja, hogy megakadályozza az eltávolítását. Az RFC 2045-ben egy kicsit nem egyértelmű, hogy a nem kódolt kocsivissza-újsor **párt** hogyan kellene dekódolni, mivel az RFC engedi az újsorok levelező szerverek általi hozzáadását és eltávolítását:

```
test+of+quote+carriage+return+and+newline+%OD%0A
test of quote carriage return and
newline=20
```

A következő tesztben egy 76 karakternél hosszabb szrting van, az eredmény pedig egy = jelet tartalmaz, melyet egy szóköz követ a törésnél. A szóköz valójában egy kocsivissza-újsor pár, ami egy törést okoz a szövegben, amikor az a levelező szervereken fut át, és a levelező kliensedben lesz eltávolítva. A levelező kliensekben van egy opció, hogy becsomagolják a szöveget a megjelenítőben, ha a sor meghalad egy adott hosszúságot, de általában intelligensebbek ennél, és megkeresik a szavak közötti törést. A 76 karakteres sor nem fér be a könyv tördelésébe, így minden sorból levágtam 2 "long"-ot (12 karaktert), de az eredményt így is láthatod az utolsó sorba szúrt = jellel:

```
test+of+long%2C+long%2C+long%2C+
   long%2C+long%2C+long%2C+long%2C+iong%2C+long%2C+long%2C+
   long+text+without+newlines. test of long, long,
long, long, long, long, long, long,
   long, 1= ong, long text without newlines.
```

A következő tesztben a hosszú sort újsorokkal tördeltem. Az eredmény azt mutatja, hogy az újsorok kódolva vannak, és a kliens új töréseket szúrt be a 76. karakternél. Ebben az esetben is kiszerkesztettem két „long"-ot, hogy a sorok beférjenek a könyvbe:

```
test+of+long%2C+long%2C+long%2C+long%2C%0A+long%2C+long%2C+long%2C+long%
   2C+long%2C+long%2C%0A+long+text+with+newlines. test of long, long,
long, long,=0A long, long, long, long= , long,
   =0A long text with newlines.
```

A következő tesztben a hosszú sorok kocsivissza-újsor párokkal vannak tördelve. Az eredmény azt mutatja, hogy a kocsivissza-újsor párok nincsenek kódolva, de a sorok törésére a levelező kliens felhasználja őket. Mindenesetre a trükkös munka, a kódolás, kikódolás és az RFC 2045 mind sokkal egyszerűbb lenne, ha a kocsivissza és az újsor a sztringben kódolva lenne, és az új töréseket a kódolás rakná be:

```
testkor-r!ong%2C + iong%2C+long%2C +long%2C + long%2C +long%2C%0D%0A+long%2C+
   long%2C+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C%0D%0A+long+
   text+with+carriage+returns+and+newlines.
test of long, long, long, long, long, long, long, long,
   long, long, long, long, long, long, long, long text with carriage
   returns and newlines.
```

imap_qprint()

Az **imap_qprint()** az **imap_8bit()** fordítottját csinálja, az idézőjeles nyomtatható sztringet normál sztringre konvertálja vissza. A következő kód kikódolja és **kiíratja** a **$after** első elemét:

```
print("<br>" . imap_qprint(Safter[0 ] ) ) ;
```

Az eredmény:

```
test of doublequote, and equal,
quote, =.
```

quoted_printable_decode()

***A* quoted_printable_decode()** az **imap_qprint()** egy alternatívája. Ugyanazt a konverziót kell elvégeznie, de úgy tűnik, van néhány különbség. A **quoted_printable_decode()** nem igényli az IMAP-kiterjesztést, így akkor lehet hasznos, amikor egy külső program által gyűjtött leveleket olvasol.

A következő kód visszaállítja a **$before-t,** ciklussal végigfut rajta, kikódolja az egyező értéket a Safter-ba, és kiíratja a kikódolt értéket az eredeti mellé. Mindkét érték URL kódolású, így láthatod a speciális karaktereket is. Ha a kikódolt sztring megegyezik az eredetivel, a kikódolt érték zölddel van kiemelve, ha nem, akkor a nem egyező sztring pirossal:

```
reset($before);
while (üst ($k, $v)  = each($before)

    $decoded = quoted_printable_decode($after[$k] ) ;
    print ("<brxf ont color = \ "blue\ " >" . urlencode{$v)     "</font>" );
    if($decoded == $v)

        print { "<brxf ont color=\ "green\ " >" . urlencode ( \$decoded) .
            "</fontxbr>" ) ;

    else

        print ("<brxf ont color=\"red\">    . urlencode($decoded)
              "</fontxbr>" );
```

A következő négy eredmény azt mutatja, hogy a kikódolt sztringek megegyeznek az eredetivel:

```
test+of+quote%2C+%27%2C+doublequote%2C+%2 2%2C+and+equal%2C+%3D.
test + of +quote%2C+%2 7%2C+doublequote%2C+%2 2%2C+and+equal%2C+%3D.

test+of+quote+carriage+return+%OD
test+of+quote+carriage+return+%OD

test+of+quote+newline+%OA
test+of+quote+newline+%OA

test+of+quote+carriage+return+and+newline+%OD%0A
test+of+quote+carriage+return+and+newline+%OD%0A
```

A következő két eredmény összehasonlítása azt mutatja, hogy a kikódolt sztringek nem egyeznek meg az eredetivel (a könyv tördelése miatt a megjelenített sorokat rövidebb sorokra bontottam):

```
test+of+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C+
   long%2C+long%2C+long%2C+long%2C+long%2C+long%2C+long+text+without+
   newlines.
test+of+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C+
   long%2C+long%2C+long%2C+l%0Aong%2C+long%2C+long%2C+long+text+without+
   newlines.

test+of+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C%0A+long%2C+
   long%2C+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C%0A+long+text+
   with+newlines.
test+of+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C%0A+long%2C+
   long%2C+long%2C+long%2C+long%0A%2C+long%2C+long%2C+long%2C%0A+long+
   text+with+newlines.
```

Egyik esetben sincsenek az újsorok eltávolítva az eredményül kapott sztringből. Vajon ez a **quoted_printable_decode()** hibája, vagy az RFC 2045-ben van egy lyuk? Úgy vélem, hogy az RFC túlságosan is összetett, olyan mértékben, hogy megbízható kódoló és kikódoló írása már nagyon bonyolult.

A következő eredmények összehasonlítása azt mutatja, hogy a kikódolt sztring megegyezik az eredetivel:

```
test+of+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C%0D%0A+long%2C+
   long%2C+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C%0D%0A+long+text+
   with+carriage+returns+and+newlines.
test+of+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C%0D%0A+long%2C+
   long%2C+long%2C+long%2C+long%2C+long%2C+long%2C+long%2C%0D%0A+long+text+
   with+carriage+returns+and+newlines.
```

***imap** _utf7_encode()*

Az imap_utf7_encode() egy sztringet fogad el, és a módosított UTF-7-nek megfelelően kódolt sztringet ad vissza, ahogy az az RFC 2060-ban le van írva

(www.faqs.org/rfcs/rfc2060.html). (Az **imap_utf7_decode()** a kódolt sztringet az eredeti sztringre kódolja ki.) A kódolást az olyan nemzetközi karaktereket tartalmazó postaládanevekhez használják, amelyeknek a karakterei kívül esnek a szabványos nyomtatható karakterkészleten.

Az URL-nek és más, létező kódolási formáknak lehet értelme a nevek kódolásánál, de a szoftverelérés bizonyos formáit lelassíthatják. Az RFC 2060 kódolás a könyvtárszerkezet karaktereit, például a perjelet kódolatlanul hagyja, hogy az megfeleljen az egyes postaládákat külön könyvtárként tároló levelező rendszereknek. A következő adatok azt mutatják, hogy néz ki a módosított UTF-7-kódolás. Az adatkönyvtárakban gyakran használt karaktereket, így perjelet (/), hullámjelet (~), and-jelet (&), gondolatjelet (-) és görög P karaktert tartalmaz:

```
$before[]       "ma i1/i n";
$before[]       "~john";
$before[]       "amper&sand";
$before[]       "new-mail"; "Basic
$before[]       Greek:      §"; "Basic
$before[]       Greek:      §§"
```

Az adatokat ugyanabba a kódba táplálom be, mint amit az **imap 8bit()** tesztelésénél használtam, azzal a különbséggel, hogy az **imap_8bit()-et** az **imap_utf7_encode()-re** cseréltem:

```
$after[$k]      =       imap_utf7_encode($v);
```

A /, ~ és - jeleket tartalmazó eredmények nem különböznek az inputoktól, így kihagytam őket. Az & jelet &—ra fordította, ahogy itt láthatod:

```
amper&-sand
```

A görög karakter a következő eredményben a %DF, &3w- -ra fordítódott. A & egy kódolt sztring kezdetét jelzi, a - pedig a végét. A 3w magát a karaktert jelzi:

```
Basic^Greek%3A+%DF
Basic     Greek:
&3w-
```

imap_u tf7_de co de ()

Az **imap_utf7_decode()** egy sztringet fogad el, és az RFC 2060 szerint kikódolt sztringet adja vissza. (Az **imap_utf7_encode()** végzi a kódolást.)

A kikódoló függvény teszteléséhez az adatokat ugyanabba a kódba **tápláltam** be, amit a **quoted_printable_decode()** teszteléséhez használtam. Az egyetlen különbség, mint az látható, hogy a **quoted_printable_decode()** helyett az imap_utf7_decode()-függvény szerepel. A kikódolt szöveg megtekintéséhez írasd ki az eredményt:

```
$decoded    =    imap_utf7_decode($after[0] ) ;
```

Az összes output-sztringet összehasonlítottam a $before-ban levő eredeti sztringekkel; a kódolás és kikódolás tökéletesen működött.

imap_utf8()
Az imap_utf8()-sztringet fogad el, és az UTF8-ra kódolva adja azt vissza (RFC 2044).

imap_base64()
A Base64 kódolást a fejezet „MIME"-részében taglalom részletesen. Az imap_base64() a base64_decode()-függvénynek az IMAP-függvény megfelelője, amely egy kódolt sztringet fogad el, és a kikódolt sztringet adja vissza.

imap_binary()
Az imap_binary() a base64_encode()-függvény IMAP-függvény megfelelője, amely egy sztringet fogad el, és a base64-szabványra kódolt sztringet ad vissza.

Egyéb függvények

Az előző részekben a postaláda eléréséhez és a postaládán belüli elemek eléréséhez használt függvényekkel foglalkoztam. Ebben a részben a többi IMAP-függvényről, így a hibakezelő-függvényekről, a tesztelésre használt függvényekről, illetve a korábban leírt függvényeknél manuálisabb megközelítéssel dolgozó függvényekről lesz szó.

imap_aíerts()
Az imap_alerts() az IMAP-szerver hibaüzeneteit tartalmazó tömböt ad vissza. A szerver hibaüzeneteinek listája meg van tisztítva, így minden üzenetet csak egyszer kapsz.

imap_clearflag_full() és imap_setflag_full()
Az egyes levélen levő jelzők azt mutatják meg, hogy a levelet még nem olvasták, piszkozat vagy ki van törölve. Az imap_clearflag_full()- és az imap_setflag_fuli()-függvények beállítják és törlik a jelzőket. Általában a törlési jelzőt az imap_delete()-tel állítják be, és az ol-vasatlan jelző is törlődik, amikor törlőd a levelet, így erre a két függvényre nem igazán van szükség.

imap_errors()
Olvasd el az imap_alerts()-t, mert a figyelmeztetések olyan helyzeteket jelezhetnek előre, amelyek hibához vezethetnek. Az imap_errors() egy tömböt ad vissza, amely a korábbi hibaüzeneteket tartalmazza, és törli a szerver hibalistáját, így minden hibaüzenetet csak egyszer kapsz meg. Amikor új IMAP-kódot tesztelsz, ellenőrizd a figyelmeztetéseket és a hibákat minden IMAP-függvény után, míg meg nem győződsz arról, hogy minden működik.

imap_last_error()
Az imap_last_error() az utolsó hibaüzenetet adja vissza, és nem változtatja meg az IMAP-szerver hibalistáját (vagyis azt, amit az imap_errors() ad vissza). Ez a függvény csak egy hibaüzenetet ad vissza, így lehet, hogy egy fontos nyomot hagysz figyelmen kívül, ha az utolsó hibát valójában egy azt megelőző hiba okozta. Ha az imap_last error()-t használod, alkalmanként használd az imap_errors()-t is, hogy ellenőrizd és töröld az összes hibát.

imap_fetchheader()
Az imap_fetchheader() az imap_header() egyszerűsített változata, amely egy csatlakozási azonosítót, egy üzenetszámot és néhány jelzőt fogad el. Egy nyers fejrészsztringet ad

vissza, amely az RFC 2822-höz (www.faqs.org/rfcs/rfc2822.html) alkalmazkodik. Tőled függ, hogy kikódolod-e a fejrészeket. Használd az imap_header()-t, ha pedig elakadsz a furcsa eredményekkel, próbáld ki az imap_fetchheader()-t annak megállapítására, hogy a problémát új, szokatlan vagy hibás fejrész okozta.

imap_mime_header__decode()

Az imap_mime_header_decode() a MIME-fejrészsztringeket egy tömbbe kódolja ki, ahol minden elemnek két értéke van, a charset és a text. Ha a szöveg az alapértelmezett us_ascii karakterkészletben van, akkor a charset a default-ot tartalmazza. A következő' kóddal, amely a charset-et és a szöveget egy egyszerű listában íratja ki, gyorsan letesztelheted a MIME-fejrészeket. A htmlentities() segítségével elkerülheted az oldal HTML-jét felbolygató karaktereket. Ahhoz, hogy minden egyes karakterkészlet helyesen jelenjen meg, be kell állítani a HTML-t vagy ami a böngésződben működik. A kód feltételezi, hogy a $header-t az imap_header()-rel hoztad létre:

```
$strings = imap_mime_header_decode($header);
while(list($k, $v) = each($strings))
    {
    print("<br>charset: " . $v["charset"] . ", text: " .
        htmlentities($v["text"]));
```

imap_uid()

Az imap_uid() csatlakozási azonosítót és üzenetszámot fogad el, és az üzenet UID-jét adja vissza. Az imap_msgno() az UID-bó'l adja vissza az üzenetszámot. Az UID egy 32 bites üzenetazonosító, amely egyedi a postaládán belül, és a törlések és egyéb postaládán belüli műveletek után is az marad. Az UID-t használhatod arra, hogy egy offline session után összehasonlítsd a leveleket a szerveren levőkkel.

imap_msgno()

Az imap_msgno() csatlakozási azonosítót és UID-t fogad el, és az UID-nek megfelelő üzenet számát adja vissza. Az imap_uid() az üzenetszámból adja vissza az UID-t. Az UID magyarázatát az imap_uid()-nél találod.

imap_ping()

Ha azt gyanítod, hogy egy LDAP-szerver offline-ra dobott, vagy emlékeztetni akarod a szervert, hogy te is itt vagy, használd az imap_ping()-et. Az imap_ping()-nek csak a csatlakozási azonosítóra van szüksége, és a függvény igazat ad vissza, ha a kapcsolat még él.

Akkor használhatod ezt, ha írtál egy kötegelt PHP-programot arra, hogy a háttérben üldögéljen, és olyan hosszú távú levelezési feladatokat végezzen, mint a nem kívánt reklámlevelek ellenőrzése vagy a szabadságon levő emberek leveleinek átirányítása. A program csak ott ül, és néhány percenként ellenőrzi a postaládát az új üzenetekért. Ahhoz, hogy az egyes ellenőrzések között életben tartsd a kapcsolatot, használd az imap_ping()-et.

imap_rfc822_parse_adrlist()

Az imap_rfc822_parse_adrlist() egy címsztringet és egy alapértelmezett host-nevet fogad el. Egy tömböt ad vissza, amelyben az egyes címekhez négy értéket tartalmazó objektumok

vannak. A négy érték: **mailbox, hőst, personal** (a személy neve) és **adl** (at domain list; vagyis címek a @**petermoulding.com** formátumban). A következő kód egy címsztringet elemez, és kiíratja az eredményt. A függvény olyan sztringek esetén hasznos, amelyek több címet tartalmazhatnak, például a To (Címzett), Cc (Másolatot kap) és **Bcc** (Titkos másolatot kap) mezők:

```
Saddresses = imap_rfc822„parse_adrlist($address, "a_test_domain.com")
; while(list($k, $v) = each(Saddresses)) {
    print("<br>mailbox: " .
          htmlentities($v->mailbox) ", hőst: " .
          htmlentities($v->host) ", personal: " .
          htmlentities($v->personal) ", domain: " .
          htmlentities($v->adl) ) ;
```

imap_rfc822_parse_headers()

Az **imap_rfc822_parse_headers()** az **imap_header()** egyszerűsített változata, amely egy fejrészeket tartalmazó sztringet és az alapértelmezett host-nevet fogadja el, és a fejrész értékeit tartalmazó objektumot ad vissza.

imap_rfc822_write_address()

Az **imap_rfc822_write_address()** a postaláda nevét, a hőst nevét és egy személy nevét fogadja el. Egy e-mailcímet ad vissza, az RFC 2822-re formázva. A következő kóddal tesztelheted le:

```
print("<br>address: " .
    htmlentities(imap_rfc822_write_address{ "peter",
    "petermoulding.com", "Péter M")));
```

Az eredmény:

```
address: Peter M <peter@petermoulding.com>
```

Levélfej részek

Sokféle fejrész van használatban, ezek közül némelyek minden levélben kötelezőek, a többi pedig opcionális. A MIME-fejrészek csak akkor szükségesek, ha a levél csatolt állomány(ok)at tartalmaz.

Minimális fejrészek

íme egy példa a leveleknél és híreknél minimálisan szükséges fejrészekre, amelyeket úgy hoztam létre, ahogy azt a „Levél küldése" Gyors megoldásban láthatod. A To (Címzett) fejrész a megfelelő postaládába irányítja a levelet. A **From** (Feladó) fejrész segítségével a levél fogadója válaszolni tud rá. A **Date** (Dátum) fejrész segítségével megtudhatod, melyik levél mikor érkezett, a **Subject** (Tárgy) fejrész pedig jelzi, hogy miről szól a levél. A To, From és Subject a minimálisan szükséges azonosítók az olyan függvényekben, mint a **mailQ,** a levelezőrendszer pedig beszúrja a dátumot:

```
Subject: PHP Sydney meeting
   Date: Sun, 01 Jul 2001 19:51:48 D1000
   From: address@the_mail_server.com.au
   To: peter@a_web_site_somewhere.com
```

További fejrészek

Itt találod a többi fejrészt, amelyeket használhatsz a kísérletezéseid során. Elérhetőségük és értelmezésük a levelező klienstől és a szerver szoftverétől függ. Vannak szoftverek, ame^ lyek megkülönböztetik a kis- és nagybetűket, így a fejrészeket pontosan úgy gépeld be, ahogy itt látod:

- **Mailing-List** - Ezzel a fejrésszel megnevezheted a levelezőlistát, amelyet a levél cím zéséhez használtál. Ez is egy módja annak, hogy tudasd az emberekkel, hogy miért kapták a levelet, de nem a legjobb megoldás, mert sokan nem tudják, hogy tekintsék meg a fejrészeket.
- Bcc - Ez a fejrész több levelezési címet is elfogad, és minden címzettnek másolatot küld, de a Bcc-címek egyikét sem tünteti fel a levélben. Ha levelet küldesz egy alkal mazottnak, küldhetsz másolatot a humán erőforrás osztálynak Bcc-vel. Előbb-utóbb azonban az alkalmazott úgyis hall pletykákat a levelek titkos másolatairól, így nem fog bízni az e-mailekben. A Cc őszintébb, nyíltabb, és mindenképpen azt kell a Bcc-vel szemben preferálni.
- Reply_To - Ez a fejrész olyan címet ad meg, ahova a fogadó akkor válaszolhat, ha a From-cím nem megfelelő, például amikor egy értékesítő küld neked egy levelet, de azt akarja, hogy egy általános sales-es levélcímre válaszolj, amíg ő szabadságon van.
- **Return_Path** - Ez a fejrész mutatja a levelezőrendszernek, hogy hova küldje a hiba üzeneteket.
- **X-Mailer** - Ez a fejrész megnevezi a levél elküldéséhez használt terméket. Akkor használható, ha hirdetni akarod mondjuk a PHP-t (vagy magadat).
- **X-Priority** - **1** Levelednek a legmagasabb prioritást adja, de ritkán van hatása. Ha va lamilyen probléma lelassítja a levelezést, akkor az minden levelet lelassít. Fölöslegesen ne jelöld meg sürgősnek a levelet, mert ez az egyik nyom, amire azok a nyavalyások figyelnek, akik kéretlen kereskedelmi levelekkel elárasztják a rendszert.

MIMÉ

A MIMÉ a Multipurpose Internet Mail Extensions (Többcélú internet levelezési kiterjesztések) egy csoportja, amely kiterjeszti a levél eredeti RFC 822-es definícióját. Valójában az RFC 822 nem sokkal csinál többet annál, mint elmondja, hogy irányíts egy sztringet az Interneten. A MIME-kiterjesztések adják meg a különböző karakterkészletek és fájlmellékletek használatának szabályait és irányelveit.

A MIME-fejrészeket a csatolt állományokhoz és a speciális karakterkészletekhez kell használni. A MIME-fejrészeket a következő részben találod.

MIME-Version

A MIME-Version akkor szükséges, ha a levélben bármi is MIME-kódolásban van, és ekkor a MIMÉ verziószámát mutatja meg. Az aktuális és egyetlen verzió az 1.0, ahogy azt a következő fejrész mutatja:

```
MIME-Version: 1.0
```

Tehetsz megjegyzést a MIME-Version fejrészbe, de a fogadó szoftver figyelmen kívül fogja hagyni:

```
MIME-Version: 1.0 (Produced by my PHP script)
```

Content-Type

A Content-Type alapértelmezésben a text/plain; charset = "us-ascii", de sok más értékre is beállítható, például text/html; charset=iso-8859-l-re, ha HTML is van a leveledben. A következő kódpélda egy olyan e-mailből való, amely tartalmaz néhány különleges karaktert, például ®-t. Vannak Content-Type-fejrészek, amelyekben idézőjelek között van a charset-hez rendelt érték, vannak, amelyekben nem. Vedd észre, hogy ez a fejrész pontosvesszőt (;) használ az értékek elválasztására, míg némely más fejrészek a vesszőt használják:

```
Content-Type: text/plain; charset="iso-8859-1"
```

Ha a levél csatolt állományokat tartalmaz, a Content-Type-fejrész úgy néz ki, mint a következő fejrész (minden egy sorban). A boundary-érték egyfajta pszeudo-véletlen sztring, amely az egyes csatolt állományok elejét és végét jelzik:

```
Content-Type: múltipart/mixed;
    boundary = "----------- 53B319 8E1AEEC7 DDA5 89 60 06"
```

Minden egyes csatolt állomány az állományra specifikus fejrészek halmazát tartalmazza. A következő három fejrész egy programot tartalmazó csatolt állomány előtt volt. Vedd észre, hogy a csatolt állomány kódolása base64-es, amelyet a 13. fejezetben mutatok meg:

```
Content-Type: application/octet-stream; name="test.exe"
Content-Transfer-Encoding: base64 Content-Disposition:
attachment; filename="test.exe"
```

Content-Transfer-Encoding

A Content-Transfer-Encoding a legtöbb olyan levélben megtalálható, amely tartalmazza a Content-Type-fejrészt. Olyan opciói vannak, mint a 7bit, 8bit, base64, quoted-printable és binary, és egyedi kódolási típusokat is engedélyez. Az alapértelmezett a 7bit, így a szokásos 7bit ASCII e-mailekből ez a fejrész általában kimarad, de minden más esetében használni kell. A Content-Type alatt láthatsz rá példát.

Content-ID

A **Content-ID-t** már a Web előtt létrehozták. Segítségével ahelyett, hogy csatolnád az állományokat, csak hivatkoznod kell rájuk, A fogadó kiválaszthatja a számára szükséges állományokat anélkül, hogy az összest meg kellene kapnia. Jó néhány megvalósítása van az ötletnek, amelyeket a weblapokon fájlokra mutató linkek modern módszere idejétmúlttá tett.

Content-Description

Szeretnél leveledben megjegyzést fűzni a csatolt állományokhoz? A legtöbb ember egyszerűen ír egy bekezdést az állományról. A **Content-Description** segítségével formálisan adhatsz megjegyzést az állományhoz, hasonlóan a HTML képtag-ek **alt** szövegéhez. A következő példa egy olyan megjegyzést mutat, amelyet akkor írnál, ha te lennél a Happy Virus szerzője:

Content-Description: The attachment is not a virus. Ha! Ha! Ha!

Content-Disposition

A Content-Disposition egy kísérleti fejrész, mely azt mondja meg a levél olvasójának, hogy egy csatolt állományt belsőleg (a levélablakban) kell-e megtekinteni vagy sem, ahogy a következő példa is mutatja. Ha egy üzenet néhány képet és egy programot tartalmaz, a szerző valószínűleg azt akarja, hogy az olvasó a képeket belsőleg nézze meg, de a programot természetesen ne. A levelező programodban kellene, hogy legyen olyan beállítás, hogy a **Content-Disposition-t** felülírhasd, így a levél szövegén kívül semmi mást ne láss, és csak utána dönthess arról, hogy mi mást akarsz megnézni. Ennek az olyan állománytípusoknál van nagy jelentősége, amelyek vírusokat, illetve olyan weboldalakra mutató linkeket továbbíthatnak, amely oldalakon zavaró hirdetések tucatjait találod:

Content-Disposition: inline

Működési üzemmódok

Az elektronikus leveleket legalább háromféle módon érheted el. Az aktuális üzemmód segíthet a PHP alapú levelezés legjobb megközelítésének kiválasztásában.

Offline

Offline módban a levelező kliensed vagy levelező felhasználói ügynököd (mail user agent MUA) leszedi egy levelező szerverről az új leveleket és a munkaállomásodon tárolja azokat. A leveleket a munkaállomásodon böngészed, ott válaszolsz vagy írsz új leveleket, majd a levelező szerverre küldöd őket, hogy az Internetre kerüljenek.

Az offline levélgyűjtés alapértelmezett protokollja a POP3, mert a POP3 mindent megcsinál, amire offline módban szükséged van, és semmi feleslegeset nem tesz. Az SMTP-t levélküldésre használják, és jól passzol a POP3-hoz. A POP3 és az SMTP egy alternatívája az IMAP, de az IMAP a szükségesnél többet is elvégez, többletráfordításokat és telepítési

problémákat okoz, és csak néhány e-mail kliens támogatja. A legtöbb Internet-szolgáltató a POP3/SMTP-kombinációt használja az IMAP helyett.

Ha a PHP használatával akarod a saját levelező kliensed megírni, telepítsd a munkaállomásodon az Apache-ot, a PHP-t és a MySQ

L-t, és a PHP-t használ az e-mailek munkaállomásról való visszakeresésére, megjelenítésére és elküldésére. A PHP csatlakozó függvényei - amelyeket a 13. fejezetben mutatok be - használhatók az e-mailek POP3-on keresztüli visszakeresésére, az IMAP-függvények az IMAP-on keresztül visszakeresi

k és elküldik az e-maileket, a levelező függvények pedig az SMTP-n keresztül küldik el őket.

Az offhne mód valódi hátránya akkor derül ki, amikor távol vagy a gépedtől, és igazán fontos információra, például egy hotel nevére van szükséged, amely a gépeden figyel egy e-mailben. Ha géped történetesen egy notebook, akkor ott lehet veled a levél, illetve ott lehet egy tolvajnál, aki ellopja egy reptéri poggyászkocsiról a bőröndöd.

Egyszer egyik barátom este érkezett egy nagy ázsiai városba, és kénytelen volt szembesülni azzal, hogy a csomagjai máshova érkeztek meg. Az összes kontakt telefonszám a csomagban volt, a biztonsági másolat pedig egy iroda fiókjában lapult egy floppyn, történetesen nem Ázsiában. Az irodai dolgozók pedig otthon legmélyebb álmukat aludtak. Ha szerencséd van és olyan cégnél dolgozol, amelyik mindig a Hiltonba foglal szobát, akkor függetlenül attól, hogy melyik városban vagy, egyszerű a megoldás. Fogsz egy taxit, mivel minden taxisofőr tudja, hol a Hilton. A barátom ott állt a repülőtéren, és nagyon szeretett volna ilyen cégnél dolgozni.

Online

Amikor a levelezést online éred el, ugyanúgy böngészed a leveleidet, mint a webszerver bármelyik más fájlját. A levelek a szerveren maradnak. A kliens csupán egy böngésző, amely a világ bármely munkaállomásán lehet - még a San Francisco-i Quetzal Internet Café egy kísértetiesen zöld iMac-jén is. A kisméretű iMac-billentyűzet lelassíthatja a gépelést, de nem akadályozza meg, hogy elérd a három menüvel és hat mozival odébb levő szervert.

A POP3 és az **SMTP** nem működnek online hozzáférés esetén. A szerver és a munkaállomásod közötti protokoll a HTTP és az IMAP kombinációja. Figyeld meg hogyan használhatod ezeket.

Ha csak tisztán IMAP-ot akarsz a gépedről a levelező szerverre, telepíts egy az IMAP-nek megfelelő kliensalkalmazást a gépedre, vagy telepítsd az Apache-ot és a PHP-t, és engedd, hogy a PHP az IMAP-pal kommunikáljon a levelező szerverrel. A Windows-felhasználók az Apache helyett használhatnak Personal Web Services-t (PWS), de a PWS-ből működő alkalmazásokat varázsolni pont olyan nehéz, mint az Apache/PHP kombinációjának telepítése. A PHP-szkriptek IMAP-függvényeket használnak a megfelelő szerverekkel folytatott IMAP-kommunikációhoz.

Ha nem akarod a PHP-t a gépedre telepíteni (ami nagyjából olyan, mint amikor egy színész visszautasítja az Oscar-díjat), beállíthatod a webszervered, hogy levelező kliensként működjön és az oldalakat bármilyen böngészőből bárhonnan böngészheted. A levelek a szerveren maradnak, a webszerver végzi a levelező szerverrel az IMAP-kommunikációt, a böngésző pedig a webszerverrel folytatott HTTP-kommunikációt.

Kapcsolat nélkül

A kapcsolat nélküli mód a notebookokhoz és betárcsázáshoz való. A notebook levelező kliense tárcsázza az Internet-szolgáltatót, csatlakozik a levelező szerverhez, begyűjti a leveleket, és bontja a vonalat. Az eredeti levelek a levelező' szerveren maradnak, a notebookon levő másolatokkal dolgozol, és olyan gyakran tárcsázol be, hogy szinkronban tartsd a kliensed a szerverrel. A kimenő levelek várnak a csatlakozásra. A levelek törlése addig nem változtatja meg a szerveren levő leveleket, amíg nem csatlakozol. Ha ellopják a notebookod, veszel egy újat, betárcsázol és az új notebook a még a szerveren levő összes levél által életre kel.

A kapcsolat nélküli módot nem igazán tudod egy Internet-cafében használni, mert ott általában nem engedik meg, hogy megváltoztasd a szoftver-konfigurációkat. Nincs sok értelme PHP-val levelező klienst írni a kapcsolat nélküli módhoz, mert a kliensnek ismernie kell a tárcsázással kapcsolatos sajátosságokat betárcsázást fogadó szerverrel. Ennyi erővel tárcsázhatsz manuálisan is, és indítsd a PHP-t hagyományos levelező kliensként.

Átmeneti

Ha notebookod van és különleges biztonságra van szükséged, beállíthatod a levelező klienst úgy, hogy a leveleket ne a notebookon tárold. A levelek akkor töltődnek le, amikor bekapcsolod a notebookot, és törlődnek, amikor bezárod a levelező klienst. Ha a notebooko-dat akkor lopják el, amikor ki van kapcsolva, a tolvaj üres bejövő postaládát talál, és semmit nem fog megtudni a Bahamákon levő titkos bankszámláidról.

Az átmeneti mód PHP-beli megfelelője a levelező szkript szerveren való elhelyezése, amikor a notebookot pusztán böngészőként használod. Csak arról győződj meg, hogy a böngésző üríti a cache-t, amikor egy böngészősession-t bezársz.

Gyors megoldások

A PHP levelező függvényeinek telepítése

A PHP-ban a **mailQ** és az **ezmlm_hash()** be van építve. Az IMAP-függvényeket telepíteni kell. Lehet, hogy a php.ini SMTP-beállítását is konfigurálnod kell.

Windows NT

Az IMAP Windows NT-ben való aktiválásához állítsd le az Apache-ot, másold a c:/Program Files/php/extensions/php_imap.dll-t a c:/winnt/system32/-be, távolítsd el a pontosvesszőt (;) a következő sor elől, és indítsd eí az Apache-ot:

```
;extension=php_imap.dll
```

Ha PHP-t CGI-ként futtatod, nem kell a webszervert újraindítani. A Windows 2000-ben ugyanúgy kell telepíteni, mint az NT-ben, a Windows 98 viszont némileg más könyvtárneveket használ.

Két sort kell a php.ini-ben megváltoztatnod ahhoz, hogy a **mail()** működjön. Nézd meg a php.ini következő két sorát:

```
SMTP = localhost
sendmail_from = me@localhost.com
```

Változtasd a **localhost-ot** a levelező szervered nevére. A **me@Iocalhost.com** helyére egy érvényes e-mailcímet írj, amelyet a kimenő levelekben alapértelmezetten használsz a **From címre.**

Unix

Az **IMAP** Unix alatti telepítéséhez be kell szerezned a C kliens-könyvtárat q washingtoni **egyetemről (ftp://ftp.cac.washington.edu/irnap/)**.

Másold a c-client/c-client.a-t egy könyvtárba, mint például a /usr/local/lib/. Másold a c-client/ rfc822.h-t és a mail.h-t az include-könyvtárba, például a /usr/local/include/-ba. A következő opcióval fordítsd be a PHP-t:

```
--with-imap
```

A mail() működéséhez lehet, hogy meg kell változtatnod egy sort. Keresd meg a következő sort a php.ini-ben, távolítsd el a pontosvesszőt (;), és változtasd meg a sort, hogy a te **sendmail-edre** mutasson. Tartalmazhat paramétereket is, az alapértelmezett a **sendmail -t -i:**

```
,-sendmail^ath =                                    ?,îMA
```

Levél küldése

Szórakozz a PHP-szkriptből küldött levelekkel, de először a saját címeddel teszteld a dolgot, mielőtt a gyanútlan barátaidat zaklatod az eredményekkel. A következő, „Egy levél küldése" című részben megmutatom, hogy minimálisan mi szükséges ahhoz, hogy levelet küldj, de ez kevés ahhoz, hogy gyakorlatias is legyél. A következő, „A **From** fejrész elküldése" című részben megmutatom azt, hogyan küldheted el a **From** fejrészt is, ami elengedhetetlen ahhoz, hogy a levél használható legyen. A „Több fejrész küldése" című rész azt mutatja meg, hogyan adhatsz hozzá több fejrészt, és itt kipróbálhatod, hogy mire van szükséged. Végül az „Egy üzenet küldése több címzettnek" című rész azt mutatja meg, hogyan tudsz egy levelet több embernek különböző megszólítással elküldeni.

Egy levél küldése

Amikor egy levelet küldesz el, csak arra címre van szükséged, amelyre azt küldeni akarod, egy tárgyra, az üzenetre és a mailQ-függvényre. A következő példa a PHP Sydney tagjainak szóló titkosított üzenetet mutat meg. (A valódi üzenetek a helyszínt és a dátumot, illetve a fel- és leiratkozási információkat is tartalmazzák.) Az újsorok (\n) új sorokat törnek a levélben. A felhasználó levelezőprogramja további töréseket szúrhat be a hosszú sorokba:

```
$to = "peter@a_web_site_somewhere.com";
$subject = "PHP Sydney meeting";
$message = "A reminder.\n"
    "The next meeting of PHP Sydney is Tuesday
    night.Xn" "The meeting starts at 7:00pm.\n" . "\n" .
    "Peter\n";
```

A következő kód a **mail()** használatával elküldi a levelet, és igazat ad vissza, ha a **mail()** képes volt kommunikálni a levéltovábbító ügynökkel (mail transport agent - MTA). Ha az MTA nem képes a levél elküldésére, attól a **mail()** még adhat vissza igazat, így az MTA hibanaplóját kell ellenőrizned, ha meg akarod tudni, hogy a leveled valójában elment-e. Amikor az MTA-t teszteled, a saját címedet beírhatod Cc vagy Bcc-ként (később elmagyarázom ezeket is):

```
if(mail($to, $subject, $message))

    print ("<brxf ont color=\ "green\ ">Mail sent to " . $to .

".</font>" ); else

    print ( "<brxf ont color= \ "red\ " >Mail f ailed . </f ont>" ) ;
```

Unixban a Sendmail a szokásos MTA. Vannak alternatívák, de neked nem kell tudnod az MTA nevét, mivel a folyamatot a rendszernek kell kezelnie. Az MTA-k több változata létezik a Windows NT és Windows *9x* alatt is. Annyi szoftver van a gépemre telepítve, beleért-

ve a POSIX-kiterjesztéseket is, hogy nem tudom, jelenleg mi látja el az MTA-szolgáltatást. A Win32 php.ini-ben a localhost az alapértelmezett, mely bizonyos rendszerbeállításokra működik, az alternatíváknak pedig egy levelező szervert kell a gépedre telepíteniük, vagy az SMPT-nek a levelező szervert tartalmazó szerverre kell mutatnia.

Néhány MTA egy fejrészt szúr be, amellyel azonosíthatod a forrásszoftvert, de amelyiket én használom, az nem. Itt az üzenet Netscape 4.77-es böngészőben, ha a View Headers (fejrészek megtekintése) Normal-ra van állítva:

```
Subject:   PHP Sydney meeting
   Date:   Sun, 01 Jul 2001 19:51:48 D1000
   From:   address@the_mail_server.com.au
     To:   peter@a_web_site_somewhere.com

A reminder.
The next meeting of PHP Sydney is Tuesday night.
The meeting starts at 7:OOpm.

Péter
```

Hogy néz ki a View Headers, amikor All-ra van állítva? A következőkben ugyanennek a levélnek az összes fejrészét láthatod. Ha a levél az egész világot bejárta, sok Receíved:-fej-rész lesz, minden szerverre egy. A Received-fejrészt használhatod a levelező szervered konfigurációjának ellenőrzésére és a kéretlen kereskedelmi levelek eredetének kiderítésére is. Sajnos csak a bénákat lehet így elkapni, mert a tapasztaltak hamis fejrészeket csinálnak (ahogy a levélben sem igaz semmi sem):

```
      Return-Path:      <address@the_mail_server.com>
    Delivered-To:      peter@a_web_site_somewhere.com
        Received:     from a800 (CPE-144-132.nsw.bigpond.net.au
         [144.190.50.132]} by a_web.com (Postfix) with SMTP id
         9257D17A08 for <peter@a_web_site_somewhere.com>; Sun, 1
         Jul 2001 19:51:32 +1000 (EST) Date: Sun, 01 Jul 2002
         19:51:48 D1000 From:     address@the_mail_server.com
         Subject:     PHP Sydney meeting
              To :     peter@a_web_site_somewhere.com
      Message-ID:      <20010701095132.92 57D17A08@a_web.com>
          Status:
 X-Mozilla-Status:      8001
X-Mozilla-Status2:      00000000
          X-UIDL:      3aad41fa00000403
```

A To mező több címet is tartalmazhat, és erre példát - Cc-vel - a megoldás későbbi részében találsz. Gyakran kapok kereskedőktől olyan e-maileket, amelyekben a teljes marketinges e-mailcímhstájuk egyetlen sztringként van betéve a To mezőbe. Vannak, akik nem tudják, hogyan kell e-mailt küldeni, és cégük egyik legértékesebb eszközét, a levelezési listát egyetlen e-mailben kiadják. Ezzel feldühíted a címzetteket, hiszen e-mailcímüket rengeteg más embernek kiadod. A több To címzett helyett használd inkább az „Egy üzenet küldése több címzettnek" részben bemutatott megközelítést.

A From fejrész elküldése

Talán észrevetted, hogy az előző szakaszban furcsa From-cím volt. Ha egy levélből kihagyod a From-címet, az egyik MTA be fog szúrni egy címet. Vannak olyan cégek, illetve honlap-üzemeltetők, akik jobban szeretik, ha az alkalmazottak kihagyják az egyéni **From**-címeket, és a levelező szerver szúrja be az általános címet. Más cégek az egyéni vagy az osztályhoz kapcsolódó címeket preferálják. Ebben a részben néhány módosítást hajtok végre az *előző* példán, hogy a From-címet belerakhasd a levélbe. Ehhez először egy From-címre van szükséged:

```
Sfrom = "everyone@thatusesphp.com.au";
```

A From-cím egy olyan paraméterbe van beszúrva, amelyet több levélfejrész kezelésére hoztam létre. A következő kód egy **$addtional** nevű mezőt állít be a fejrészekhez, majd hozzáadja a From-címet **($from);** amennyiben a $from-ban van adat, akkor a **"From:"** előtaggal adja hozzá:

```
$additional = "
if(isset(Sfrom) and
                strlen($from))

   $additional .= "From: " . Sfrom;
```

Változtass meg egy valamit az előző kódon: add a **$additional~t mailQ** paramétereihez, mint itt:

```
if(mail($to, Ssubject, Smessage, $additional)>
```

Amikor a levél megérkezik a From:-sor a From-címet fogja tartalmazni, mint itt:

```
From: everyone@thatusesphp.com.au
```

Az elrejtett fejrészek tartalmazzák a szerver címét, amelyet további fejrészekkel lehet az üzenetbe berakni, de a From-cím általában elegendő arra, hogy a címzett válaszolni tudjon:

```
Return-Path:    <address@the_mail_server.com>
     Sender:    address@the_jnai l_server . com
```

Két másik módja van a From-érték beállításának. A PHP php.ini-jében van egy opció az alapértelmezett From-cím beállítására, ami jó módja az alapértelmezés beállításnak, ha az oldalad más oldalakkal osztozik egy MTA-n. A php.ini-t felülírhatod a .htaccess-fájlokkal, ha az oldalad egy virtuális oldal a sok között.

Volt egy rendszer, amelyen egy másik alternatívát használtam egy darabig, mert a **From:** hozzáadása a további fejrészekhez nem működött megbízhatóan. A php.ini alapértelmezett beállításait felül lehet írni ini_set()-tel, ahogy azt itt láthatod. Amikor ezt az opciót a jelenlegi konfigurációmon teszteltem, az **ini_set** szintaktikai hibát jelzett:

```
        'sendmail_from", $from);
mi set
```

Több fejrész küldése

Előfordul, hogy további fejrészeket kell használnod. A Cc több e-mailcímet fogad el, és másolatot küld minden címre. A To-címzett és minden Cc-címzett látja a To-címet és a Cc-címeket. A Cc, akárcsak némely másik fejrész, megkülönbözteti a kis- és nagybetűket. A további fejrészek listáját ennek a megoldásnak a végén találod.

A következő címeket használom a példakódban. A $cc egy több Cc-címet tartalmazó tömb, ennek segítségével mutatom meg a többszörös címzés technikáját. A $cc tartalmaz egy üres sztringet is, mert az problémát okozhat:

```
$cc[]  = "info@some_other_site.com";
$cc[]  = "";
$cc[]  = "info@yet_another_test.com";
$from  = "everyone@thatusesphp.com.au";
```

Ahhoz, hogy a program több fejrészt vegyen figyelembe, a fejlécek egy $add nevű tömbbe kerülnek, és a levelezési függvény számára hozzáadom a $add-et a $additional sztringhez. A From-címet hozzáadom a $add-tömbhöz, amennyiben a $from-ban van adat, akkor a "From:"-előtaggal:

```
if (isset($frora)      and
    strlen($from)) {$add[]      .=
    "From:     "     .     $from;
```

A következő kód ellenőrzi, hogy a $cc tömb-e, végigfut rajta, és ha talál benne nulla hosszúságú szrtringet, azt eltávolítja. Majd az implode használatával az összes elemet a $c sztringbe egyesíti. Ha az olyan mezőkben, mint a Cc vagy a Bcc, több elemet használsz, az elemeket vessző-szóközzel kell elválasztani, így az implode egy vessző-szóközt tartalmaz. Ha a $cc nem tömb, akkor a kód a $cc-t egy egyetlen elemet tartalmazó sztringként kezeli, és az egyetlen elemet adja hozzá. Fontos az input-címek tisztán tartása, mert egy címben egyetlen hiba elég ahhoz, hogy az üzenet ne érjen el mindenkit. Ha gondod van a listával, próbáld meg a levelet a címzetteknek egyenként elküldeni:

```
reset($cc);
if(isset($cc)) {
    if(is_array($cc) ) {
        while(list($k,     $v)     =
             each($cc)) {
            if(!strlen{$v)) {
                unset($cc[$k]);


        $c    =    implode(",     ",
        $cc);  if(strlen(Sc))
             í
             $add[]    =    "Cc:    "    .    $c;
```

```
elseif(strlen($cc)}
    $add[] = "Cc: " . $cc;
```

Bármennyi további fejrészt hozzáadhatsz az egyetlen címet tartalmazó From-, vagy a több címet tartalmazó Cc-kód bemásolásával. A következő lépés a $add-tömbben levő összes szöveg összerakása egy hosszú, fejrészeket tartalmazó $additional nevű sztringbe. A következő kód ellenőrzi, hogy a $add érvényes tömb-e, és a $separator-ban levő érték használatával összefűzi a $add-ben levő szöveget. A $separator Windows és Windows NT alatt \r\n-re van állítva, Unix alatt pedig \n-re. Vedd észre, hogy az elválasztó is hozzá van adva a fejrészek sztríngjének végéhez. Úgy találtam, hogy ha a $additional bármilyen elvá-lasztót tartalmaz, akkor a $additional végére is kell egy elválasztó. Amikor a $additional csak egy fejrészt tartalmaz, az elválasztót hozzá is adhatod és el is hagyhatod; a levelező szoftver nem törődik vele, hogy az elválasztó \n vagy \r\n vagy egyáltalán nincs ott:

```
Sadditional  =   "";
Sseparator   =  "\r\n";
if(isset(Sadd)    and          $add)
is_array
    $additional = implode(Sseparator, $add) . Sseparator;
```

Vedd észre, hogy a levél törzse egy üres sorral van a fejrészektől elválasztva. Ha egy üres sor becsúszik a fejrészek közé, a sor utáni fejrész az e-mail törzsében fog megjelenni. Végezz el minden szükséges ellenőrzést annak megállapítására, hogy a kód biztosan nem ad nulla hosszúságú vagy üres elemeket a fejrészekhez. Végső biztonsági ellenőrzésként használd a következő kódot, mielőtt a Sadd-et a $additional-be szerkeszted:

```
reset($add);
whiledist ($k,    $v)    =    eacíi ( $add) )
    {                                                    i
    if(!strlen($v))                                      {
        {
        unset ($add[$k] ) ;                              <
```

A mailQ körül levő kódot nem kell megváltoztatni. Ha a $additional üres, attól még a mail() elküldi a levelet.

Egy üzenet küldése több címzettnek

Egy üzenet több címzettnek való küldésének legjobb módja, ha minden egyes embernek küldesz egy levelet, így nem látják saját címüket egy óriási To-listán nyilvánosságra hozva. Az egyénenként elküldött leveleket testre is szabhatod. A következő kód mindenkinek küld egy levelet, aki rajta van a listán.

A lista bárhol lehet: fájlban, adatbázisban vagy űrlapon. Annak érdekében, hogy a kód bármilyen forrásból származó listával működjön, a következő kód egy $mail_list-et hoz létre. Ezt egy tömbre konvertálja, és az elküldő függvényt egy ciklusba helyezi, hogy a tömb adatait beletáplálja. Bizonyos lépéseket a saját kódoddal helyettesíthetsz, ha saját forrásból hozod létre a tömböt. A listában soronként egy elem van, amely egy nevet és egy e-mailcímet tartalmaz pipakarakterrel elválasztva (|). A kód a nagymennyiségű szöveg PHP-kódba való juttatásának egy egyszerű módját használja. Az endoflist-et (lista vége) bármilyen szöveges sztringgel helyettesítheted, ami nem fordul elő az adatokban:

```
$mail_líst = <<<endoflist
John S
BrownIjsb@somewhere_or_another.com Péter
Ipeter@a_web_site_somewhere.com endoflist;
```


A következő kód a $mai]_list-listát egy $to-tömbre konvertálja, amely olyan szerkezetű, hogy a levélküldést, illetve a tömb adatbázisból való létrehozását egyaránt könnyűvé teszi:

```
if(isset($to)) {
    unset($to)
    >                                  '  •     s ; ■.     .-       ■      -  ;"n/t/P    - '• '
$x = explode("\n", $mail_list);
whiledist ($k, $v) = each($x)) {
    list($name, Saddress) = explode("I", $v);
    $to [] = array("address" => $address, "name" => $name);
```

Az explode() fogja a listát, és egy sorokból álló $x nevű tömbbe bontja le, ciklussal végigfut a tömbön, és létrehozza a végsó', $to-tömböt. A $x minden sora a | jelnél van explode által elválasztva, az első rész a Sname-be, a második pedig a Saddress-be kerül. A két mező a Sto-ban egy elembe kerül. (Csak egy kis emlékeztető a tömbökről: Ha a $to korábban sztringként van definiálva, a PHP reklamálni fog, amikor megpróbálod a $to-t tömbre konvertálni. Ha a $to már tömb, a kód az elemeket a már létező elemek után rakja be. Mindkét problémát elkerülendő, beraktam az unset($to)-t.)

A $mail_list-elemek lehetnek a Péter <peter@a_web_site_somewhere.com> kiterjesztett cím formátumban. Használd a kiterjesztett címeket levelezési címként, és vágd le az első részt névként. Úgy találtam, hogy bizonyos MTA-k nem fogadják el a kiterjesztett címeket, vagy csak a kiterjesztett címek korlátozott számú változatát fogadják el, így tartsd a címeket a rövid formában mindaddig, míg nem működik minden.

A fejrészt feldolgozó kód ugyanaz, mint az *előző* részben. A postázó kód változik, ahogy azt a következő kódban láthatod, hogy a Sto-n ciklussal végigfutva minden elemhez külön levelet hozzon létre. A whileQ-ciklus végigfut a $to-n, és az elemeket a $v-be adja vissza, amely a címzett nevét és címét tartalmazza. A $message-dzsel kezdődő sor a megszólítást teszi az üzenet elejére, jelen esetben a Dear John,-t. Az if(mail())-sorban a címet a $v["address"]-re cseréltem, csakúgy, mint a két print-sorban:

```
whileflist($k, $v) = each($to))
    {
    Smessage = "Dear " . $v["name"] . ",\n" . Smessage;
    if(maii($v["address"], $subject, $message, $additional))

        print ( "<brxf ont coior=\ "green\ " >Mail sent to "      $v [ "address "]
              ".</font>" );

    else

        print { "<brxf ont color=\"red\">Mail failed to .      . Sv [ "address "]
            ".</font>" ) ;
```

Ha a levél a címeket a kiterjesztett formátumban tartalmazza az < és > jelek között, akkor a print-utasítások nem fognak működni, mivel az output egy része HTML-ként lesz értelmezve. Ezt a megjelenítési problémát megelőzendő, tedd a **htmlentities()-t** a cím köré, ahogy itt látod. Az összes megjelenítendő levélmező köré rakj **htmlentities()-t,** hogy ezt a problémát elkerüld. Ha megjeleníted a levél törzsét, és meg akarod őrizni a sortöréseket, használd az nl2br(htmlentities())-t, mert az **nl2br()** minden egyes újsorhoz
-t ad:

```
print ( "<brxf ont     color=\ "green\ " >Mail     sent     to     "
            .     htmlentities($v["address " ] )       .     ".</font>"      ) ;
```

Amikor nagy leveleket küldesz többször, vagy kis leveleket nagyon sokszor, előfordulhat időtúllépés. A **set_time_limit()** használatával állítsd át a PHP időkorlátját 30 másodpercről a szerinted elegendő időtartamra, ahogy azt a következő kód mutatja. Az időkorlátot 0-ra állítva megengedheted, hogy a szkript a végtelenségig fusson, de ez nagyon veszélyes, ha a szkriptben véletlenül van egy végtelen ciklus. A legbiztosabb megközelítés az, ha másodpercenként egy bizonyos számú levéllel számolsz, a példában 2-vel, és azt megszorzod az üzenetek számával:

```
set_time_limit(count($to)     2)
```

Most már mindenhez van kódod, kivéve a csatolt állományok küldést. A levelezőlisták követelményeinek is már a 99,8 százalékát tudod kezelni.

Levél küldése csatolt állománnyal

Ez a megoldás a levélküldés korábbi szakaszokban nem tárgyalt 0,2 százalékával foglalkozik. Ez a legösszetettebb rész, amelyben vagy nagyon megbízol, vagy légy felkészülve arra, hogy rengeteg időt fogsz a kísérletezgetéssel eltölteni. Ez az, ahol a MIME-t hozzáadod a levelezéshez és ahol csatolt állományokat teszel abba. Ha egyszer belefogsz, a levelezésed addig nem fog működni, amíg minden tökéletesen nem klappol. Ezért kell a levelezést először csatolási lehetőség nélkül kialakítanod. Nézd át alaposan az *előző* megoldásokat, és amikor mindenféle fejrésszel tudsz már levelet küldeni, akkor állsz készen arra, hogy a MIME-vel kísérletezz.

Tesztadatok kiválasztása

A következő lista egyetlen levél tesztüzenete. A tesztüzenet két tesztfájllal lesz párosítva, amelyek a csatolt állományként gyakran előforduló két fájltípust reprezentálják: a szokatlan karaktereket tartalmazó szöveges fájlokat és a képfájlokat. A következő legnépszerűbb csoport a futtatható fájlok, de az ilyen fájlokban terjeszkedő vírusok miatt ezekben egyre kevésbé bíznak meg a felhasználók. Amikor a MIME-kódodat teszteled, egy fájltípussal kezdj, majd tesztenként add hozzá a következő típust, míg az összes olyan fájlt le nem teszteled, amelyet használni akarsz.

Vedd észre, hogy az első definíció a kocsivissza-újsor elválasztó sztring, az \r\n. Az *előző* levelezési megoldásban csak a küldő program követelményei miatt kellett aggódnod, mivel az Interneten minden más egyetlen \n-nel működött. A MIME-üzenetek nem működnek, ha egyetlen \n-t vagy Macintosh-os változatát, az \n\r-t használod. Némely üzenetek átmennek, de nem mind. Ez a közted és a címzett között levő összes szerver levelező szoftverén múlik:

```
$separator = "\r\n";
Sfrom = "everyone@thatusesphp.com.au";
Sto = "peter@a_web_site_somewhere.com";
$salutation = "Péter";
$subject = "Test mail attachments";
$message = "The enclosed files are tests." . $separator;
```

Ezután tesztfájlokra van szükséged. Találtam egy nagyon kicsi GIF-fájlt, és létrehoztam egy néhány mondatot tartalmazó rövid szöveget, amelyben olyan különleges karakterek vannak, mint a ". A fájlokat az üzenethez fogom csatolni, így a későbbi egyszerűbb feldolgozás érdekében a neveket egy tömbbe rakom. A kód az egyes fájlokhoz tartozó tömbelemhez további attribútumokat fog hozzáadni, így a tömbelemek is tömbök, amelyekben a fájl **nevéhez tartozó elem** neve **["file"]:**

```
$attachments[]["file"] =
"./backspace.gif"; Sattachments[]["file"] =
"./test.txt";
```

Fájlinformációk gyűjtése

A kód az egyszerű tesztelés és módosítás érdekében apró részekre van bontva. Ha a kód már működik, a feldolgozási többletráfordítások csökkentésére közelebb hozhatod egymáshoz a részeket, de a legnagyobb ráfordítás maga a levél továbbítása lesz. A következő kód egyszerűen ciklussal végigfut a csatolt állományok tömbjén **(Sattachments),** elválasztva a fájlnevet ["name"] a teljes elérési úttól **["file"]. A ./backspace.gif-ből backspace.gif,** a **./test.txt-ből test.txt** lesz:

```
reset ($attachments) ,-
while (üst ($k, $v) = each ( Sattachments ))
    í
    Sattachments[$k]["name"] = basename($v["file"]);
```

Amikor az egyes fájlokat beolvasod, szükséged lesz a fájl méretére, ezért használd a következő kódot a fájlméret kiderítésére és a tömbhöz való hozzáadására. Ennél a kódnál derül ki először, ha a fájl hiányzik, ezért add hozzá ehhez a részhez a szokásos hibaüzenet-küldő funkciókat is.

```
reset(Sattachments);
while(list($k, $v) = each($attachments))
    {
    $attachments[$k]["size"] = filesize($v["file"]);
```

Ha űrlapot szeretnél létrehozni, hogy az emberek kiválasszák a csatolandó fájlokat, az űrlapokat a 9., a fájlfüggvényeket pedig a 8. fejezetben találod.

il

Hivatkozás:	oldal:
Fájlok feltöltése	**284**

A MIME-feldolgozás az üzenethez csatolt állomány típusától függ. A következő kód a fájltípus meghatározásának gyors és megbízhatatlan módja - a kód függ a fájlvégződésektől, de nem minden fájl illetve operációs rendszer használ végződéseket, és vannak olyan fájltípusok, amelyek többet is használnak:

```
reset(Sattachments);
while(list($k, $v) = each(Sattachments)) {
    switch(substr (strrchr ( $v [ "f ile"] , "."), D) {
        case "gif":
            Sattachments[$k]["type"] = "image/gif; name=\""
            . Sattachments[$k]["name"] . "\"»; break;
        case "txt":
        Sattachments[$k]["type"] = "text/plain; charset=iso-8859-l"; break;
        Default:
        Sattachments[$k] ["type"] = "application/octet-stream" ; }
    Sattachments[$k]["encoding"] = "base64";
```

A fájlfüggvények, illetve képek esetében a 11. fejezetben bemutatott képfüggvények segítségével további ellenőrzéseket végezhetsz. Megengedhető az is, hogy ha a fájl típusa nem szokványos, akkor a felhasználó gépelje azt be, vagy egy legördülő listából válassza ki. A MIME-típusnak pontosnak kell lennie, ha azt akarod, hogy a fogadó meg tudja nyitni a böngészőjében a fájlt, mert a böngészőnek tudnia kell, hogy milyen fájlnéző programot indítson. Amikor a fogadó csak lemezre menti a fájlt, a MIME-típus bármilyen általános érték lehet, amely segít a levelező böngészőnek a szöveges és bináris mód közül a megfelelő kiválasztásában.

A kód ciklussal végigfut a csatolt állományok tömbjén, és a substr() és a strrchar() használatával megragadja a fájlnévbó'l a fájlvégződést (ami az utolsó pont utáni minden). A switch()-függvény a fájlvégződés alapján kiválaszt egy műveletet, és a műveletek a MIME-fejrészekben való használathoz beállítják a fájltípust ["type"]. Van egy alapértelmezett művelet, a application/octet-stream, amit ebben a példában nem teszteltem. MIME-típusok tucatjával fogsz találkozni, és a legegyszerűbben úgy tudod meg, hogy milyen fejrész kell hozzájuk, ha küldesz magadnak levelet a csatolt fájlokkal. A Netscape levelezőjében használd a View Page Source-t a fejrészek megtekintésére.

Hivatkozás:	oldal:
Sztringfüggvények	57
Képinformációk gyűjtése	384

MIME-fejrészek létrehozása

A MIMÉ további fejrészeket igényel az üzenet elejére és minden egyes csatolt állomány elé, illetve egy határoló (boundary) jelölést azok közé. Ezek a fejrészek a $add-tömbbe kerülnek, amelyet az előző megoldásokban fejrészekre használtam. Az előző kódot ebben a megoldásban is használhatod. Bizonyos fejrészek a $message_prefix-be kerülnek, amelyek hozzáfűzhetek az üzenet törzsének elejéhez, a többi MIME-fejrész pedig a $message_suffix-be kerül, a kódolt csatolt állományokkal együtt. A fejrészeknek és a csatolt állományoknak a $message-től különálló kezelésének az az oka, hogy ha a kódot levelezőlistán használod, az egyes címzetteknek testre szabott üzenetet küldhess:

```
$message_prefix = "";
$message_suff ix = "" ;
```

Lehet, hogy a rendszered olyan, hogy a szkript eleje és a csatolandó állományok beszúrása között sok kód van, így a következő kóddal ellenőrizheted, hogy a csatolandó állományok tömbje létezik-e, tömb-e és vannak-e feldolgozandó elemei:

```
if(isset($attachments)    and              ($attachments)
   is_array and  count($attachments)
   >   0)
```

A kód ezután határoló mezőt hoz létre, ahogy itt láthatod. A határoló mező egyedi sztringet igényel, amely elválasztja a kódolt állományokat. A példában elvégzett összes kódolás base64 típusú. Ez nem tartalmaz kötőjeleket, így azok működni fognak. Rászántam az időt, hogy a határolót egy kicsit egyedibbé tegyem, és elkerüljem azt a szituációt, amikor valaki egy fejrészt kötőjelekkel húz alá, ami gyakori a szöveges csatolt állományoknál. A base64 kódolást - egy kis többletráfordítással - nyugodtan használhatod bármire, és így elkerülheted azt, hogy egyedi határoló sztringet kelljen generálnod:

```
   list(Sx, Sy) = explode("     microtime());
   $boundary = "boundary--"    $y . substr($x,  2)
```

Két MIME-féjrészt adtam az üzenethez, jelezve a levelezőrendszernek, hogy az üzenet MIME-kódolást tartalmaz. Ha MIME-t adsz hozzá, akkor az egész üzenetnek alkalmaz-kodnia kell a MIME-szabványhoz, és bármilyen hiba tönkreteheti az egész üzenetet:

```
$add[] = "MIME-Type: 1.0";
$add[] = "Content-Type: multipart/mixed; boundary=\"" . $boundary
```

A határoló sztring két kötőjellel a végén egyszer van használva. Minden további használat-hoz két, az elejéhez adott kötőjelre van szükség, így add hozzá ezeket:

```
$boundary = "--" . $boundary;
```

Az üzenettörzsbe MIME-fejrész kell annak jelzésére, hogy szöveg van benne. A következő kód a megfelelő fejrészeket adja hozzá, és nem nagyon romolhat el, ha csak nem speciális karaktereket tartalmaz (ha nincs szükséged speciális karakterekre, változtasd a **Content-Transfer-Encoding-ot 8bit-re,** a **charset-et** pedig **iso-8859-l-re):**

```
$message_prefix .= Sboundary . Sseparator
    . "Content-Type: text/plain; charset=us-ascii" .
    $separator . "Content-Transfer-Encoding: 7bit" .
    $separator . $separator;
```

Vedd észre, hogy elválasztó van a fejrészek végén, mert ez jelzi a böngészőnek, hogy a fej-rész hol végződik, és hol kezdődik az üzenettörzs. Ha kihagyod a második fejrészt, a szö-veg nem fog megjelenni. Ha azt gyanítod, hogy egy levelező szkript ilyen hibát tartalmaz, tegyél egy dupla elválasztót az üzenet közepére, és figyeld meg, hogy eltűnik-e a szöveg pont a dupla elválasztó után.

Ha az üzenet nem elválasztóval végződik, tegyél be egyet, hogy az ne keveredjen a csatolt állományok fejrészeivel. Hozzáadtam az extra elválasztót az üzenet végződéshez, így az üzenet nem fog megváltozni. Ha testre szabott üzenetet hozol létre, ezt az ellenőrzést tedd az üzenet végére, és tedd az elválasztót oda:

```
if(substr($message, -1) !=
    $separator) {$message_suffix .=
    $separator;
```

MIME-üzenetrészek létrehozása

A kód hátralevő része a csatolt állományok listáján fut végig. Ha bármilyen kód van a tömb létrehozása és használata között, állítsd a tömböt a reset()-tel az elejére:

```
reset(Sattachments);
while(list($k, $v) = each(Sattachments))
```

A következő kód egy csatolt állomány elé rakja a fejrészeket. A fejrészek a $message_suffix-ba kerülnek, és határoló sorral kezdődnek, hogy jelezzék az új állományt. A **Content-Type** mindenképpen szükséges, és a tömb típusát használja. A **Content-**

Transfer-Encoding szintén kötelező, itt a példában minden base64-gyel kódolt. Némely adat kódolásának van egy kicsit hatékonyabb módja is, de aggodalomra nincs ok. A base64 kódolás egy 12MB-os fájlt 16MB-ra kódol, míg más kódolási technikák csak 14MB-ot vagy éppen 17MB-ot használnak fel. Csak akkor keress magadnak másik kódolási technikát, ha gyakran küldesz egy meghatározott típusú nagy fájlt, illetve meggyőződtél róla, hogy a címzett böngészője ki tudja kódolni a speciális kódolásod. A **Content-Disposition** fejrész Netscape 4.77-ben nem okozott változást. Próbálkozzál bátran, és kísérletezd ki, hogy melyik levelezési programok értik meg a fejrészeidet:

```
Smessage_suffix .= $boundary . $separator
. "Content-Type: " . $v["type"] . $separator
. "Content-Transfer-Encoding: base64" . $separator
. "Content-Disposition: inline; filename=\"" .
$v["name"]
. " \"" . Sseparator . Sseparator,-
```

Itt az idő, hogy beolvasd a csatolt állományt. Nyisd meg a fájlt olvasási módban, olvasd be egyszerre a korábban kiderített méret használatával, majd zárd be. Windows és Windows NT alatt add a bináris attribútumot a fájl beolvasáshoz (használj **"rb"-t** az "r" helyett), különben elveszítheted az olyan bináris fájlok egyes részeit, mint a képek. A végén nem fogLC a hibát látni, csupán a címzett kaphat furcsa eredményt, például csak félig megjelenő fájlokat. Az esetleges hibákat a fogadott fájlt elmentve és annak hosszúságát az eredeti fájléval összevetve ellenőrizheted. A bináris beolvasás problémája rövid fájlokat eredményez:

```
$f = fopen($v["file"], "rb"); $x
= fread($f, $v["size"]);
fclose($f);
```

A fájlt tartalmazó sztringnek base64 kódolásúnak kell lenni a **base64_encode()** használatával, majd ahogy itt is látod, bontsd azt a chunk_split()-tel kezelhető sorhosszúságúra. A **chunk_split()** minden 76. karakterhez \r\n-t szúr, és az opcionális paramétereivel beállítható más karakterszám és elválasztójel, ha másmilyen elválasztásra akarod használni:

```
$message_suffix .= chunk_split(base64_encode($x));
```

Miután az összes állományt csatoltad, szúrj be még egy határoló sort. Ez a sor egy kicsit más: van két további kötőjel a végén. Ez a MIME-kódolású levél továbbításánál (forward) fontos, mert ekkor a levelező szoftvernek a MIME-határolók egy halmazát egy másik halmazba **kell** beágyaznia. A határolók egymásba ágyazása egyedi határolóértéket igényel, és itt lesz hasznos a határoló sor microtime része:

```
$message_suffix .= $boundary Sseparator;
```

Nem MIME-f ej részek létrehozása

A következő kód hasonló az előző megoldásokban használtakhoz, annyi változtatással, hogy egy To-címet és szabványos elválasztó sztringet használ:

```
;m
```

```
if (isset($from) and strlen($from)) {
    $add[] = "From: " . Sfrom;    ,-.
    ,                              'Sfiimíí'
Sadditional   =   " ■■;
if (isset (Sadd)    and    is_array (Sadd) )
    {
    $additional = implode(Sseparator, $add)
                                                    .
                                                $separator;
```

A levél elküldése

Most már elküldheted a levelet. A következő kód a elő- és utótag mezőkkel egészíti ki és elküldi az üzenetet. Ebben az üzenetben csupán a megszólítás testre szabott. Előfordulhat, hogy további testre szabott információt akarsz az üzenetben megjeleníteni, például ha Word-dokumentumot vagy futtatható fájlt küldesz csatolt állományként. Kezd a szkriptet a fájlok vírusellenőrzésével, majd az eredményeket testre szabott üzenetként tedd az egyes csatolt állományhoz:

```
                                                    \n"
$m = $message_prefix . "Dear " . Ssalutation . "
    . $message . $message_suffix;
if(mail($to, $subject, $m, Sadditional))
    {
    print ("<brxf ont color=\ "greenA ">Mail sent to .
        htmlentities(Sto) . ".</font>" );

else

    print ("<brxf ont color=\ "red\ " >Mail failed to .
        htmlentities($to) . ".</font>" );
```

Bármit elküldhetsz levélben. A világ rád vár, de csak apró lépésekben haladj. A MIME-levelezés nem tudja a hibákat saját maga kijavítani - a legkisebb hiba az egész levelezést leállíthatja, és órákon át törheted a fejed a megoldáson. Az eredményeid a levél útjának első és utolsó elemének függvényében változnak. A levélküldő programodnak, amely a PHP-függvényektől kapja a leveleket, számtalan olyan sajátossága lehet, ami leállíthatja a levelet. Az utolsó elem a fogadó levelező böngészője, és lehet, hogy az nem támogatja a MIMÉ egyes célratörőbb sajátosságait, ezért mindent tesztelj le először. Kérd meg a címzettet, hogy küldje neked vissza a megkapott levelet, hogy a csatolt állományokat elmentsd a lemezedre és összehasonlítsd az eredeti fájlokkal.

Levelezési címek ellenőrzése

Ha e-mailcímet űrlapon keresztül is elfogadsz, valószínűleg még az idő alatt ellenőrizni akarod a helyességét, amíg a látogató a gépe előtt ül. Ennek egyetlen megbízható módja egy visszaigazoló levél küldése. Kérd meg a látogatót, hogy nézze meg, megkapta-e a visszaigazoló leveledet.

Mi van, ha a felhasználó nem várhat? Számtalan javaslat van arra, hogyan használj például reguláris kifejezéseket az e-mailcímek ellenőrzésére, de ezek egyike sem működik. A levelezési címek annyira rugalmasak, hogy semmilyen használható eszközzel nem ellenőrizheted a teljes címet; csak részekben tudod ellenőrizni, és minden rész, amit ellenőrzői, megváltozhat.

Az ellenőrzési javaslatok tesztelésére a következő listát állítottam össze, amely jó és rossz címeket egyaránt tartalmaz. A levelezőlista-kezelőktől, például leiratkozási címekről érkező címek sokkal bonyolultabbak ezeknél, de szerencsére azokat nem szokták egy lekérdezési űrlapba beírni. A .com-, .net- és .org-kiterjesztések hamarosan hét társat kapnak, és így egy a .com-ot ellenőrző rutinnak az összes többi lehetőséget végig kell néznie. A legalább 239-féle országkódot szintén ellenőrizni kell:

```
$e[] = -■;
$e[] = "peter";
$e[] = "peter@",-
$e[] = "peter@petermoulding";
$e[] = "peter@petermoulding.";
$e[] = "peter(|petermoulding.com";
$e[] = "x@y.z";
$e[] = "x@y.z.au";
$e[] = "Péter Moulding <peter@petermoulding.com>";
```

A következő két reguláris kifejezést e-mailcímek ellenőrzésére javasolják. Az első egy hírcsoporton jelent meg valahol. A második ennek egy módosítása, amit én kísérletezgettem ki. (A PHP-ban két típusa van a reguláris kifejezéseknek: a POSIX és a Perl stílusú.) Különböző hírcsoportokban ennek a kifejezésnek számtalan, csak egy vagy két karakterben különböző változatával találkoztam:

```
$f[]    =
'iyv[_a-z0-9-] $f []   =      .[_a-zO-9-]+)*@[a-zO-9-]+(\.[a-zO-9
11/v[_a-z0-9-]                .[_a-zO-9-]+)*@[a-zO-9-]+(\. [a-zO-9
```

A 15.1 ábra mutatja a két reguláris kifejezéssel a tesztcímeken elvégzett próba eredményét. Vedd észre, hogy a második változat a jobb, de még nem tökéletes, és egyik sem keres domaintípust vagy országkódot.

A következő kód végezte el az összehasonlítást. Használd ezt a reguláris kifejezésekkel való kísérletezgetésre, és ne csüggedj, ha az eredmény nem működik megbízhatóan:

```
print("<table border=\"3\"><tr>"
    . "<tdxfont color=\ "Green\ "><em>Eregi pattern</emx/fontx/td>"
    . "<td> ,-</tdx/tr>" ); while(list($k, $fv) = each($f))
!wiwrj..vs;öBJiuK»*Kirj.\<i«
-MU*. print ("<trxtdxfont color=\ "Green\ " >" . $fv . "</f ontx/td>"
■:     . "<td> </tdx/trxtrxtdxem/em>Email address</emx/td>"
»,-     . "<tdxem>Result</emx/tdx/tr>" ) ;
"reset($e) ;                                                    n
   whiledist ($k, $ev) = each(Se))

       print ("<trxtd>" .  htmlentities{$ev)              )
       if(eregi($fv, $ev)
```

```
        í
        print("<td>Ok<td>"      );
    else
        print ("<tdxf ont color = \ "Red\ ">No</fontx/td>" )
    print("</tr>"
print("</table>" );
```

öíi A .ÖJ5ríxntifl3Ü5

Eregi pattern .	
r a.z0-9-]+(V[_a-z0-9-]+)*@[a-20-9-]+(\.[a-z0-9-]+)*$]	
Email address	*Resui*
	No
peter	No
peter@	No
peter@petermoulding	Ok
peterppetermouíding.	No
peter@petermoulding.com	Ok
x@y.z .	Ok
x@y.z.au	Ok
Peter Moulding <peter@petermoulding.com>	No
[a-z0-9-]-Kl[_a-z0-9-]+)*@[a-z0-9-]+(V[a-z0-9-]+)+$j	
Email address	*Resui*
|NO ;	
jpeter | No	
|peter@	
peter@petermoulding **No**	
peter@petermoulding. :|No	
peter@petermoulding.com ~|Ök~	
s@y.z **Ok**	
x@y.z.au **Ok**	
iPeter Moulding <peter@petermoulding.coni> 1JNÖ	

15.1 ábra Reguláris kifejezésekkei leellenőrzött e-mailcímek

Hogyan léphetsz az ellenőrzésben eggyel előbbre? Vágd szét a problémát kezelhető darabokra. Először szúrd be a következő kódot, hogy a tesztcímek listáján ciklussal végigfusson. Hirtelen egy kisebb probléma jelentkezik. Ha a $e2-ben kevesebb mint két elem van, az e-mailcím nem lehet teljes. Ha a $e2-ben több mint két elem van, a cím érvénytelen:

```
$e2 = explode("@", $ev);
```

Most már két különálló problémaként ellenőrizheted a $e2 érvényességét. A @ jel bal oldalán levő cím, a felhasználói név csinos reguláris kifejezésekkel ellenőrizhető, vagy teszteld

le, hogy nincs-e benne érvénytelen karakter. A nem angol e-mailcímek megjelenése miatt gyakorlatilag semmi sem tekinthető érvénytelennek. A cím jobb oldali része, a domain egy vagy több explode-dal ellenőrizhető, mint itt:

```
Sd = explode( ".", $e2[1]);
```

A $d részenként tartalmazza a domainnevet. Ha a $d utolsó része, a legfelső elem két karakter hosszú, országkódként ellenőrizhető. A következő elem, vagy ha az utolsó nem országkód, akkor az utolsó elem domaintípusként ellenőrizhető, és ha érvénytelen, akkor hibát jelez. Némely országnak saját domaintípusai vannak, így ezekre az országokra a kódot meg kell változtatni.

Ha már ellenőrizted az országkódot és típusát, rákereshetsz a domainre, hogy meghatározd, létezik-e. Használható az a trükk is, hogy a 13. fejezetben mutatott függvények közül használsz egyet az oldal vagy az oldal fejrészének lekérésére, hogy a domainből megállapíthasd, hogy aktív-e. Idáig már annyi munkát végeztél, hogy beláthatod, egy visszaigazoló le-\é\ küldése sokkal egyszerűbb és megbízhatóbb. Ha belebonyolódsz a reguláris kifejezés eredményébe, annak akár az is lehet a vége, hogy úgy engedsz el egy potenciális vásárlót, hogy még az érvényes e-mailcímét sem tudod. Majdnem minden harmadik e-mailcímet hibásan gépelnek be, és kevesen térnek vissza egy oldalra, ha nem kapják meg az e-mailt. Ragadd meg a címüket, küldj nekik üzenetet, és kérd meg őket, hogy ellenőrizzék, hogy az üzenet megérkezett-e.

Hivatkozás:

Reguláris kifejezések **21**

<U

.:-i-.ütB^ aháJugeR

ói í, ad *htiltt*
tó*

,fU..

,-. -fc

16. fejezet

Hálózatok

'jjst ?i.:i! ínért í

jJSfC ffOl! ÍI£SSni,7L!i

Gyors megoldások

Aoí

üArjvoÁ A .I

Áttekintés

Tudod mi az az *IPcím, host-név és DNS}* Ha tudod, vagy szeretnéd megtudni, akkor készen állsz a hálózat állapotát felismerő szkriptek írására és az Internet nyújtotta /'- lehetőségek mélyreható kiaknázására. Ez a fejezet azokat a hálózattal kapcsolatos utasításokat mutatja be, amelyeket nem használtunk más fejezetekben. Sokuk új, kísérleti stádiumban van, és egyáltalán nem ajánlatos élesben működő oldalakat kiszolgáló szerveren használni őket.

A PHP-ben vannak utasítások a Domain Name Service rekordok (DNS, Domain név szol-■_ gáltatás) visszanyerésére, vagyis kereshetsz az elérhető domainek, valamint Web Ditributed Data Exchange-elemek (WDDX, Weben keresztüli szétszórt adatcsere) között, tehát infor-mációt cserélhetsz távoli szerverekkel. A PHP-hez most adnak egy hibakeresőt is, ami kü-— lönösen jól jön, ha a hálózati kapcsolatok problémáit próbálod feltárni.

Mivel a hálózati szkriptek hosszú ideig futhatnak, a rendszer- és magánnaplózást szintén tárgyaljuk. Egy szkript, amely a világ különböző pontjain elhelyezett tartalomszerverekről gyűjt cikkeket, 10 percig is futhat, és óránként újraindítható. Nem lehetsz ott minden tíz-perces időszakban, vagyis a hibák (vagy a siker) naplózása elkerülhetetlen a további fejlesz-téshez és a problémák meghatározásához.

A hibakereső

A PHP hibakeresője még fejlesztés alatt áll, ezért amíg a hibakereső elkészül naplózásra és egyéb technikákra lesz szükséged kódod követéséhez. Számos különböző hibakeresőt használok, és mindegyik jelentősen igénybe veszi az erőforrásokat, ezért céltudatosan vá-laszz hibakeresőt. Rendszerint már a hibakereső puszta bekapcsolása is növeli a rendszer

' terhelését anélkül, hogy akár egyetlen sort is megvizsgált volna. Az üzleti célú online-rend-szerek csapnivaló teljesítményét gyakran a tesztcélokat szolgáló hibakereső és naplózó rendszer maradványai okozzák. Vagyis a rendszer üzembe helyezését megelőző minőségbiztosítási teszteléskor soha ne mulaszd el eltávolítani a hibakereső kódot.

~ A következő utasítások használhatók a hibakeresővel:

- debugger_on() - Egy címet fogad el paraméterként, és a címhez kapcsolja a .. .. hibakeresőt. Ez az utasítás még fejlesztés alatt áll, ezért ebben a könyvben nincsenek példák. . j/-
- debugger_off() - Ha csak a kódod egy részén akarsz hibakeresést végezni, akkor kapcsold ki a hibakeresést a debugger_off()-utasítással.

DNS- és MX-rekordok

Az URL-ek neveket tartalmaznak, de az Internet-címek számokból állnak, ezért a DNS számokra fordítja le a neveket. A következő utasításokkal megkaphatod a fordításhoz szükséges nyers rekordokat:

- **checkdnsrr()** Ez a csak Unix-on működő utasítás megvizsgálja, hogy egy IP-cím vagy egy host-név szerepel-e a DNS-rekordokban. A checkdnsrr()-t a „DNS-rekordok vizsgálata" című Gyors megoldásokban használjuk.
- getmxrr() Egy Internet host-névhez tartozó Mail Exchange-rekordokat (MX, Levél csere) adja vissza. Hasonlóan működik, mint a checkdnsrr()-utasítás MX-opcióval.

Host-nevek ^w .^m

Ha szükséged van egy host-névre, akkor három utasítás lehet a segítségedre. A *bost-név* egy domain név alatt elérhető szerver neve. Például a **www.petermoulding.com** URL-hez a www a szerver neve és a **petermoulding.com** a domain név. Sok modern website több szervert használ egy domain név alatt, ezáltal láthatatlanná válik az általuk használt technológia azok számára, akik a host-névre keresnek. Valójában az is előfordulhat, hogy egy domain névhez nem tartoznak meghatározott host-nevek, és minden tevékenységet a protokoll oszt szét a szerverek csoportjai között. Az emberek régebben ilyesmit állítottak be a HTTP-oldallekérések kiszolgálására, mint a **www.petermoulding.com,** és minden más szolgáltatáshoz valami hasonlót, mint a **mail.petermoulding.com.** Egy modern weboldalon a **petermoulding.com** kezel minden mást és a webszerver, vagy egy intelligens előoldali router irányítja a levéllekéréseket a mail-szerverhez, és a HTTP-oldallekéréseket a webszerverhez. A következő utasításokkal tehetsz szert a host-névre:

- **gethostbyaddr()** Argumentumában egy IP-cím szerepel, és visszatérési értéke az a host-név, amelyhez a megadott IP-cím tartozik. Ha nincs hőst allokálva az IP-címhez, akkor az IP-címet adja vissza.
- **gethostbynameQ** Paramétere egy host-név és a hőst IP-címét adja vissza. Ha nincs ilyen nevű hőst, akkor változatlanul adja vissza a host-nevet.
- **gethostbynamel()** Paramétere egy host-név, a visszaadott érték pedig egy tömb, amely a host-hoz tartozó IP-címeket tartalmazza. Ha nincs ilyen nevű hőst, akkor a visszatérési érték hamis. Egy címről egy másikra történő átálláskor egy host-hoz több IP-cím is tartozhat. Ha egy hőst több szolgáltatást nyújt, akkor is több IP-címmel rendelkezhet.

IP-címek

Az Interneten jelenleg az IP 4. verzióján (IPv4) alapulnak az IP-címek, amelyek valahogy így néznek ki: 125.3.0.7. Az IPv4 címeket 32 bites jelöletlen egész számként kezelik az Interneten. Már útban van az IP 6. verziója (IPv6). Ez 128 bites címeket használ, melyek 8 db 16 bites részből állnak. Az IPv6 címek valahogy így néznek ki: f4cO:101:10:b05:110:2cOe:llc4:aQ14. A PHP-nek 128 bites jelöletlen egészeket kell kezelnie, hogy együttműködhessen az IPv6-tal.

\i

ip2long()

Az ip2long()-utasítás vesz egy IP-címet, és az IPv4 belső hálózati számává konvertálja, amely egy 0-tól 4 milliárdig terjedő egész szám. A következő kód a 125.3.0.17 IP-címet egy belső számmá konvertálja, és megjeleníti az eredményt:

```
print("<br>".ip21ong("125.3.0.17"));
```

A következő eredmény egy negatív számot mutat, mivel a PHP előjeles 32 bites egészként kezeli a számokat:

-16580591

Az sem segít, ha a típus meghatározásával megpróbálod felülbírálni a mezőformátumot, ami azt jelzi, hogy a számot az ip2long()-utasítás konvertálta előjeles egész formátumba. A következő megközelítés a szám manuális visszakonvertálása a maximális 32 bites egész érték hozzáadásával, ami 2 a 32. hatványon, vagy 256 a 4.-en, és valamivel meghaladja a 4 milliárdot. A következő két sor két példát mutat a szám meghatározására (használd azt, amelyiket könnyebben megjegyzed):

```
define("fourbillion",4294967296);
$four_billion =256 *256 *256 *256;
```

A következő kód hozzáadja a meghatározott **fourbillion** (négymilliárd) értéket az **ip2long()** eredményéhez, ha az eredmény kevesebb, mint nulla:

```
$x = ip21ong("255.3.0.17");
if($x < 0)
    {
    $x += fourbillion;

print("<br>" . $x);
```

Az eredmény:

```
4278386705
```

Az IP-cím (a pontozott forma, mint pl. 255.3.0.17) jobb megértése érdekében a következő kód az előzőtől eltérő módon készít egész számot a címből. Az IP-cím minden része (a pontokkal elválasztott számjegyek) egy 0-tól 255-ig terjedő szám, és mindegyik rész 8 bitet képvisel a 32-ből, a magasabb rendű bitekkel kezdve. A következő kód részekre vágja a pontozott formátumú IP-címet, majd végigmegy a részeken, és az előző összeget megszorozza 256-tal, majd hozzáadja a következő részt az összeghez:

```
$a = explode(".", "255.3.0.17");
$b = (double) 0;
while(list($k, $v> = each($a))
                                    .ki >
$b = (double) ($b * $v;
256) } print("<br>" . Sb
);
```

Az eredmény újra: ;

```
4278386705
```

Ha szeretnéd, hogy az utoJsÓKód az IPv6-címeket is kezelje, akkor cseréld le a pontot kettőspontra az explode()-utasításban, a $b*256-ot pedig $b*256*256-ra, mint itt látható:

```
$b =(double) ($b *256 *256)+hexdec($v) ;
```

A $v-t hexadecimálisból szintén egésszé kell konvertálni, ezért illesztjük a hexadecQ-utasí-tásba. Ha szeretnél töménytelenül sok információt kapni az IPv6-ról, akkor látogasd meg a www.ipv6.org/ oldalt.

long2ip()

A long2ip()-utasítás a belső IP hálózati számot hagyományos IP-címmé alakítja, mint itt látható:

```
print("<br>".long2ip (2097348625)) ;
```

Az eredmény:

```
125.3.0.17
```


Protokollnevek és -számok

Kevés PHP-utasítás igényli a protokoll meghatározását, és amelyeket használtam, mind a protokoll nevét igényelték. Akkor lehet szükséged a protokoll számára, ha mélyre merültél a hálózati szoftver vagy az adatátviteli naplóállományok rejtelmeiben. A következő két utasítás lehetővé teszi a protokollnevek és protokollszámok közötti átjárást, és mindkét utasítás a Unix alatt megtalálható /etc/protocols-fájlra hivatkozik (ami azt is jelenti, hogy Windows vagy Windows NT alatt nem használható):

- getprotobyname() Paramétere a protokoll neve (csupa kis vagy csupa nagybetűvel) és hamisat ad eredményül, ha nem találja a protokoll nevét.
- getprotobynumber() A protokoll száma alapján megadja a protokoll nevét, vagy hamisat ad, ha nem találja a számot.

Szolgáltatás nevek és port-számok

Alkalmanként egy szolgáltatás használatához szükséged lehet a port számára, amelyen elérhető a szolgáltatás. A következő utasítások lehetővé teszik, hogy megállapítsd egy szolgáltatás port-számát, vagy egy porthoz kapcsolódó szolgáltatás nevét.

getservbyname()

A getservbynameQ-utasításnak a szolgáltatás és a protokoll nevét kell megadni, és a megadott kombinációhoz kapcsolódó port-számot adja vissza.

getservbyportQ

A getservbyport()-parancs paraméterei a port száma és neve. A szolgáltatás nevét adja eredményül,,vagy hamisat, ha nem találja a protokoll számát. A következő kód 1-től 199-ig _- vizsgálja végig a portokat olyan szolgáltatás után kutatva, amely a User Datagram Protocol-t (UDP-t) használja:

```
for($i = 0; $i < 200; $i + + )
    if($n = getservbyport($i, "udp"))
        print("<br>" . $n);
        $i
```

Csatolók

A következő rész a csatolókhoz (socket) tartozó utasításokkal foglalkozik és azokat a kapcsolódó utasításokat is lefedi, amelyek nem szerepeltek a 13. fejezetben.

pfsockopen()

A **pfsockopen()** az **fsockopen() állandó** változata. Paramétere egy host-név, egy port-szám, egy hibaszám, egy hibasztring, és egy időkorlát (másodpercben). Visszatérési értéke a csatolót meghatározó erőforrás-azonosító. Az utasítás egy Internet-kapcsolat létrehozásához megnyit egy TCP-csatolót egy hőst (domain vagy IP-cím) egy portjához. Unix-kapcsolatokhoz a host-név értéke a csatoló útvonala, a port-szám pedig 0. UDP-kapcsolatok létrehozásához a host-nevet ebben a formában kell használni: **udp://host.**

A **pfsockopenQ** eredményeképp kapott erőforrás-azonosítót egy fájlfüggvényhez, pl. az fgets()-hez a fájl azonosítójaként lehet használni. Ha egy hiba lép fel, a hibakód és a hibaüzenet a hibakód és hibasztring változókba kerül:

```
$hostname = "a.hőst.com";
$errno = 0;
$error ="";
if($socket = pfsockopen($hostname, 80
                        $errno , $error))
    print("<br>Socket open for " . $hostname);
else
    print("<br>pfsockopen error. Hőst: " . $hostname .
        ", error: " . $errno . ": " . $error);
```

socket_get_status()

A socket_get_status()-parancsot a pfsockopen()-nel, fsockopen()-nel vagy a socket()-pa-ranccsal lehet használni. Leginkább az első kettőhöz illeszkedik, a passzívan figyelő alkalmazásokhoz pedig inkább az utóbbival használatos.

A socket_get_statús() argumentumában a pfsockopen() vagy fsockopen() eredményeképp kapott csatoló azonosító szerepel és egy státustömböt ad eredményül, mint a következő kód mutatja:

```
$array = socket_get_status($socket) ;
while (üst ($k, $v) = each ($array))
{
    if ($k == "unread_bytes") '  íO'#^ - - .
                                                                    IV;
        print ("<br>Unread bytes:

    elseif($v)
    print("<br>"  .  $k);
```

A tömb első három eleme logikai típusú, és általában akkor érdekesek, ha értékük igaz. Többnyire a negyedik értékre vagy kíváncsi, amely a csatolóban maradt adatok hossza.

socket_set_blocking()

A következő példában a socket_set_blocking()-utasítással bekapcsoljuk (igaz értékre állítjuk), majd kikapcsoljuk (hamis értékre állítjuk) a blokkolást.

```
socket_set_blocking($socket, true) ;
socket_set_blocking($socket,falsé);
```

Mikor a blokkolás be van kapcsolva, az fgetsQ és hasonló utasítások arra várnak, hogy egy csatolótól adatokat kapjanak, mielőtt visszaadnák a vezérlést a PHP-szkriptnek. A blokkolás akkor előnyös, ha a szkripted semmit nem tesz, mindössze egy tartalomszerverről származó adatokra vár. Ha a blokkolás ki van kapcsolva, az fgets() és társai azonnal visszaadják a vezérlést, még akkor is, ha még nincsenek meg az adatok. Akkor kapcsold ki a blokkolást, ha azt szeretnéd', hogy a szkripted egy új rekordot keressen, majd elmenjen és valami mást csináljon a rekorddal vagy akár anélkül, w"—' —— ■ --"_^i

A blokkolás összezavarja az embereket, ezért még egyszer átfutok az opciókon. Létrehozol egy szkriptet a szervereden, amely más szerverekről gyűjt tartalmat. Úgy állítod be a szkriptet, hogy soha ne lépje túl az időkorlátot, és magára hagyod. Azt szeretnéd, hogy a szkript örökké fusson, és semmi mást ne tegyen, csakis tartalmat gyűjtsön. Ezért bekapcsolod a blokkolást a szkriptben, és az addig vár, amíg nem lesz adat a csatolóban, majd feldolgozza az adatokat. A szkript órákat vagy egy egész éjszakát tölthet a következő adatra való várakozással, vagyis az összes időkorlátot 10 000 vagy 100 000 másodpercre kell állítanod, hogy a szkript boldoguljon a feladattal.

Egy másik megoldás, ha kikapcsolva hagyod a blokkolást, és a szkript visszaadja a vezérlést, ha nincs adat. Úgy írd meg a szkriptet, hogy ebben az esetben befejezze a futást. Állítsd be a cron-t vagy más ütemező programot, hogy rendszeres időközönként, mondjuk öt percenként futtassa a szkriptet, és a szkript minden kapcsolódáskor felveszi az elérhető adatokat. Ha ezt a megközelítést alkalmazod, meg kell bizonyosodnod arról, hogy a forrás képes az adatok pufferelésére a kapcsolatok közötti időszakban. A forrásnak ezen kívül olyan gyorsan kell tudnia továbbítani az adatokat, amilyen gyorsan te olvasni tudod őket, mert különben folyamatosan adatkiesés fog fellépni.

socket_set_timeout()

Ha jól számolok, a következő socket_set_timeout()-kód 2,5 másodperces csatoló időkorlátot állít be:

```
socket_set_timeout($socket,2,500000) ;
```

A socket_set_timeout()-utasítás paraméterei a csatolóazonosító, az időkorlát másodpercben és az időkorlát mikroszekundum értéke. Az utasítás összeadja két időkorlát értéket. A mikroszekundum milliomod másodpercben van megadva, vagyis 500 000 mikroszekundum fél másodperc.

Rendszernapló

Hogyan rögzíted a hálózati eszközök használatát és hibáit? Az egyik lehetséges megoldás, hogy a rendszernaplóhoz írod a bejegyzéseket. Mielőtt nekifognál a tesztelésnek, ellenőrizd, hogy olvashatod-e a rendszernaplót. Ha van jogosultságod a rendszernaplóhoz, és a rendszergazda rendszeresen elemzi a rendszernapló-állományt, akkor a naplóbejegyzésekben rögzítheted az egyes honlapok, szerverek és rendszerek közötti kapcsolódási problémákat. A rendszergazda megoldhatja a adatkieséseket, időtúllépéseket és minden egyéb leállást, ami megszakítja a szkriptjeidet. Windows NT-n a syslog-üzenetek (rendszernapló) a Windows NT Event Log (Eseménynapló) állományába kerülnek.

Ha nincs hozzáférésed a rendszernaplóhoz, és senki nem elemzi a naplóállományt, akkor a 8. fejezet fájlfüggvényeit használva megírhatod a saját naplódat. A Gyors megoldások között „Saját napló írása" címmel találsz egy rövid verziót minimális hibakereséssel.

define_syslög_variables()

A define_syslog_variables()-utasítás a többi rendszernaplóparancs által használt konstansokat deklarálja. Ezért ezt az utasítást le kell futtatni egyszer az openlogQ vagy a syslogQ használata előtt.

Az openlog()-utasítás megnyit egy kapcsolatot a rendszernaplóhoz, de a syslog() automatikusan megnyitja a kapcsolatot, ezért az openlogQ nem szükséges, hacsak nem változtatod meg az alapértékeket. Viszont ha a webszervered vagy a PHP ugyanebbe a naplóállo-

mányba ír, akkor a megváltoztatott értékek rájuk is ugyanúgy vonatkoznak. Én inkább távol tartom az Apache és a PHP hibáit a rendszernaplótól, ezért ez csak akkor okoz problémát, ha a honlapod sok más honlappal ugyanazon a szerveren osztozik, és nem változtathatod meg a konfigurációs fájlokat.

Az openlog() argumentumában egy sztnngazonosító, egy opció és egy eszköz paraméter szerepel. Ha az opció paraméterértéke LOG_CONS, akkor a rendszernaplóba történő' beírás kudarca esetén a rendszefkortzolra irányítja a hibaüzeneteket. A LOG_NDELAY-érték hatására azonnal megnyit egy kapcsolatot a rendszernaplóhoz. Az alapbeállítás szerinti LOG_ODELAY-érték elhalasztja a rendszernapló-kapcsolat megnyitását, amíg az üzenet nincs naplózva. Lássunk egy tesztet az eszközparamétert LOG_USER-re állítva:

```
openlog("My     test:",LOG^NDELAY,LOGJJSER);
```

l \%sv nöiahtiDq JIOL

syslog()

A syslog()-utasítás argumentumában egy prioritásparaméter és egy üzenetsztring szerepel. A prioritás az openlog() eszközparaméterével közösen azt határozza meg, hogyan kezelje az utasításüzenetet. A következő felsorolás a prioritásokat mutatja sorrendben. Kérdezd meg a rendszergazdádat, ő hogyan ellenó'rzi az egyes prioritásokat. Ha pl. azt jelented, hogy egy távoli tartalomszerver nem elérhető, akkor lehet, hogy azt szeretnék, hogy olyan szinten jelenítsd a hibát, hogy a rendszer egy üzenetet küldjön a rendszergazda csipogójára:

```
LOG_EMERG
LOG_ALERT
LOG_CRIT
LOG_ERR
LOG_WARNING
LOG_NOTICE
LOG_INFO
LOG_DEBUG
```

A következő kód egy egyszerű teszt, a révén meggyőződhetsz arról, hogy működik a syslog() és hogy megfelelő hozzáféréssel rendelkezel (minden használni szándékolt prioritási szintet próbálj ki):

```
syslog(LOG INFO,"I am about to flood this log with messages");
```

closelog()

A closelog()-utasítás nem szükséges a rendszernapló lezárásához, de ha hamarabb le akarod zárni a naplóállományt, csak illeszd be a következő sort. A closelogQ-utasításnak nincsenek paraméterei.

```
closelogQ;
```

NFS

- A Network File System (NFS) 3-as verziójának definícióját az ftp://ftp.isi.edu/in-notes/rfcl813.txt-dokumentumban találod, a következő', 4-es verzióról szóló híreket pedig a www.nfsv4.org/-oldalon. Az NFS lehetővé teszi a Unix-rendszerek számára a partíciók megosztását. Ha az l-es meghajtó a b-rendszerben van, akkor a c-rendszerben ugyanúgy
: felmountolhatod az l-es meghajtót, mintha a c rendszer része lenne. A Samba egy olyan termék, amely a Windows-felhasználóknak is lehetővé teszi, hogy bekapcsolódjanak a megosztásba.

Ne feledd, hogy a partíciómegosztás révén elért fájlmegosztás egy kicsit bonyolultabb. Ha valakinek korlátlan joga van egy teljes partíciódra vagy egy könyvtáradhoz, akkor mindent törölhet az adott partíción vagy könyvtárban, vírusokat másolhat a merev lemezre, és álta lában teljes pusztítást végezhet. . ■ > . . - ,. ,

A meghajtó megosztása olyan teljesítményjegyeket eredményezhet, amelyeket gyakran a „totális katasztrófa" szakkifejezéssel illetnek. Képzeld el, hogy megosztod a meghajtódat, ezáltal lehetővé téve, hogy egy másik honlap naponta kimásoljon onnan egy 200 bájt méretű fájlt, amely az aranymosás alapanyagát képező iszap legfrissebb árfolyamát tartalmazza. Na most a másik honlap tulajdonosa úgy dönt, hogy egy kis extra munkaterületet foglal a merev lemezeden egy 30 000 szereplős Quake 3 deathmatch számára. Ekkora adatmennyiség oda és visszamozgatása a hálózaton keresztül (a Quake 3 adat, nem a 200 bájtos aranymosóiszap-fájl) olyan károkat tud okozni a hálózatodnak, mint 300 tinédzser egy közepes méretű pizzának. Neked nem sok marad.

Az NFS a legmegbízhatóbb módja a szerverek ellenőrzésének a hálózaton keresztül, és a meghajtómegosztást úgy használja, hogy intelligensen tudsz hozzáférést biztosítani az adatokhoz. Ha az adatokra tényleg szükség van több szerveren, és a szerverek gyakori hozzáférést igényelnek, akkor vizsgáld meg a replikáció lehetőségét a szerverek között és az NFS-szel csak a replikációt tedd lehetővé, ne az adatokhoz való hozzáférést.

NIS

A Network Information Service (NIS) lehetővé teszi az információk megosztását a szerverek között. A NIS-t régebben YP néven ismerték (Sun Yellow Pages), de a Yellow Pages néhány országban bejegyzett márkanév, ezért a Sun NIS-re cserélte a nevet. Az NIS telepítéséhez Unix alatt az --enable-yp-paraméterrel fordítsd le a PHP-t. " *-

Az új NIS+-hoz további biztonsági elemeket adtak, de még nem elég érett, ezért problémáid lehetnek a NIS+ megbízható implementálásával.

.1310 ■

Térkép . < :o*

Egy NIS-térkép (map) egy relációs adatbázis nézetéhez hasonlóan a kiválasztott adatok meghatározott sorrendben való visszanyeréséhez tartalmaz instrukciókat. Ebben a részben a passwd.byname nevű példa NIS-térképet használjuk, amely a kapcsolódó felhasználónevek sorrendjében nyeri vissza a jelszavakat:

```
$nis_map    ="passwd.byname";          r^..     na: ■     >.;.-     o.r-:. ;
```

yp_get_default_domain()

Az yp_get_default_domain()-utasítás a NIS-keresésekre használt szerver NIS domain nevét adja vmza. A NIS használata előtt minden szerver (vagy hőst) egy NIS domainhez kötődik. Az **yp_get_default_domain()**-utasítás eredménye a kötődésben használt NIS-domain név, amely más NlS-utasításokhoz is használható. Ha az **yp_get_default_domain()** utasítás kudarcot vall, hamis eredményt ad vissza:

```
if($nis    domain    =    yp_get_default_domain() )

    print("<br>"    .
    $nis_domain);                    .-e:
else

print("<br>yp_get_default   domain()     failed.");
```

yp_master()

Az yp_master() paramétere az NIS-domain név és egy NIS-térképnév, eredménye pedig térképet szolgáltató felügyelő számítógép neve:

```
if($nis_master    =    yp_master($nis_domain,     $nis_map))

    print ("<br>"     .    $nis_master);

else
    {
    print("<br>yp order() failed to find: " . $nis map);
    }
```

yp_order()

Ha már birtokodban van a NIS-domain, akkor megkaphatod a NIS-térképet. Az **yp_order()** egy NIS-domain nevet és egy NIS-térképnevet igényel, és a térkép rendszámát adja eredményül. Ha nem találja meg a térképet, akkor hamisat ad eredményül:

```
if ($nis_order    =    yp_order($nis_domain,     $nis_map))
    {
    print("<br>"     .     $nis_order);
    }
else                                                    '       .
    {
    print ("<br>yp_order()     failed    to    find:    "    .    $nis_map);
```

yp_match()- ■ ■ ; **- - ^ d . ^ ^ ^**

Az yp_match() a NIS-térkép és a keresett érték megadása után megtalál egy bejegyzést egy NIS-domainben. Az egyezésnek pontosnak kell lennie, különben az yp_match() eredménye hamis:

```
$user = "fred"
if<$nis entry = yp match($nis_domain, $nis_map, $user))
    {
    print("<br>" . $nis_entry) ;


    print("<br>yp_match() failed to find: " . $user);
```

yp_first()

Az yp_first()-utasítás az első bejegyzést találja meg a térképen, vagy hamisat eredményez. A bejegyzést egy tömbben adja vissza, ami a bejegyzés értékét és kulcsát tartalmazza, lásd a következő példát:

```
if($entry = yp_first($nis_domain, $nis_map))

print ("<br>Key: " . $entry [ "key" ] . ", value: " . $entry ["value"'_ > : ) else

    print("<br>yp_first() failed with map: " . $nis_map);
```

-r<?r é- :, (! vvn-k

yp_next()

Az yp_next()-utasítás egy kulcsot követő első bejegyzést talál meg a térképen. A kulcsot a harmadik paraméterben lehet megadni, és általában az yp_next(), vagy yp_first()-parancs előző futtatásából származik. A bejegyzést egy tömb formájában adja vissza, amely ugyanúgy, mint az yp_first() esetében, a bejegyzés értékét és kulcsát tartalmazza. A következő kód közvetlenül az yp_first()-kód után fűzhető, hogy végiglépkedjen a NIS-térkép bejegyzésein:

```
while ($entry' = yp next ($nis_domain, $nis_map, $entry["key"])) {
    print("<br>Key: " . $entry["key"] . ", value: " .
    $entry["value"]) ;     i
```

. " :bnii oJ D9Xí65 i >

._qy

WDDX 'ö* adatl-ó :■{ ::i -«;•. - *M* ■. lu-.on:,^ 1 *ki:.*<\v..:

A Web Distributed Data Exchange (WDDX), amelynek leírása a www.openwddx.org/ oldalon található, lehetővé teszi, hogy hálózati átvitel céljából XML-formátumba konvertáld az adatokat. Egészeket, sztringeket, tömböket és objektumokat konvertálhatsz egyszerű

sztringgé, amely session-rekordként tárolható, egy adatbázisba helyezhető, e-mailbe illeszthető'vagy Internet Protokollal továbbítható. A fogadó alkalmazás WDDX-et használva visszakonvertálhatja az adatokat az eredeti formátumba. Az adatstruktúra sztringgé konvertálását *besorolásnak (serializmg)* nevezik, az eredeti adatok visszaállítását a sztnngből pedig *kisorolásnak (desenalizing)* hívják. A WDDX Windows-os telepítéséhez a c:\Program Files\php\dlls\expat.dll fájlt másold be a c:\windows\system könyvtárba. Windows NT-n az expat.dll-t másold a c:winnt\system32\ könyvtárba. A WDDX XML-t használ, és az XML bele van fordítva a Win32 bináris állományaiba. Unix alatt telepítsd a legfrissebb Apache verzióhoz biztosított expact-könyvtárat és a PHP-t a —with-xml, és —enable-wddx paraméterekkel fordítsd le.

A session-ökhöz nem szükséges a WDDX, mivel a PHP rendelkezik egy beépített besoro-lóval, de úgy is dönthetsz, hogy a WDDX-et használod a PHP besorolója helyett. Mindeddig nem találtam nyomós okát, hogy lecseréljem a PHP saját besorolóját (serializer) WDDX-re és a WDDX-nek számos hátránya is van. A WDDX egy külső könyvtártól függ, hosszabb session-rekordokat eredményez, a WDDX XML-tag struktúrája nem a session-adatokhoz van optimalizálva, és a WDDX jelenleg nem érhető el a GPL nyílt forráskódú lincence alatt, mert még mindig a Macromedia birtokában vannak a szerzői jogok.

WDDX csomag

Egy WDDX által besorolt (bekódolt) adatokat tartalmazó sztringet WDDX-csomagnak hívnak. Elméletben egy WDDX-csomag beágyazható egy nagyobb sztringbe, és nagyobb sztringtől függetlenül visszanyerhető. Egy weblap, egy fájl vagy egy e-mail is tartalmazhat a WDDX-csomagot és a wddx_deserialize()-utasítás visszafejti a csomag tartalmát, anélkül, hogy el kellene különítened a csomagot a sztring többi részétől. A gyakorlatban a wddx_deserialize() meghívja a PHP XML-eszközét, amely az expact könyvtárat használja. A visszafejtés minősége az egész XML-sztringen múlik. Ha a teljes weblap vagy fájl, illetve sztring jól formázott XML, és egyik része sem okoz problémát a WDDX-nek, akkor a WDDX-csomag használható lesz. Ha a csomag visszafejtésének befejezése előtt XML-hi-bák lépnek fel, akkor a visszafejtés kudarcot vall:

- wddx_serialize_value() Egy értéket fogad, amelyet egy WDDX-csomagba rendezve egy sztringben ad vissza.
- wddx_serialize_vars() Egy vagy több változót fogad, amelyeket egy WDDX-csomag ba rendezve egy sztringben ad vissza.
- wddx_packet_start() Paramétere egy opcionális megjegyzés, eredménye egy WDDX-csomag építéséhez használható csomagazonosító.
- wddx_addjvars() Egy csomagazonosítót és egy vagy több változót fogad, és hozzá adja a változókat a csomaghoz.
- wddx_packet_end() Egy WDDX csomagot tartalmazó sztringet egy asszociatív tömbbé alakít. \ *,M)l)* j-atmar
- wddx_deserialize() Egy WDDX csomagot tartalmazó sztringet fogad el, és asszociatív tömbként adja vissza a csomag tartalmát.

Az „XML-be kódolás WDDX-szel" Gyors megoldásokban példákat találsz a WDDX utasítások használatára és a PHP serialize() utasításával való összahasonlításra.

CORBA.....;1T±-"T1'

A CORBA a Common Object Request Bróker Architecture (Egyszerű Objektumlekérést Közvetítő Architektúra) (www.corba.org/).

Ha azt szeretnéd, hogy az objektumok kommunikáljanak egymással, de nem szeretnéd folyamatosan átírni az egyes objektumokat, akkor elég egyszer átírnod őket, hogy tudjanak a CORBA-n keresztül kommunikálni és hagyhatod a CORBA-t, hogy tolmácsoljon az objektumok között. Elméletben az objektumok különböző gépeken is lehetnek, de természetesen a teljesítmény siralmas volna, és a hálózat igénybevételét visszaélésnek lehetne minősíteni, ha az alkalmazásod legaktívabb részébe hálózati hozzáférést iktatnál.

Az ORBit egy objektumlekérés-közvetítő, amelyet a GNOME-projekthez terveztek (http://gnome.org/). A GNOME egy ingyenes, nyílt forráskódú szoftver, amellyel Unix-os felhasználói felületet lehet létrehozni. A projekt a CORBA-t használja a Bonobo nevű saját komponensmodelljének felépítéséhez. A PHP néhány CORBA-utasításához az ORBit előtagot használják, más utasítások a satellite_-előtaggal vannak ellátva.

A PHP-ben vannak kísérleti utasítások, amelyek révén egészen új programozási stílussal kapcsolódhatsz a CORBA-objektumokhoz. Mennyire kísérletiek? Képzelj el egy embert, aki odasétál hozzád, és megkérdi, szeretnél-e vásárolni az első eldobható papírejtőernyőből. Az ügynök biztosít arról, hogy még soha senki nem sérült meg a papír ejtőernyőtől. Bár már sokan meghaltak a fölbe csapódástól, de a kérdéses papírernyő még senkinek nem okozott bántódást.

A következő függvények leírása az aktuális dokumentáción alapszik, és nem a weblapomon szerzett valós tapasztalatokon. Ha szeretsz objektumokkal dolgozni, akkor kedvedre lesznek ezek az utasítások. Sok más ember számára ezek az utasítások túlzásnak számítanak majd. Miért használnál CORBA-t, mikor az ODBC és más megközelítések egyszerűbb kommunikációt kínálnak az alkalmazások között? A CORBA képes automatizálni bizonyos bonyolult kommunikációkat a különböző szállítóktól származó alkalmazások között, és hálózaton keresztül is működik, tehát az emberi erőforrás alkalmazásod elküldheti egy dolgozó jelzáloghitel-kérelmét a bank jelzálogfeldolgozó szerverének, válaszolhat a banknak az alkalmazott bérét, házas voltát, bonuszait, betegszabadságát, DNS-ét, gyógyszersze-dési adatait vagy bármi egyebet firtató kérdéseire, még az elutasító értesítést is fogadhatja, mindezt emberi beavatkozás nélkül. Ne feledd azonban, hogy a CORBA az előttről származik, mielőtt az XML népszerűvé vált volna, és ma már sok business-to-bisiness adatmegosztási séma egyszerű XML-t használ (20. fejezet). Mielőtt elmerülnél a CORBA-ban, vizsgáld meg az iparágadban jelenleg használt adatmegosztási rendszereket.

orbitobject() - ^

Az orbitobject()-utasítás egy ORBit-objektum egy új előfordulását hozza létre az Interoperable Objekt Reference-t (IOR, Együttműködő Objektumreferenciát) használva paraméterként. A következő kód azt mutatja, hogyan kellene az utasításnak működnie, feltéve, hogy az IOR-t már megadtad a $ior-változóban.

```
$object =new orbitobject($ior);
```

orbitenumQ

Ha már hozzáfersz az objektumhoz, akkor szükséged van egy hivatkozásra az objektumon belül, és ehhez használod az orbitenum()-utasítást. A következő kód egy azonosítóhoz tartozó felsorolásazonosítót ad eredményül. Az azonosítók listáját a CORBA-objektumod készítője biztosítja:

```
$enumeration =new orbitenum("nameofsomething");
```

Az azonosítóknak van egy hosszabb, formálisabb nevük, amelyet a következő kód mutat. Az IDL az Interface Definition Language-re (Interfész definiáló nyelvre) vonatkozik, a név végén a :1.0 pedig a verzió ellenőrzését teszi lehetővé:

```
$enumeration =new orbitenum("IDL:nameofsomething:1.0") ;
```

orbitstructQ

Az **orbitstruct()**-utasítás eredményeképp kapott struktúrával távoli objektumokat lehet elérni:

```
$structure =new orbitstruct("structure" ) ;
```

A struktúrát feltöltheted értékekkel, és hozzáadhatod az orbitobject()-utasítástól kapott objektumhoz, illetve arra használhatod a struktúrát, hogy megkapd az objektum értékeit.

satellite_caught_exception()

A satellite_caught_exception() igazat eredményez, ha az előző orbit-utasítás hibát okozott. A következőképpen teszteld az utasítást:

```
if (satellite_caught_exception ())
    printC'Error occurred.");
```

satellite_exception_id()

A **satellite_exception_id()-utasítás** egy hibahivatkozást ad meg az objektumtárból, ami nagyjából megfelel a PHP-hibafüggvényei, pl. a **mysql_errno()** által visszaadott hibakódnak:

```
if (satellite_caught_exception() )
    ("<br" . satellite_exception__id () ) ;
```

satellite_exception_value()

A satellite_exception_value() egy struktúrát ad vissza, amely a legutolsó hibáról tartalmaz adatokat:

```
if (satellite_caught_exception ())
    $error_object = satellite_exception_value() ;
```

Az utasítás nagyjából megfelel a mysql_error()-nak vagy hasonló hibaüzenet utasításoknak. A hibát okozó objektum és az objektum készítőjének hibakezelési dokumentációja alapján úgy kell elkészítened a kódot, hogy végigmenjen a hibaobjektumon.

Tömörítés

A tömörítés növeli a CPU-használatot, annak érdekében, hogy tárhelyet vagy hálózati forgalmat takarítson meg. A helyi tárhely vagy helyi hálózat kímélése rendszerint nem indokolja a be- és kitömörítéssel járó többlet processzoridő'-felhasználást, de költségeket takaríthat meg, ha fizetsz azért, hogy adatokat továbbíts a világ bármely pontjára, vagy azért, hogy valaki más szerverén tárolhasd az adataidat.

Melyik a legjobb tömörítési fajta? A népszerűség segít a kérdés eldöntésében. Ha egy népszerű tömörítési eljárást választasz, akkor valószínűleg a fogadó fél is rendelkezik majd a kitömörítő szoftverrel. Ha nem, akkor a kitömörítő szoftvert is el kell küldened a tömörített adatokkal. -wj

A sebesség szintén segít a kérdés megítélésében. A hálózatokat megcélzó tömörítő eljárások mind a be-, mind a kitömörítést nagy sebességgel végzik. A kiadókat megcélzó tömörítő szoftverek esetében megfelelő lehet a lassabb tömörítés, mert a tömörítést csak egyszer kell elvégezni, az eredményül kapott állomány extra kicsi méretű lesz, és a kiadó így több fájlt tud elhelyezni egy CD-n. A kiadók által használt tömörítők a kitömörítési sebességre helyezik a hangsúlyt a betömörítés rovására, mivel a fizető fogyasztók csak kitömörítést végeznek.

Bizonyos eljárások használatát szerzői jogi problémák korlátozhatják, vagyis győződj meg arról, hogy a tömörítő módszer, valamint a szoftver, amelyben a módszert használják, mentes a szerzői jogoktól.

.y

A bzip2 minden bizonnyal egy új tömörítő eljárás, mivel nem hallottam róla mindaddig, amíg nem fedeztem fel a http://sources.redhat.com/bzip2/ oldalon. A forráskód nyílt, BSD stílusú licenccel rendelkezik, és felveszi a versenyt a GNU-Zipp forráskódjával.

bzcompressQ

A bzcompressQ-utasítás a bzip2-ehárást használve betömöríti az argumentumában szereplő sztringet: f ' ^

```
$compressed =bzcompress(Sdata);
```

Az opcionális második paraméter a tömörítéshez használt blokkok számát adja meg. Értéke 1-től 9-ig terjed, az alapérték 4. Az opcionális harmadik paraméter egy workfactor nevű értéket ad meg, amely O-től 250-ig terjedhet, és az alapbeállítás szerint 30. Kevés információ áll rendelkezésre a workfactor hatására vonatkozóan, és nincs közvetlen ok sem a blokkok számának, sem a workfactor-nak a módosítására.

n

bzdecompressQ «■..«.- »

A **bzdecompress()** parancs a bzip2-eljárással kitömöríti a paraméterében szereplő sztringet:

```
$data =bzdecompress($compressed);          ■     í
```

Az opcionális második paraméter hatására a bzip2 kevesebb memóriát használ, és fele olyan gyorsan fut (ami általában nem jó ötlet). ^ ,d3 ,onOi(:Í!f;t **vtp** =•>- -

bzope/70

A bzopen() igen hasonlóan működik a 8. fejezetben tárgyalt fopen()-hez. Egy fájlnevet fogad el és megnyitja írás vagy olvasás céljából, amit a második paraméter w- vagy r-értéke határoz meg. A parancs egy erőforrásazonosítót vagy más néven fájlazonosítót ad vissza, amelyet más bz-utasításokban használhatsz:

```
if($fíle    =                    T_            01          ; bi
bzopenC'test.nz",               i^-í

    print ("<br>bzopen worked.");

else {
    print("<br>bzopen failed.");                              A
                                          £íi     ti*
```

bzreadQ

A bzread()-utasítás a fájlban lévő adatokat adja eredményül egy fájlazonosító és egy opcionális olvasási hossz alapján. A hossz alapbeállítás szerint 1024 bájt:

```
$data   =bzread($file);
```

bzwríteQ

A bzwrite() paraméterei a fájlazonosító, az adat és egy opcionális hossz, és hozzáírja az adatot a fájlhoz. Ha meg van határozva a hossz, akkor leáll, mikor elérte az adott hosszt. A **bzwrite()** nem működne az előző részben bzopen()-nel megnyitott példafájllal, mert a fájl olvasási hozzáférésre volt megnyitva:

```
if ( Ibzwrite ($file,    $data) )                    '■TI,.D
   1                                                   *
   print ("<br>bzwrite   failed.");           :K       £■■
```

bzdoseQ ,-..-■••

A **bzclose()** majdnem ugyanúgy működik, mint az **fclose();** a fájlazonosító alapján bezárja a fájlt:

```
if (!bzclose($file) )          —,.,,.„ =>v«ir

                                         rra
    print ("<br>bzclose
    failed.");
```

bzflushQ

***A* bzflush()** paramétere egy fájlazonosító, és az utasítás kikényszeríti az írásra váró adatok kiírását a megadott fájlba: . ■■v .... - ,.-•..*«>.« «'■ .-<g 7^9 zá ,m

```
if (!bzflush($file))
    print("<br>bzflush failed.");
```

bzerrnoQ ™><^d

A bzerrno()-utasítás egy fájlazonosítót fogad, és egy hibakódot ad vissza, ha hiba lépett fel a fájlhoz való utolsó hozzáférés során:

```
print("<br>bzerrno:".bzerrno());
```

bzerrstrQ

A bzerrstr()-utasítás egy fájlazonosítót fogad, és egy hibasztringet ad vissza, ha hiba lépett fel a fájlhoz való utolsó hozzáférés során:

```
print("<br>bzerrstr:".bzerrstr());
```

bzerrorQ

A bzerror()-utasítás egy fájlazonosítót fogad, és egy hibakódot és hibasztringet tartalmazó tömböt ad vissza, ha hiba lépett fel a fájlhoz való utolsó hozzáférés során:

```
$array =bzerror();
print("<br>bz error ".Sarray ["errno"] ..",".$array [ "errstr"]);
```

A gzip a GNU (www.gnu.org/software/gzip/gzip.html) nyílt forráskódú tömörítő szoft vere, amely először használta a .gz formátumot a tömörített fájlokhoz. A zlib (www.gzip.org/zlib/) ugyanezt a funkcionalitást biztosítja a PHP számára. Úgy tűnik, a zlib kezd előtérbe kerülni, mert a gzip és más alkalmazások is áttérnek a zlib-könyvtár használatára. Mindkettő jól működik, és lehetővé teszik a zip (.zip) és a GZ (.gz) fájlok ol vasását. A PHP GZ utasításai sztringekre is alkalmazhatóak hálózati átvitel előtt. A zlib ak tiválásához Windows NT és Windows 2000 alatt állítsd le az Apache-ot, másold a c:/Program Files/php/extensions/php_zlib.dll fájlt a c:/winnt/system32/ könyvtárba, távolítsd el a pontosvesszőt a következő példában szereplő sorból a php.ini-ben, és indítsd újra az Apache-ot:

```
; extension=php_zlib .dll
```

Ha a PHP CGI-ként használod, akkor nem kell újraindítanod a webszervert. Más Windows verziókhoz másold be a c:\Program Files\php\extensions\php_zlib.dll fájlt a c:\windows\system\ könyvtárba.

Ha a legújabb PHP-verzióval rendelkezel, próbáld meg az fopen()-utasítást használni, mint itt:

```
$file =fopen("zlib:/test.gz","r");
```

A PHP fájlfüggvényei már elfogadják a zlib: útvonalelőtagot, és automatikusan használják a zlibttömörítést. Az „Adatok tömörítése zlibbel" Gyors megoldásban példákat találsz a zlib használatára, és egy gyors tesztet, amely a zlib:-opcióval használja a standard fájlfüggvényeket.

Gyors megoldások

i

DNS-rekordok vizsgálata

h s n-

Egy domain létezését számos módon ellenőrizheted. A DNS-rekordok vizsgálatából megtudhatod, hogy a domain létezik-e, de azt nem, hogy a domain egy szervere online állapotban van-e vagy sem. A legjobb módja annak, hogy meggyőződ) egy szerver online állapotáról, ha egy hálózati- vagy fájlfüggvényt használva megnyitsz egy fájlt a szerveren. A DNS-rekordok ellenőrzése inkább arra használható, hogy meggyőződj arról, hogy egy domaint már regisztráltak-e.

A következő példa a **checkdnsrr**()-parancsot használja annak kiderítésére, hogy regisztrálta-e már valaki a yahoo.com site-ot:

í-7. q

```
if (checkdnsrr ("yahoo. com") )      ■   '  - í / ı . »  „'>■,,,,„.■,„.,

    print ( "<br>checkdnsrr () worked. "); ,{ (i^-j-^t
    }                    .  ,  ,  .= ,,,.;: 3.-h;r ,.,  ).
else
    {                              y:f."     S-.-h     «  ▮ ▮    ▮   .
    print ("<br>checkdnsrr ()   failed.");       (    -ütrifc
```

Az eredmény Windows-on vagy Windows NT-n a következőképp néz ki:

```
Warning:checkdnsrr()is  not  supported  in  this  PHP  build
(Figyelem:  a  checkdnsrr()  nem támogatott a PHP ezen
változatában)
```

q

Az egyik Windows-modulból hiányzik a DNS-rekordok vizsgálatára szolgáló kód, ezért ez az utasítás nem működik. A következő Unix-ból származó eredmény azt közli, hogy valaki már regisztrálta a domaint:

```
checkdnsrr () worked.          .....«„„■»».
```

Mi történik, ha egy ilyen nevet ellenőrzői: yahxxxx.com? A következő eredményt kapod, amire számítani lehetett:

```
checkdnsrr()failed.
```

) f ú C

Most futtasd le magad is a tesztet a yahOŰ.com-ra. Vajon gondolt a Yahoo! egy olyan domain regisztrálására, amelyet olyan könnyű elgépelni, mivel a 0 billentyű közel van az o-hoz és hasonlóan is néz ki?

, « - , • . . « . .
íiöisasi MöniBrnoQ «wi* f Hl"

A checkdnsrr() egy második paramétert is elfogad, a type-ot, amelynek az alapbeállítás szerint MX az értéke. Az első paraméter domain vagy host-név helyett egy IP-cím is lehet. A következő kód a Stype-tömbben megadott különböző lehetséges típusokat vizsgál végig a $host-tömbben megadott host/domain neveken, hogy megmutassa, milyen eredményekre számíthatsz:

```
$host[]   =   "yahoo.com";
$host[]   =   "216.115.108.243";
$type[]   =   "MX";
?type[]   =   "A";
$type[]   =   "NS";
$type[]   =   "SOA";
$type[]   =   "PTR";
$type[]   =   "CNAME";        v.<si^'
$type[]   =   "ANY";                                   ^ ,§,.:, K^i>: -
print ("<table   border=\"3\">") ;                     '".
print ("<tr><td> </tdxtd>"   .   $host[0]   .   "</tdxtd>"   .   $host[i;

while (üst ($kt, $t) = each($type)

    print ("<trxtd>" . $t . "</td>");
    reset ($host);
    while (Üst ($kh, $h) = each(Shost)) ieq-()mrtM:>3íb s tbisq oy

        print("<td>");
        if (checkdnsrr($h/ $t)

            print("checkdnsrr() worked." );
  it z -
        else                    r y •  ,—r·          a r          ,r[i]

            print ("checkdnsrr() failed."); }
        print("</td>");
        }                              '".■'.  lnt*  '"
print("</tr>"); }
print("</table>");
```

A teszt eredményeit a 16.1. ábra mutatja. Figyeld meg, hogy az IP-címmel nem volt siker. A Yahoo-nak több száz szervere van, és csak néhánynak van bejegyezve az IP-címe.

____ |yahoo.com 216.115.108.243
MX |checkdnsrr() **worked.** checkdnsrr()
failed ■ fis Ján nsyii tíí
JA **j|checkdnsrr() worked ^heckdnsrrO failed**
l (checkdnsrrO worked. checkdnsrr()
failed.
SOA |checkdnsrr() worked. checkdnsrr() failed.
heckdnsrr() failed **checkdnsrr() failed** „.;■
.,,; ÍCNAME'checkdnsrrO failed. checkdnsrr() failed. ANY
checkdnsirO worked. checkcliisrrO failed.

16.1 ábra Domainek tesztelése checkdnsrr()-rel

MX-rekordok megszerzése >

Az MX-rekordok az Interneten a forgalom átirányítására szolgálnak. Képzeld el, hogy A-ból B-be utazol vonattal, mondjuk Hornsby-ből Sydney-be. Sydney-ben van egy állomás

vet.

ezen a vonalon, a neve Wynyard. Ahhoz tehát, hogy eljuss Sydney-be, a Wynyard-ra kell menned. Egyszerű.

Az Interneten ehhez MX-rekordokat használnak. Ha a B-szerverbe szeretnéd irányítani a forgalmat, de a B-szerver nem közvetlenül kapcsolódik az Internetre, hanem egy C-szerve-ren keresztül, akkor készíthetsz egy MX-rekordot, ami a B-be irányuló forgalmat C-be irányítja, ahonnan majd a C-szerver küldi át B-be. ■ „ . . . .

A getmxrrQ paramétere egy host-név és egy, a host-hoz tartozó MX-rekordokat tartalmazó tömböt ad eredményül. A következő kód végigmegy a host-nevek listáján, amely egy érvényes és egy érvénytelen host-nevet tartalmaz. A kód a getmxrr()-utasítás révén mindegyik host-ról kap egy tömböt, és felsorolja a tömb tartalmát. A getmxrr()-utasítás hamisat eredményez, ha nem találta a host-ot:

```
$host[] = "yahoo.com";                          ..          -..    .<„í . s?-:
$host[] = "yahxxxx.com";     ..,.;. ,,;,,.,,,. ,| ;,-l- -■,. ; . i-.-«t^. ->. ,   í while
(üst ($kh, $vh) = each($host))

    $array = array0;
    if (getmxrr ($vh, $array) )         .- ,:.. ,   . J,   I  ■ ■ i|   < t í

        while (üst ($k, $v) = each ($array) )  '" 31. c5! I .? i-' '

            print("<br>getmxrr(" . $vh . "): " . $v);

        *                              - é ?é lenti re-"
    else                                                                  buri

                                    $vh      ")    failed.");

        print ("<br>getmxrr("                                             rtA
```

1 -..uiT: (5tf iS'rsss-

Az eredmények a következő sorokban láthatók, ahol az első négy a yahoo.com-hoz tartozik, az ötödik az érvénytelen domain névhez. Figyeld meg, hogy az MX-rekordok mail-szerverekre mutatnak. Az MX-rekordok gyakran mutatnak mail-szerverekre, és az MX-rekordok értékét az SMTP-vel használják:

```
getmxrr (yahoo, com) :mx2 .mail. yahoo . com
getmxrr (yahoo . com) :mx3 .mail. yahoo . com
getmxrr(yahoo.com):mta-vl8.mail.yahoo.com
getmxrr (yahoo . com) :mxl .mail. yahoo . com
getmxrr (yahxxxx .com) failed.      >      ■   ■'    -
```

+srrí33p-. ;r;>" ; - rí

A host-név megszerzése *

A *host-név* egy domainen belül megtalálható szerver neve, és az ebben a megoldásban bemutatott függvények mind a host-név meghatározásában segítenek.

A host-név megszerzése cím alapján *w*

A következő kód vesz egy IP-címlistát, és mindegyik címhez megkeresi a host-nevet:

```
$ip[] = "216.115.108.243";
-'' $ip[] = "1.115.18.23";
while (üst ($k, $v) = each($ip))
í

if($host = gethostbyaddr($v))
                                        $host);
   print("<br>gethostbyaddr(\"" . $v . "\"): "

else
    {
   print("<br>gethostbyaddr\"" . $v . "\") failed.");
```

Az első IP-címet a yahoo.com pingelésével kaptuk, vagyis ennek a címnek érvényes hőst névvel kellene rendelkeznie. A második IP-címet egy olyan előtaggal láttuk el, amelyet az Internet kezdeti időszakában osztottak ki, de ez a cím nincs egy host-hoz allokálva:

```
gethostbyaddr("216.115.108.243");img3.yahoo.com
,„, gethostbyaddr("1.115.18.23"):l.ll5.18.23
```

Vedd észre, hogy nem gépelnél be szervernévként „img3"-at, ha a Yahoo!-t akarod elérni. Egyszerűen csak ,>yahoo.com"-ot írnál be és a Yahoo terheléskiegyensúlyozó szoftvere vagy hardvere allokálná a kérésedet a megfelelő szerverhez.

A host-cím megszerzése név alapján

A következő kód két host-nevet keres meg a gethostbyname()-utasítással és az IP-címüket adja eredményül:

```
$name[] = "img3.yahoo.com";
$name[] = "xyzyx.yahoo.com";
while (üst ($k, $v) = each ($name) )

    if($address = gethostbyname($v))

        print ("<br>gethostbyname(\"" .
                                        $address);
    $v else

    print("<br>gethostbyname\"" . $v ,
    }'}                                 "\") failed.");
```

Az első név az előző tesztből származik, vagyis számíthatsz arra, hogy működni fog. A második név, noha nagyon sok szerverük van, egyelőre még nem létezik a Yahoo!-nál.

l-e-, p-tie utaz-). U

Figyelem: Mikor először megírtam a "A host-cím megszerzése név alapján" kódot, akkor megpróbáltam tömbként újra felhasználni a $host-ot, ott, ahol most a $name szerepel. A PHP nehezményezte, hogy egy sztringet próbáltam meg tömbként használni, ezért megváltoztattam a nevet. A másik megoldás az unset($host)-utasítás használata lett volna.

Az eredmény itt látható:

```
gethostbyname("img3.yahoo.com"):216.115.108.243
gethostbyname("xyzyx.yahoo.com"):xyzyx.yahoo.com
```

mart í.H

A gethostbyaddr()-utasításhoz hasonlóan a gethostbyname() egyszerűen visszaadja az input, ha nem találta meg a keresett értéket.

Jobb megoldás lenne, ha az utasítás sikertelenség esetén hamisat adna vissza, és a következő kód pontosan ezt teszi. A felhasználó által definiált gtip()-függvény egyszerűen összehasonlítja a gethostbyname() eredményét az inputtal, és hamisat ad vissza, ha a két érték megegyezik. Ha az érték különbözik az inputtól, akkor természetesen a gethostbyname() eredményét adja vissza. A gethostbyname()-utasítás ilyetén való módosítása akkor hasznos, ha a szkriptednek ki kell emelnie a kudarccal végző'dő kereséseket:

```
ínamé[]   =   "img3.yahoo.com";
$name[] = "xyzyx.yahoo.com";
function getip($name)

    $address = gethostbyname($name);
    if($address === $name)

        return (falsé) ;

    else

        { return ($address);

    I

while (üst ($k, $v) = each($name))

    if(Saddress = getip($v))

        print("<br>gethostbyname(\"" . $v

    else

        print ("<br>gethostbyname\""   .   $v   .   "\")   failed.");
        f H ' i i x A
```

A következő eredményt kaptuk:

```
gethostbyname("img3.yahoo.com"):216.115.108.243
gethostbyname"xyzyx.yahoo.com")failed.
```

A getip()-struktúra más gethost, és hasonló utasításokra is alkalmazható annak érdekében, hogy szükség esetén hamis értéket eredményezzenek. Néhány más utasítás érvényes érték-

ként O-t ad vissza, és hiba esetén -1-et, ami a PHP3-ban hasznos, de a PHP4-ben szükségtelen, mivel a -1 hamissal is felcserélhető, és a PHP4 az új összehasonlító operátor, a = = = segítségével meg tudja különböztetni a 0-t a hamistól.

Host-címek listázása név alapján

Ha nem egy szerverhez, hanem egy domain névhez vagy egy elosztott szerverhez keresed az IP-címet, akkor gyakran egynél több IP-címet kapsz eredményül. A gethostbynamel()-utasítással megkaphatod az ilyen IP-címek listáját. A gethostbynamel()-utasítás egy host-nevet vagy egy domain nevet fogad el inputként, és az IP-címek tömbjét adja eredményül. Az utasítás kudarc esetén hamisat eredményez, ami rendszerint sokkal hasznosabb, mint a gethostbynameQ eredménye. Ez elég ok arra, hogy akkor is előnyben részesítsem a gethostbyname()-utasítással szemben, ha csak egy IP-címre számítok.

A következő kód három nevet dolgoz fel, kettőt az előző példából, plusz egy olyan domaint, ami biztosan több IP címet eredményez, a yahoo.com-ot:

```
set_time_limit(100);
$site[] = "img3.yahoo.com";
$site[] = "xyzyx.yahoo.com";
$site[] = "yahoo.com";
while(list($k, $v) = each($site))

    if($list = gethostbynamel($v) )

        while (üst ($ks, $vs) = each($list))

            print("<br>gethostbynamel(\"" . $v . "\"): " . $vs) ;

        }
    else print("<br>gethostbynamel\"" . $v . "\") failed.");
        {
```

A kód hasonló a gethostbyname()-utasításhoz használt kódhoz, de ki kellett egészíteni, mivel a gethostbynamel() outputja egy tömb. A plusz ciklus miatt a szkript túllépte az időkorlátot, ezért a set_time_limit()-paranccsal megnöveltem a rendelkezésére álló időt.

Az itt látható eredményben két IP-cím szerepel a yahoo.com-hoz. Figyeld meg, hogy a kudarcot eredményező keresés az input helyett hamisat adott vissza:

```
gethostbynamel("img3.yahoo.com") :216.115.108.243
gethostbynamel"xyzyx.yahoo.com")failed.
gethostbynamel("yahoo.com") :216.115.108.243
gethostbynamelCyahoo.com") : 216 .115 .108 . 245
```

Protokollszámok felsorolása

Ha szeretnéd tudni egy protokoll számát, egyszerűen futtass egy oldalt a következő kóddal. A kód 0-tól 200-ig végigmegy a számokon, és minden protokollnevet megjelenít, amit megtalál:

```
for($i = 0; $i < 200; $i++)
    if($n = getprotobynumber($i))
        \
        prifit("<br>" . $n);
        $i
```

Az eredmény itt látható:

```
    icmp
    igmp
    ggp
    ipencap
    st
    tcp
    egp
12: pup
17: udp
20: hmp
22: xns-idp
27: rdp
29: iso-tp4
36: xtp
37: ddp
39: idpr-cmtp
41: ipv6
43: ipv6-route
44: ipv6-frag
50: ipv6-crypt
51: ipv6-auth
58: ipv6-icmp

60:
73:
81:
89:
94:
59: ipv6-nonxt
    ipv6-opts
    rspf
    vmtp
    ospf
    ipip
98:  encap
```

Ennek az ellenkezőjét is megteheted, vagyis a protokoll nevéhez is megkeresheted a számot a következő kóddal. Ez a példa egy protokollnevet kis- és nagybetűkkel is ellenőriz, hogy meggyőződhessünk arról, hogy mindkettő működik:

```
$protocol[] = "tcp";
$protocol[] = "TCP";
```

($protocol))

```
{
if($n = getprotobyname($v))
    {                                                    JI
lJ,u print ("<br>getprotobyname(" . $v ." ):". $n) ;  !óJ

elsi

-i-f-^      print ("<br>getprotobyname ("    .   $v   .   ")    failed.");
```

Itt következik az eredmény, amely azt mutatja, hogy a getprotobyname()-utasítás nem különbözteti meg a kis- és nagybetűket: -ty''1*1'

```
getprotobyname(tcp):6
getprotobyname (TCP) : 6                    ' '='-;nút-'
```

Adatok besorolása WDDX-szel

A WDDX utasítások két megközelítést kínálnak az adatok besorolására: az egylépcsős feldolgozást és a háromlépcsős csomag építést. Az egylépcsős feldolgozáshoz egyaránt használhatóak a wddx_serialize_value() és a wddx_serialize_vars() parancsok. Ha egy nagy adathalmazt akarsz egy sztringbe besorolni, akkor a három lépcsős feldolgozást használd: hozz létre egy WDDX csomagot a wddx_packet_start() utasítással, adj hozzá változókat a csomaghoz a wddx_add_vars() utasítással és a sztringet létrehozva zárd be a csomagot a wddxjpacket_end() utasítással.

;=v

qfcb

wddx_serialize_value()

A következő kód besorol egy sztringet, egy egész számot és egy lebegőpontos számot, amelyek különböző mezőtípusok a PHP-ben. A besorolás után mindhármat megjelenítjük. A printutasításban az eredménysztringet a htmlentities()-parancsba ágyazzuk, mert az eredménysztring XML-t tartalmaz, és a böngésző megpróbálná értelmezni az XML tag-eket:

```
$value[]  =  "a  string  \"  <>";          ^j,  ^   ,ju....   .,s.|u-.--
$value[]  =  99;                           .          '   " ^ ,, ^.,'/' .....
$value[]  =  99.99;                        z)'\'1 — — K -v1, u'^ '■■''          li  1 t
while (üst ($k,  $v)  =  each ($value) )   .,1 j   ■)',■:
                                                              m;

    $string  =  wddxserializevalue($v);
    print("<br>"  .  $v  .                                                  qe
    "<br>    " .
    htmlentities($string)) ;
```

Az itt látható eredmény azt szemlélteti, hogy a WDDX nagyon sok tag-et hoz létre, és mindkét számtípust ugyanúgy kezeli. Mikor egy szkripted visszafejt egy WDDX-sztringet, a PHP-nek ki kell találnia, hogy milyen számok vannak eltárolva a <number> és </number> tag-ek között:

```
a string <>
    <wddxPacket version=' 1. 0 ' ><header/Xdata><string>a string
    " &lt; &gt;</stringx/data></wddxPacket> 99
    <wddxPacket version= '1.0' ><header/xdata><number>99</nuraber>
    </data></wddxPacket>
99.99
    <wddxPacket version='1.0'><header/xdata><number>99.99</number>
    </data></wddxPacket>
```

wddx_deserialize()

Hogyan kapod vissza az adatokat a **wddx_serialize_value()** eredményéből? Add hozzá a következő kódot, amely visszafejti az utolsó wddx_serialize_value() által létrehozott sztringet:

```
print("<br>".htmlentities(wddx_deserialize($string))};
```

Az itt látható eredmény a **Sstríng-változó** utolsó értéke:

```
99.99
```

serializeQ

Felér-e a PHP serialize-utasítása a WDDX hasonló utasításával? A következő kód ugyanazt az adattömböt a PHP **serializeQ** utasításával kódolja be XML-be:

```
reset ($value) ;
while (üst ($k,    $v)    =    each ($value) )
    {
    $string = serialize($v);
    print("<br>" . $v . "<br> Snbsp; " .
        htmlentities($string));
```

A következő eredmény a PHP serialize()-utasításának két előnyét mutatja: az eredményként kapott sztring kevesebb helyet foglal, és az utasítás különbséget tesz az egész számok - i: - és a lebegőpontos számok (double) - d: - között. Ha csak a PHP-n belül tervezel besorolást és kisorolást, akkor jobban jársz a PHP utasításaival:

```
a    string    "    <>
    s:13:"a                <>
    string
99
    i:99;
99.99
    d:99.99;
```

wddx_serialize_vars()

A wddx_serialize_vars() egy dologban különbözik a wddx_serialize_values()-utasítástól: a wddx_serialize_vars()-utasítás a változó nevét is bekódolja, vagyis a visszafejtő utasítás a változó tartalma helyett magát a változót állítja vissza. A következő kód egy változót sorol 5e, a korábbi WDDX példákból származó sztringet:

```
$a ="a string \"<>";                    "***":  r<l
$string =wddx_serialize_vars("a");
print("<br>".htmlentities($string));
```

Vedd észre, hogy a változó nevét sztringként kell átadnod. Ha ehelyett a változót adnád meg - wddx_serialize_vars($a) -, akkor az utasítás a $a-ban lévő értéket próbálná figyelembe venni, és ezt az értéket próbálná változó névként használni. Nem találna ilyen nevű változót, nem kódolna be semmit és nem eredményezne hibát sem.

Az eredmény:

```
<wddxPacket version=' 1. 0 ' xheader/xdataxstructxvar name= ' a ' >
<string>a string "&lt; &gt;
</stringX/varx/structx/datax/wddxPacket>
```

Az eredmény ugyanazt az értéket mutatja <string> tag-ben, mint a wddx_serialize_value() példában, de itt a <string> tag a <var> tag-be, az pedig a <struct> tag-be van ágyazva. A következő tesztben több változó szerepel, köztük egy tömb. A wddx_serialize_vars() tetszőleges számú paramétert elfogad, vagyis az összes szükséges változót egy szrtingbe kódolhatod:

```
Sb =99;                                   ;ibhs^'1H[1]I i. ,í
$c =99'.99;
$value =array ($b, $c) ;                          ,vj .^j ,,.
$string =wddx_serialize_vars("b","c","value");
print ("<br>".htmlentities($string));
```

A következő eredményben egy **<struct>** tag van, amely számos **<var>** tag-et tartalmaz. A tömböt tartalmazó <var> tag-ben van egy **<array>** tag, amely tömbként határozza meg a változót:

```
<wddxPacket version= ' 1. 0 ' xheader/xdataXstructXvar name='b'>
<number>99</number></varXvar name='c'><number>99.99</number></var>
<var name='value'xarray length='2'><number>99</number>
<number>99. 99</number></arrayX/var></structx/datax/wddxPacket>
```

;"<:- " pilisi3 *t*

wddxdeserializeQ változókkal

Hogyan kapod vissza az adatokat a wddx_serialize_vars()-ból? Add hozzá következő kó dot az előző példa visszafejtéséhez: v

```
var_dump (wddx_deserialize ($string) ) ;      rjs> .,...,.. ,  •  Vy.   ,i
```

Az itt látható eredmény azt szemlélteti, hogy a **wddx_deserialize()**-utasítás egy tömböt eredményez, amelv az eredeti változókat tartalmazza:

```
array(3) {["b"]=>int (99) ["c"]=>float(99.99) ["value"]=>array(2)
    {[0   ]=>int(99)[1   ]=>float(99.99)}}
```

Szükséged lehet olyan kódra, amely az eredeti változókat külön változókba különíti el. A wddx_serialize_vars() nyilván arra a legalkalmasabb, hogy egy sima változót vagy egy tömböt besorolj, amit el tudsz nevezni a kisoroláskor.

^ 98efnö:..~.

A

wddx_packet_start()

t, .,; . . .

Ha egy nagy adathalmazt szeretnél besorolni, hozz létre egy WDDX-csomagot a wddx_packet_start()-utasítással, lásd itt:

```
$packet =wddx_packet_start("Another test");
```

Az egyedüli paraméter egy opcionális megjegyzés, amit beilleszt a csomagba. Az eredmény egy erőforrás-azonosító, a csomagazonosító, amelyet a wddx_add_vars() és wddx_packet_end()-utasításokban használhatsz a csomag azonosítására.

wddx_add_vars()

Add hozzá az összes változót egy WDDX-csomaghoz a wddx_add_vars()-utasítással, mint itt látható:

```
$a   ="a   string   \"<>";          >dnöliid dsdÁo,ima 3
wddx_add_vars($packet,"a");
$b = 99;
$c =99.99;
$value =array(5b,$c);
wddx_add_vars(Spacket,"b","c","value");
```

Az első paraméter a wddx_packet_start()-ból származó csomagazonosító, ezután a wddx_serialize__vars()-utasításhoz hasonlóan tetszőleges számú változónevet megadhatsz.

wddx_packet_end()

,·hw ^,AmA,;j „ ^

Ha befejezted a változók hozzáadását a WDDX-csomaghoz, akkor a wddx_packet_end()-utasítással konvertálhatod a csomagot egy XML-be kódolt sztringgé, lásd a következő példát (a példakód meg is jeleníti a sztringet, vagyis az eredményt is láthatod):

```
$string   =wddx_packet_end($packet);
print("<br>".htmlentities (Sstring));
```

A következő példa a wddx_packet_end()-ből származó sztring, amely azt szemlélteti, hogy a csomagépítő eljárás ugyanarra az eredményre vezet, mint hogyha az összes változót egyetlen wddx_serialize_vars()-utasításba gyűjtötted volna:

```
<wddxPacket   version='1.0'><header><comment>Another   test</comment>
</headerxdata><structxvar   name=' a ' >
<string>a   string   "&amp;lt;&amp;gt;</string></var><var   name='b'>
<number>9 9</number></var><var   name='c'><number>99.99</number></var>
```

```
<var name='value'Xarray length='2'><number>99</number>
<number>99.9 9</number></array></var></structx/data></wddxPacket>
```

Az egyetlen plusz szolgáltatás az opcionális megjegyzés; a wddx_deserialize() nem adja vissza a magjegyzést, vagyis a tesztelésen kívül csekély haszna van.

Adatok tömörítése zlib használatával

A gz-utasítások magukban foglalják mind a fájlfüggvényeket, mind a nyers adatfüggvényeket. Ez a megoldás a nyers adatfüggvényektől indul, vagyis egy üzenetbe vagy adatbázisba történő beépítés céljából is tömöríthetsz. Később tárgyaljuk a teljes fájlok tömörítését, amivel csökkentheted a hálózaton továbbított fájlok méretét.

A következő kód vesz egy sztringet, betömöríti a gzcompressQ-utasítással, és megjeleníti az eredményt.

```
$compressed =gzcompress("Clarinet Quintet K.581 -lst Movement");
print("<br>".htmlentities($compressed));
```


Az itt látható eredmény alapján elképzelheted, mit jelent a tömörítési ráta, és hogyan használhatod a tömörített adatokat:

```
xoesÍI,ÉÍK-Q, Íi+Ö z | t ° tÁ% úe'©y ©y%
```

A tömörített sztring mérete nem sokban különbözik az eredetitől, aminek oka főképp a tömörítési többlet. Egy hosszú sztring betömörítése már megtakarítást eredményezne, mivel a tömörítés révén megtakarított hely nagyobb lenne, mint a tömörítési többlet. Azt is figyeld meg, hogy a karakterkészlet olyan karaktereket tartalmaz, amelyek nem mennének át e-mailben vagy hasonló átviteli eljárással. A tömörített sztringet base64 kódolással lehetne átküldeni, ami felemésztené a tömörítéssel megspórolt helyet. Annak bizonyítására, hogy a tömörítés könnyen visszacsinálható, itt van a $compressed-sztringet kitömörítő kód:

```
$uncompressed =gzuncompress(Scompressed);
print ( "<br>" . htmlentities ($uncompressed) ) ; ^...íÁÜLÍY im.iL ist;
```

Itt látható a kitömörítés eredménye, egy sztring, amely megegyezik az eredeti teszsztringgel:

```
Clarinet Quintet K.581 -lst Movement
```

A zhb a deflációt is támogatja, amely hasonló a tömörítéshez, de nem olyan megbízható, mint ahogy ezt a következő gyors teszt is mutatja: ; asrjj?f"i' TTJ-1 . "<

```
$deflated =gzdeflaté("Clarinet Quintet K.581 -lst Movement"); X'r:
print ("<br>" .htmlentities ($deflated) ) ;
```

A kód ugyanazt a sztringet deflálja, amit a gzcompress()-utasításhoz használtunk. A „ gzdeflate()-utasításnak egy opcionális második paraméter is megadható, amivel meghatározható a 0-tól (nincs tömörítés) 9-ig (maximális tömörítés) terjedő tömörítési szint. A magasabb szintű tömörítés több processzoridőt használ.

Az itt bemutatott eredmény hasonló a gzcompress()-ből nyert tömörített sztringhez:

SÍI,ÉÍK-Q,íí+Ö_zIt ° tÁ%_úe ©y ©y%

A következő kódnak vissza kellene állítania a deflált sztrmget az eredeti formájába, és megjeleníteni az eredményt:

```
$inflated =gzinflaté($deflated) ;:
print("<br>".htmlentities($inflated));
```

A gzinflate() rendelkezik egy opcionális második paraméterrel, amellyel egy hosszt lehet megadni. Az utasítás figyelmeztető üzenetet eredményez, ha az adat hossza nagyobb a megadottnál. Ez feltehetőleg a túlcsordulás megakadályozását szolgálja, ami eredhet kitö-mörítési hibákból vagy olyan adatokból, amelyek nagyon jól összenyomhatok, és kitömö-rítve meghaladják a rendelkezésre álló memória méretét.

Az eredmény a következő hibaüzenet volt:

Warning:gzinflate:buffer error

A hiba a PHP 4.0.7 fejlesztői kiadásában fordult elő, vagyis mire megpróbálod a gzinflate() utasítást használni, már lehet, hogy kijavították a hibát.

A következő kód a zlib fájlfüggvényeit teszteli:

```
$data = "Clarinet Quintet  K.581 - lst Movement";
$gz = "./test.zip";
if($file   =   gzopen($gz,    "w"))
    {
    gzwrite ($file, $data) ;
    gzclose($file);
    if($file = gzopen($gz,    "r"))
       print("<br>File size: " . filesize(realpath($gz)));
       $text = gzread($file, 100000);
       print("<br>" . $gz . " contained:<br>" . htmlentities($text) );
       gzclose($file) ;

    else

       print("<br>gzopen failed for " . $gz );

    }
else

    print{"<br>gzopen failed for " . $gz);
```

A kód egy új tömörített fájlt hoz létre a gzopen()-utasítással, a gzwrite()-tal beírja az adatokat a fájlba, a gzclose()-zal bezárja, a gzopen()-nel megnyitja a fájlt olvasásra, a gzread()-del kiolvassa az adatokat, majd végül az eredetivel való összevetés céljára megjeleníti az adatokat. A kód egy filesize()-utasítást is tartalmaz, hogy a GZ-fájl méretét is össze lehessen hasonlítani az eredeti adatok hosszával.

```
Az eredmények: '■ ..... " ------ -i■-                    ■
File    size:58                                .              ._. - , , . ■
-^/feést.zip    contained: Clarinet
Quintet   K.581   -lst   Movement
```

A 37 bájtos tesztadat 58 bájtos fájlt eredményezett. A kiolvasott adat megegyezik a beírttal.

A 8. fejezetben olvashatsz a fájlfüggvényekről és a zlib-utasításokat egyszerűen behelyettesítheted a fájlfüggvények helyére. Azzal is kísérletezhetsz, hogy a zlib:-elő'tagot használod az útvonalakhoz; így a zlib-utasítások helyett használhatod a fájlfüggvényeket. Akárhogyan is használod, a zlib-tömörítéssel lecsökkentheted a fájlok méretét - különösen a nagy, szöveges alapú fájlok méretét - mielőtt továbbítanád őket a hálózaton.

Saját napló írása

Ez a megoldás abban az esetben hasznos, ha nincs hozzáférésed a rendszernaplóhoz, ami gyakran előfordul, ha a honlapod egy Internet-szolgáltatónál van elhelyezve. A következő kód azt feltételezi, hogy a t:/ meghajtón található log.txt fájlba naplózol, ami egy eldobható tesztpartíció a Windows NT munkaállomásomon:

```
function privatelog (Smessage)                    ,i dilst t b(>?5

    if(Sfile    =    fopen("t:/log.txt",                ff!
        "a")) {
        fwrite($file, $message . "\n");
        fclose($file) ;

    else (                          C.v.;ir. Kli
        return (falsé)

    }
```

A Windows-os gépeken rendszerint van egy c:\temporary files\ könyvtár és az Internet-szolgáltató Unix-os gazdaszerverén van egy könyvtár a weboldalaid számára valami hasonló névvel: /usr/home/petermoulding/.

A kód megnyit egy naplófájlt (és létrehoz egyet, ha a fájl nem létezik), hozzáfűz egy bejegyzést, és bezárja a fájlt, ezáltal garantálja, hogy a bejegyzés valóban beírásra kerül, akkor is, ha későbbi hiba miatt a szkript futása megszakad. Ha trükkös kódot tesztelsz, amitől elszállhat a szkripted, akkor tégy egy naplóbejegyzést minden kérdéses sor elé és mögé. A \n egy sortörést szúr a szövegbe, mikor egy szerkesztőprogrammal megnézed a fájlt. A formázást a megjelenítési igényeid szerint megváltoztathatod. Ha még tökéletesebbé akarod tenni a függvényt, akkor az üzenetet elláthatod egy előtaggal, amely a dátumot és a 2. fejezetben megismert microtime() által megadott időt tartalmazza.

Hivatkozás: **oldal:**

rand(), srand() és microtime() **62**

17. fejezet

ín , j . J -£ bo,

Objektumok

Gyors megoldások	oldal:
Objektumok mentése sessiönökben és a __ sleep() használata	585
Objektumfüggvények használata	**589**
call_user_method()	589
call_user_method_array()	590
class_exists()	591
get_class()	591
get_class_methods()	591
get_class_vars()	592
get_declared_classes()	592
get_object_vars()	593
get_parent_class()	593
is_subclass_of()	594
method_exists()	594
Honlap testreszabása objektumokkal	**594**
Hírcsoportok olvasása	**598**

■ ' ■ v , ^ - o ; .

uqu £ v loí

Áttekintés

A függvények után az objektumok jelentik a következő lépést, amelyek szintén használnak függvényeket, így nem árt elolvasnod a 10. fejezetet, mielőtt továbbmennél.

Az objektumok úgy vannak elkészítve, hogy a nagyobb fejlesztési projekteket fel lehessen darabolni kisebb projektcsoportokra, és a más projektekből származó kódok újra felhasználhatók legyenek. Ha kitalálsz egy nagyon jó naptárat egy oldalra, becsomagolhatod egy objektumba és felhasználhatod más oldalakon is.

A kedvenc példám az autóban használt mobiltelefon. Az autógyártók szabványos méretű nyílást terveznek a rádió és a CD-lejátszó számára a műszerfalba, ami a mobiltelefonok elhelyezésére nem alkalmas. Ha jó vételt és a kezeket nem igénylő használatot szeretnél a mindent behálózó zavaró kábelek nélkül, vásárolnod kell egy autós kihangosító felszerelést, azt be kell szereltetned, és az egészet eldobhatod, ha egy újabb telefont vásárolsz. Az objektumokra is ugyanez érvényes, csakhogy itt szoftverekről van szó.

Minden PHP-objektum a világ bármely PHP-szkriptjében használható, de minden naptárobjektum más *illesztőfelülettel* rendelkezik, így nem tudod azt egyszerűen kivenni és beilleszteni egy másikba. Az osztály egy objektum-prototípus, ami minden kódot tartalmaz egy objektum összeállításához, így az ugyanabba az osztályba tartozó objektumok azonos felülettel rendelkeznek, illetve a különböző osztályba tartozó objektumok különbözővel, még abban az esetben is, ha azonos függvény ellátására tervezték őket. Egy objektum felülete a tulajdonságok és módszerek összességét jelenti az objektumon belül. Egy objektumot valamely osztályból a new kulcsszó használatával, a *constructor* nevű osztályban egy speciális függvény használatával állíthatsz össze.

Osztályok

Amikor egy gyártó egy mobiltelefont készít, az a következő ütemezéssel kezdődik: „helyezze az A elemet oda," „helyezze a B elemet az A mellé," és „ragassza a C elemet az A és a B tetejére." Az *osztálynak* pontosan az az értelme, hogy az objektumnak megfelelő osztály tartalmaz minden elemet (statikus kódokat, amelyek az objektumokba kerülnek) és utasítást az objektumok felépítéséhez (dinamikus kódokat, amelyek módosítják az objektum részeit, miután az felépült).

A következő kód a legrövidebb osztály a világon. Semmit sem csinál, csak egy tárolót hoz létre a néven a kód számára. Ez az osztály egy kartondoboz, amit elemekkel és utasítások kal töltesz meg egy a típusú objektum felépítéséhez: . ,

```
class
a                                                    1; v
```

new

Az alábbiakban látható egy példa, hogyan használhatod az a osztályt egy objektum - jelen esetben a $b - létrehozására, a new speciális kulcsszó használatával.

```
$b = new a;
```

Mint egy változó

Ali shiny and new/Like tyvariablei'Constructed for the very first time (Madonna az objektum-orientáltprogramozásról énekel a Liké a variable című ismert dalában.)

Egy objektumot majdnem ugyanúgy kezelhetsz, mint egy változót: ugyanúgy összehasonlíthatsz objektumokat, mint bármely más PHP-forrást, még akkor is, ha az összehasonlításnak nincs értelme, illetve besorolhatod az objektumokat a session-rekordban való tároláshoz, habár az objektumok session-rekordban történő tárolása elég rossz ötlet.

Miért nincs gyakran értelme az objektumok összehasonlításának? Az egyik előnye az objektumoknak a változókkal szemben, hogy képesek többszörös változók kezelésére egyetlen objektumként, ahol minden változó egy tulajdonsággá válik. Az ilyen tulajdonságok összehasonlításának ugyanúgy van értelme, ahogy két bőrönd színét is össze tudod hasonlítani. Két objektum összehasonlítása már nehezebb csak úgy, mint a két bőrönd összehasonlítása egy olyan jellemző tulajdonságuk alapján, mint a tömegük. A legkönnyebb bőrönd tűnik a legmegfelelőbb választásnak, mivel könnyedén el tudod vinni, habár lehet, hogy üres. A legnehezebb bőrönd lehet, hogy ólommal van megtöltve, míg a közepesen nehéz bőrönd gyémántokkal lehet tele. Az objektumok használható összehasonlításához fel kell becsülni, hogy melyek a lényeges tulajdonságai.

Miért kockázatos az objektumok besorolása? A PHP elhagyja a kódokat, és besorol minden adatot, amikor átadsz egy objektumot a serialize()-függvénynek vagy bejegyzed azt egy PHP session-be. Az objektum 20 byte használható információt és 300.000 byte munkaváltozót tartalmazhat, amire egy session-rekordban semmi szükség. Ha az objektum úgy van kódolva, hogy felismerje a besorolást és hagyja el a 300.000 byte fölösleget, biztonságosan besorolhatod az objektumot, illetve kiemelheted belőle a kívánt értéket, és besorolhatod azt az objektum nélkül. A PHP új adottsága, hogy lehetőséget ad az objektumok optimalizálására a besoroláshoz, ezért kevés „kereskedelemben kapható" osztály fog optimalizáló objektumot létrehozni.

A következő kód létrehozza a Sc-objektumot, ugyanabban az osztályban, amiben a $b is van, összehasonlítja $c-t Sb-vel, majd kiadja azt az üzenetet, hogy c megegyezik b-vel.

```
Se = new a; if($c
=== $b)
    { print("<br>c equals
b.");
    }
else
    { print("<br>c does not equal
b.");
```

A valóságban két ugyanabba az osztályba tartozó objektum létrehozása nem feltétlenül azonos objektumokat eredményez, mivel az objektumok különböző módon konfigurálhat ják magukat a létrehozás során. Két ugyanabba az osztályba tartozó objektum közti kü lönbség megtalálásához meg kell nézned a *konstruktőr-kódjaikat,* amiről a fejezet „Konstruktőr" részében beszélünk.

Egy objektumból nem hozhatsz létre másik objektumot a new parancs alkalmazásával. A következő kód nem működik:

```
$d = new $c;
```

Az alábbi hibaüzenet jelenik meg a parancsot követően:

```
Warning: Object to string conversion
Fatál error: Cannot instantiate non-existent class: object
```

Az objektumokat másolhatod, illetve törölheted, ugyanúgy, mint a normál változókat, ez a következő két kódsorban látható:

```
$d = $c;
unset ($c);
```

Amikor egy objektumot másolsz, az objektum egy olyan másolatát kapod, ahogyan az a létrehozás pillanatában létezik. Ez rendszerint nem ugyanaz, mint létrehozni egy újat a new-paranccsal. Egy objektum készítése a new-paranccsal, más néven *létrehozás,* amennyiben az objektumorientáltak táborába tartozol, létrehoz egy új változó sorozatot az objektumban, illetve megváltoztathatja a változókat vagy a kódot, alapul véve a dátumot, az időt vagy egy meglévő azonos típusú objektumot.

StdCIaSS

Az stdCIass egy speciálisosztály meghatározás a PHP4 számára fenntartva, ezért ne használd az stdClass-t a saját kódodban.

A PHP4-ben a dupla aláhúzás (_ _) speciális függvényelőtagnévként van fenntartva. Két speciális függvény, a__ sleep(), és a __wakeupQ már használatban vannak, és a fejlesztők fenntartják a jogot, hogy __ előtaggal bármikor létrehozhassanak újabb függvényeket.

__sleep

A _sleep() egy speciális függvény a PHP4-ben, a besorolás segítésére. De miért használnád ezt, amikor a PHP rendelkezik a besorolás- (serialize()-) függvénnyel? A besoroló függvény (1. 16. fejezet) a dolgok session-rekordban történő elmentésére szolgál, de a session-rekord nem a legmegfelelőbb hely az objektumok számára. Ha megakadsz egy szoftverrel, ami egy objektumot kér a session-rekordba történő belépéshez, használhatod a _ sleepQ-függvényt az objektum osztályán belül a besorolás biztonságosabbá tételére.

Amikor a serialize()-függvény egy objektum besorolására készül, megkeresi és végrehajtja a __sleep()-függvényt az objektumban. A__ sleep()-függvény be tudja zárni a fájlokat és az adatbázis-kapcsolatokat, illetve visszaállítja a besorolásra szánt változók elrendezését,

így a __ sleep()-függvény végrehajtása során az azonosítatlan dolgok sem fognak helyet el foglalni a besoroló sztringben.

Ha a szknpt befejezését követően a besorolás-parancs automatikusan lefut, ahogy ez alap értelmezetten történik a programfolyamat során, a __ sleep()-függvény túl későn fog futni a print()-függvény használatához vagy a képernyőn történő megjelenítéshez; ezért minden üzenetet a naplókon keresztül vagy egyéb módon kell továbbítanod. A 7. fejezetben bemutatom, hogyan kell naplózni a hibákat a print()-függvény használata nélkül.

Az objektumokban összevissza található nagyméretű változók sem éppen a legszerencsésebbek. A változókat az unset()-paranccsal fel kell oldani, ha nincsenek már használatban. Ha egy adatbázisból beolvasol egy sorkészletet egy tömbbe, és több műveletet elvégzel a tömbön az oldal elemeinek felépítéséhez, használd az unsetQ-függvényt a tömbre, közvetlenül az utolsó művelet elvégzése után. Az objektum nem fogja tudni, hogy a szkript további része mikor végez az adatokkal. Például abban az esetben, ha a tömböt több folyamat használja és a szknpt bármilyen módon használhatja a folyamatokat.Ebben az esetben el akadsz a __ sleep()-függvény létrehozásával, hogy törölhesd a tömböt, mielőtt az objektumot besorolhatnád. Szükséged lehet a __wakeup()-parancsra is, hogy újra létrehozhasd a tömböt, amikor az objektum még nincs besorolva.

Találsz egy példát a __ sleepQ-függvény használatára az Gyors megoldások - „Objektumok mentése session-ökben és a __sleep()-függvény használata" - részben.

__ wakeup

A __wakeup()-függvény egy speciális függvény, amely a PHP4-ben a visszaállított objektum session-rekordból történő visszanyeréséhez van fenntartva. Amikor az uíiserializeQ-függvény rendszerint egy session-rekordból kisorolja az objektumot, az objektumnak a benne lévő adatok újbóli létrehozásához vagy frissítéséhez esetleg fájlokat kell beolvasnia,
vagy SQL-parancsokat kell lefuttatnia adatbázisokon. Erre a feladatra a___ wakeup() a megfelelő függvény. Amikor az unserialize() megszünteti az objektum besorolását, az objektumban az unserialize()-függvény helyett a __ wakeup() jelenik meg és hajtja végre a függvényt.

Tegyük fel, hogy az objektumod képes megjeleníteni egy honlapon a valutaárfolyamokat. A látogató minden oldalon az aktuális valutaárfolyamot keresi, de a szkriptet író személy az objektumot egyszerűen egy műveleti változóként regisztrálja ahelyett, hogy minden egyes oldalon az ideiglenes tárolóból kellene létrehoznia. így az objektumra marad a legfrissebb változások beolvasása minden új oldalra, ami azt jelenti, hogy minden alkalommal, amikor az objektum besorolását megszüntetjük, beolvassa a változásokat.

Én személy szerint nem regisztrálnám az objektumot. Ha az objektum műveleti változóként regisztrálva van, minden oldalon besorolásra kerül, illetve megszűnik a besorolás, még akkor is, ha nincs használatban. Ha a honlapodon a frissítések csak egy oldalon jelennek meg, és a látogatók több lap közül csak néhányszor nézik meg ezt az oldalt, az objektum besorolásának megszüntetése sokkal többször megtörténik, mint ahányszor az oldalon lévő adatokat használják.

Sl-oi
Sl-re <j; * '

Egy másik megközelítés az objektum besorolása, de az adatok kihagyása a___ sleep()- és a __wakeup()-parancsokból. Csak akkor nyerjük vissza az adatokat, amikor a függvény először kéri. Amennyiben a valutaárfolyam-tömböt több függvény használja, ellenőriztesd a függvénnyel a tömb meglétét, és akkor nyerd vissza az árfolyamadatokat, ha a tömb hi ányzik. Az adatok visszanyerésének elhalasztásával, az adatvisszanyerés nem fog megtör ténni mindaddig, amíg nem szükséges. Bár a___sleepQ-paranccsal van értelme a használa ton kívüli adatok eltávolításának, az adatok automatikus frissítését a___ wakeup()-függvény el tudja végezni az előbbi fölösleges folyamat nélkül.

Még egy dolognak kell eszedbe jutnia az objektumok besorolásával kapcsolatban: a serialize()-függvény tárolja az objektumból származó adatokat, illetve az osztályok neveit, de az objektum kódjait nem. Amikor az unserialize()-függvény kisorolja az objektumot, az unserialize() megpróbálja kiolvasni az objektum kódját az osztály definícióból, így a besorolás megszüntetése előtt a szkriptnek tartalmaznia kell az osztály meghatározását. Mivel a legalkalmasabb hely a műveleti információk visszaállítására éppen a szkript elején van, legjobb, ha az osztály definíciót a szkript eleje tartalmazza a requireQ-függvényen keresztül. Ha az osztályban lévő kód megváltozik, de más meglévő objektum még használja, a szkript lefagyhat. Ez egy másik érv arra, hogy miért ne tárolj objektumokat session-rekordban.

Tulajdonságok

Az objektum tulajdonsága egy érték, amit úgy érhetsz el, ha leírod az objektum nevét, egy kötőjelet és egy nagyobb jelet (->), majd a tulajdonság megnevezését. Ha a $b-objektum rendelkezik hosszúsággal (length), akkor a tulajdonság neve $b->length, és ezt egy változóként kell kezelned, tehát olvashatod, illetve beállíthatod. A kód következő sora megpróbálja megjeleníteni a korábban létrehozott $b-objektumból a length-tulajdonságot, habár a $b-objektumban nincsenek tulajdonságok.

```
- print("
                $b->length);
```

Az eredmény az alábbi hiba lesz:

```
Warning:  Undefined  property:  length                    iiritVíSE
```

Mi történik, ha hozzáadod a $a közepéhez a Slength = 29-et? A következő hibaüzenetet kapod, mutatva, hogy a nyers PHP-kód nem ugyanúgy működik az osztálydefiníciókban. Ha megpróbálod a $length-t, ugyanazt a hibát kapod:

```
                                        T OLD FUNCTION' or 'T FUNCTION'
Parse  error:  parse  error,
expecting or  'T  VAR'  or  "}"
```

var

A következőkben bemutatjuk egy tulajdonság meghatározásához a megfelelő kódot egy osztályban, kiegészítve a var kulcsszóval, jelezve, hogy egy változót deklarálsz. A var kulcsszó helyet hoz létre a változó számára, és a változó érték hozzáadásával egy tulajdonságot készít, amelyet meg tudsz jeleníteni:

class a ' i ','■ fúA E y-narrtbai;.' \A
.Q

var $Iength = 29; ■w;.v;íit[1]:;: i-.. 11 '■■;■;:..: jlr.Tib'.'.;': C i l i : 1 K e-
nA

Konstruktőr

A PHP4-ben egy tulajdonság var-függvénnyel történő meghatározása csak konstansokkal lehetséges. A következő lépés a *konstruktőr* nevű függvény hozzáadása, ami azáltal van azonosítva, hogy ugyanaz a neve, mint az osztálynak. A konstruktőr függvény bármit el tud végezni a változókon, és ezt akkor hajtja végre, amikor egy osztályból létrehozol egy objektumot, ami azt jelenti, hogy az egy osztályból létrehozott objektum különbözhet az ugyanabból az osztályból létrehozott bármely egyéb objektumtól, J-JVŐJ! *z ynbrnbsiss* sA

A PHP4-konstruktorok másképpen viselkednek, mint a PHP3-konstruktorok, de ez a fejezet a PHP4-ről szól. A legjobb letölthető, előre megírt, nyitott PHP-forráskódok PHP4-kompatibilisek, így nem valószínű, hogy problémáid lesznek a PHP3-kódokkal.

A PHP4-ben nincsenek megsemmisítő (destruktor) függvények. Ha a konstruktőr megnyit egy fájlt írásra, szükséged lesz egy illeszkedő megsemmisítő függvényre is a fájl bezárásához. Adhatsz az osztályhoz egy destruktor függvényt, és manuálisan behívhatod az objektum megsemmisítése előtt. Ha szeretnél egy automatikus megsemmisítő függvényt, létrehozhatod és regisztrálhatod, mint bezáró parancsot ezt mutatjuk be a „Módszerek" részben.

$this

Egy osztályon belül a Sthis hivatkozást jelent az aktuális objektumra. A $this->x egy hivatkozás az objektumon belül az x nevű változóra. A $this használata egy egyszerű problémát vet fel: aki az osztályt írja, nem tudja, hogy mi lesz az objektum neve az osztályból történő létrehozáskor. A $this az a speciális név, amely az objektumon belül állandó marad, az objektum nevére való tekintet nélkül.

A következő kód definiálja a k-osztályt, a $start-változót, és a k-konstruktorfüggvényt a k-osztályon belül:

```
class k. {
    var $start;
    function k()

        $m = explode(" ", microtime());
        $this->start = $m[l] . substr($m[0],
        1)
```

,üfrT

```
$1 =new
k;                $l->start
print("<br>"
```

A k-függvényen belül a Sthis egy hivatkozás a k-osztályon belül létrehozott objektumra. A Sthis->start hivatkozás a start nevű változóra az objektumban. A kód ezután létrehozza a Sl-objektumot a k-osztályban, és megjeleníti a $l->start-ot. A $l-ban a $this hivatkozás a Sl-re és a $this->start a $l->start-t jelenti.

Az eredmény a következő: "U-

```
995012077.37157100
```

A **microtime()** a 2. fejezetben van leírva, amely ugyanazt adja vissza, mint a **time(),** plusz az időt mikromásodpercekben. Az eredmény 1970. január 1-től kezdve ábrázolja a másodpercek és mikromásodpercek számát. A **$l->start** használata segíthet a szkript idejének vagy az időkódnak a beállításában a k-osztályból létrehozott objektumban,

A **tulajdonságok** neveiben meg kell különböztetni a kis- és nagybetűket. A következő kód megpróbálja elérni a Sl->start-t a $l-> start névvel:

```
print ("<br>"  .  $l->Start);
```

Az eredmény a következő hibaüzenet:

```
Warning:  Undefined  property:  Start                    ~)
(Figyelem:  Meghatározatlan  tulajdonság:  Start)
```

Módszerek

A módszerek függvények egy osztályban, amelyek műveleteket hajtanak végre egy objektumban. Minden, osztályban meghatározott függvény elérhető módszerként, de figyelembe kell venned, hogy mi történik, ha valaki véletlenszerűen használ egy módszert.

A következő osztály a korábbi k-osztilynak egy kiterjesztése. Ez a változat tartalmazza a start()-függvényt a Sthis->start megjelenítésére. A **start()** ezután minden, a k-osztályban létrehozott objektum módszerévé válik:

```
class  k
    í
    var                                                    A r
    $start;                                                oLv/
    function
    k ( )

        $m  =  explodeC  ",  microtime () ) ;              cinibb böiJ
        $this->start  =  $m [ 1 ]  .  substr ( $m [0 ] ,  1);      •     ibc:

    function  start()

        print("<br>Start:        $this->start);
```

A következő teszt létrehozza a $q-objektumot a k-osztályból, és végrehajtja a **$q->start()** módszert. Vedd észre, hogy a módszereket a tulajdonságoktól a módszer neve után található () különbözteti meg, és hogy a módszerek valójában közönséges PHP-függvények kicsit bonyolultabb elnevezéssel:

```
$q  =  new  k;
$q->start ();
```

Az eredmény a következő: *:<-tv.v*

```
Start:
995085968.38211500
```

A

A módszerek neveiben, a függvénynevekhez hasonlóan, nem kell megkülönböztetni a kis- és nagybetűket. A következő sor a név betűtípusának változását teszteli: ,

```
$q->Start<);
```

Az eredmény a következő: ii..U; \; • íj. a mtvk-vőn !«-\.i<;v.í.

```
Start:
995085968.38211500
```

Annak bemutatására, hogy az osztály módszerek éppen olyanok, mint bármely egyéb függvény, cseréld ki a **function start()-ot** az előző osztályból a **function** start-ra: ;, í6~, .Jfj.

```
function    start ($text)                    .    .           ,
                                                           ÍT
        print("<br>"   .   Stext
        .                                   $this->start);
```

Próbáld ki a következő kóddal történő változást: *Lsáoig") :.sva*

```
$q  =  new  k;
$q->start("Time");
```

íme a változás:

```
Time:    995085968.38211500
```

Inotiri LÓJI c

A módszerek ugyanúgy működnek, mint bármely egyéb függvény, és a többihez hasonlóan globális utasításokat igényelnek a külső változók eléréséhez, egy kivétellel: a külső változókat ugyanazon az objektumon belül a **$this->**-függvényen keresztül lehet elérni.

Megsemmisítő

· ,q ,·n· De

A PHP-ban nincs megsemmisítő (destruktor) függvény, ezért az osztályhoz hozzá kell adnod egy destruktor függvényt, amelyet módszerként kell használnod az objektum megsemmisítése előtt. A következő p-osztály tartalmaz egy p-konstruktort és egy **destroy()-**destruktort, amely kiszámítja és megjeleníti a művelet kezdete óta eltelt időt:

```
class   p                                                IM

    var    $start;
    function    p()

        $m  =   explode("   ",   microtime());
        $this->start   =   $m[l]    .   substr($m[0],
        1);

                function    destroy()           » Hull/,'         ob

        $m = explode(" ", microtime()); $end
        = $m[l] . substr($m[0], 1); $duration
        = $end - $this->start;                         iü:\t.
        print("<br>Duration: " . Sduration)
                                                          rrur,,
```

! .. A- t,

A következő kód definiálja a $q-objektumot a p-osztályból, végrehajtja a $q-> destroy ()-függvényt, illetve feloldja a $q-objektumot. Ez a manuális megsemmisítés a legmegfelelőbb választás a PHP-ben, ha manuálisan szeretnél egy függvényt megsemmisíteni egy destruktor függvénnyel:

```
$q = new p; $q->destroy unset($q) ;
```

Slarr a,: Z.U:JÍ
iiv b !-.■:'

Az eredmény a következő (és csak hat tizedes jegy pontosságú, ami a lebegőpontos aritmetikából fakad): !ij.«n st

```
Duration: 0.00010406970977783
```

*•> M> h>(\nut.

Ha szeretnéd a megsemmisítési folyamatot egy kicsit automatizálni, kifejleszthetsz egy destroyQ-függvényt a következő módon: '•" • íxe-rí . "<■: .1 ;>") rüittf

```
function destroy($object)

    eval("global \$" .
    $object
    ${$object}->destroy() ;
    unset($GL0BALS[$object] );
```

C--»9 JÍJ3V6 JÍ í

A függvényben a kód három sora egy globális utasításon keresztül éri el az objektumot, végrehajtja benne a destroy()-függvényt, majd feloldja az objektumot. A ${$object} egy alternatív jelölés egy eval()-függvény módszer végrehajtási célból történő írásához. A SGLOBALS feloldására az unset()-függvényen belüli használatakor van szükség, mivel az objektum vagy egy globális változó azonnali feloldása nem használható a függvényen belül.

A tesztkód a következő:

```
5q = new p; *;     *  .   ;   ;
destroy("q");
```

A kód létre hozza a $q-objektumot, majd megsemmisíti a destroy()-függvénnyel. Az eredmény egy megjelenő sor lesz, ami az időtartamot mutatja.

Mi van akkor, ha azt szeretnéd, hogy a destruktor automatikusan fusson le a szkript végén manuális beavatkozás nélkül? Létezik egy működőképes megoldás, amennyiben a destruktorod nem használja a print()-függvényt, vagy az echo-t. A megoldás neve register_shutdown_function(). A leállítási függvények a szkript végén futnak, miután az utolsó outputot is elküldheted a böngészőnek, illetve közvetlenül az összes változó megsemmisítése előtt. A leállítási függvényeket használhatod a fájlok bezárására, tranzakciók befejezésére, üzenetek fájlokba vagy adatbázisokba történő naplózására és minden olyan dolog megszüntetésére, amit a PHP egyébként nem tenne meg. A következő kód megpróbálja a Sq->destroy()-t regisztrálni. A $q->destroy() a leállításkor fut le, de nem fog befejeződni, mivel a függvényt megpróbálja megjeleníteni:

```
register_shutdown_function("q->destroy");
```

..",o t rőtíL',]'

Osztályok kiterjesztése

Mi van akkor, ha egy meglévő osztályt akarsz használni, de csak egy újabb módszert szeretnél hozzáadni, vagy egy kisebb változtatást végrehajtani rajta? Amennyiben nincs ezen az osztályon alapuló meglévő fájl, megváltoztathatod az osztályt. Ha a meglévő osztály használatban van a weboldaladon vagy valahol máshol, a legbiztonságosabb megoldás az, ha az eredeti osztályt érintetlenül hagyod, és definiálsz egy új, a meglévőn alapuló osztályt. Ez a megközelítés akkor működik megfelelően, ha a meglévő osztályt más biztosítja, aki frissítésekkel is ellát, mivel ezeket az osztályokat anélkül kicserélheted, hogy újra alkalmaznod kellene a változtatásaidat.

Függvények hozzáadása

Kezdjük a fejezetben már használt p-osztállyal. Az osztály egy többletmódszerrel lesz kiegészítve, ami visszajuttatja a $start-ot a formázáshoz:

```
class p

    var $start;
    function p(

        $m = explode(" ", microtime());
        $this->start = $ra[l] . substr($m[0] ,
        1);
```

Az alábbi kód kiterjeszti a p-osztályt q-osztályra, és hozzáadja a **start_time()**-függvényt, amely megjeleníti az indítási időt óra, perc, másodperc formátumban:

```
class q extends p

    function start time()

        $m = explode(" ",
        $this->start); print("<br>" .
        date("H;i:s", $m[0]
                                        • $m[l] );
```

Próbáld ki a következő kóddal kiterjesztett osztályt, hogy létrehozz az osztállyal egy $r-objektumot, majd hajtsd végre a **$r->start_time()** módszert:

```
$r = new q;
$r->start_time();
```

Az eredmény a következő:

```
17:09:09.10376000
```

Konstruktőrök a kiterjesztett osztályokban

Vedd észre, hogy a p-osztály q-ra történt kiterjesztése meghagyta a p()-függvényt mint konstruktőrt az osztályra vonatkozóan. A PHP3-ban ez nem történik meg; a PHP3 a q()-

függvényt keresi a q-osztályban, majd leáll. A PHP4 a q()-függvényt keresi a q-osztályban, ha a q()-függvény hiányzik, a PHP4 a p()-t keresi a p-osztályban. A következő kód kiter- -i jeszti a p-osztályt s-re, és kiegészíti az s() konstruktorfüggvénnyel. Az s() formázza és megjeleníti $thís->start-ot:

```
class s extends p

    function s()

        $m = explodeC'.", $this->start) ;
        print("<br>" . date("H:i:s", $m[0]) . "." . $m[l] );
```

Próbáld meg a kiterjesztett osztályban létrehozni a $t-objektumot a következő kóddal:

```
$t = new s;
```

A kapott eredményből egyenesen következik, hogy az s()-függvény működött. Megjelenített valamit, de a $this->start változó nem tartalmazott semmit, mivel a p()-konstruktor a p-osztályban nem hajtotta végre:

```
10:00:00.
```

Ahhoz, hogy mindkét osztályból származó mindkét konstruktőrt működésre bírd az objektum létrehozásakor, végre kell hajtanod az eredeti osztály konstruktorát az új osztály konstruktorából. A következő kód a p-osztályt kiterjeszti s-osztályra ugyanúgy, mint az előző példában, de ki van egészítve egy többletvonallal, hogy végrehajtsa a p()-függvényt az s()-ben lévő kód előtt:

```
class s extends p

    function s()

        $this->p();
        $m = explodeC'.", $this->start) ;
        print("<br>" . date("H:i:s" , $m[1]);
        $m[0])
```

Próbáld ki a kiterjesztett osztályon a következő kódot, hogy létrehozd a $t-objektumot:

```
$t = new s;
```

Az eredményből következik, hogy az s()- és a p()-függvények végrehajtódtak. Az eredmény a szépen megformázott idő, amit már a korábbi tesztekből megkaphattál.

```
17:09:09.10376000
```

Függvények cseréje

Ha létrehozol egy új osztályt, és a kódot ki akarod cserélni a függvényben, csak definiálj egy másik függvényt egy új osztályban, ugyanazon a néven, mmt az eredeti függvény. Egye-

dül itt tudod kicserélni a függvényt; ha a PHP lefordította a függvényt az objektum számára, nincs lehetőség a függvény cseréjére, vagy a kód cseréjére a függvényben.

A következő kód egy egyszerű osztálydefiníció. Az a-osztály tartalmazza a print_time()-függvényt, hogy egy időmegjelenítési módszert szolgáltasson:

```
class    a                                                                    ?
    *                                  i A -.jínjjjfijiffnsd ludáEÍddéU -vt Jirne ,'■>< Jix
    var    $ start;                              ' , . ' ■ ■ '
    function    a()

        $m = explodeC ", microtime () ) ;
        $this->start = $m[l] . substr ($m [0] , 1);    . " \) i;

    function print_time()                    ! ny*'1. U .;KU. $> 1g<~

        global $helpnet;
        $m = explodeC'.", $this->start) ;
        print("<br>The time is " . date("H:i:s.", $m[0]) . $m[i;
```

A következő kód kiterjeszti az a-függvényt b-re, és kicseréli a print_time()-függvényt egy kissé különböző kódra:

```
class    b    extends    a {
    function    print_time()
        {
        global $helpnet;
        $m = explodeC'.", $this->start) ;
        print ("<br>Replacement time " . date("H:i:s.", $m[0]) . $m[l]);
```

A következő kód a függvény cseréjét vizsgálja a b-osztályban, a $c-objektum b-ből történő létrehozásával, majd a $c->print_time()-módszer végrehajtásával, hogy látható legyen, melyik függvény van használatban:

```
$c    =    new    b;
$c->print_time();
```

Ebből adódik az eredmény, amely mutatja, hogy a b-osztályban a print_time()-függvény kicserélte az a-osztályban a print_time()-függvényt:

```
Replacement    time    08:24:16.16906100
```

Függvények törlése

Előfordulhat, hogy az osztály kiterjesztésekor ki szeretnél törölni egy függvényt. Habár a PHP nem engedi a függvények törlését, és ez érvényes az osztályok kiterjesztésekor is, ehelyett kicserélhetsz egy függvényt egy újra, amely semmit nem csinál, így a függvény lényegében hatástalanná válik.

Többszörös kiterjesztések

Előfordulhat, hogy egy már kiterjesztett osztályt szeretnél kiterjeszteni. A PHP lehetőséget ad a kiterjesztett osztályok újbóli kiterjesztésére a tulajdonságok és módszerek teljes öröklésével. Mivel lehet szó többszörös öröklődésről, ami két osztály egyszerre történő öröklését jelentené, kiterjesztheted az u-osztályt s-osztályra, ahol az s a p-osztály kiterjesztése, amit az alábbiakban bemutatunk. A következő tesztkód kiterjeszti az s-t u-osztályra, illetve hozzáadja az u()-konstruktort:

```
class u extends s

                          .; (í .,[03ra$)-í3d:-e
       function u()                                          r T.. ;q J!
          {                '                    i ; 9 ■
Irt       $this->s();
          $m = explode(".", $this->start); -
       print(date(" a", $m[0]) );              rJ-
                                                ■i <..'.'. i ■■,'->'!
    )
```

Próbáld ki a kiterjesztett osztályt a következő kóddal, hogy létrehozd a $v-objektumot:

```
$v   =   new
u;                                  > ;* » J i r. .ri | >y «i
```

Ebből adódik az eredmény, amely mutatja, hogy az u-osztály használta és örökölte mindkét függvényt, az s()-et az s-osztályból, illetve a p()-függvényt a p-osztályból. Az eredmény az időformátum az s-osztályból, a napszakmegjelölés (A.M./P.M.) hozzáadásával.

```
17:09:09.10376000 pm
```

```
                                    ■3 s". "i .
                                    t-.iif fi

                                          ^Tvf!< *d>") ::r.JLi'i
```

A dupla kettőspont (::) egy meglehetősen fura operátor, amelynek látszólag nincs nyilvánvaló haszna. Lehetőséget ad egy függvény osztályból történő végrehajtására anélkül, hogy létre kellene hozni egy objektumot az osztályból. Mivel nincs objektum az osztályban, nem érvényes a $this-hivatkozás a függvényben, és érvénytelenek az osztályváltozók is. Ez azt jelenti, hogy a legtöbb, amit használhatsz, egy adatformázó függvényhez hasonló általános függvény, éppen ezért ezeknek nem kell az osztályokban lenniük.

A következő kód egy általános time_format()-függvény egy formázott microtimeQ megjelenítésére, amelyet a honlapodon bárhol használhatsz. Tegyük fel, hogy a time_format() el van rejtve a p-osztályban, de úgy akarod használni, hogy ne kelljen objektumot létrehozni abból az osztályból:

```
    function   time   formát ($time)
V,  .,,,,
       {                                          ahöí
     j $m = explode(".", $time);          $m[1J);
       return(date("H:i: s", $m[0])
```

A függvény a p-osztály nevét követő dupla kettősponttal (::), majd a time_format()-fü
vény nevével érhető el:

print("<br>" . p::time formát(time( )) ; ■-;' - .....

Ha az osztályaid nem tartalmaznak rendes és használható függvényeket, amelyek megállják a helyüket, fontold meg a függvények osztálydefiníciókon kívül helyezését, hogy azok szabványos függvényként bárki által használhatóak legyenek.

ParSm ; 3 3*334 **m**nói*?-& ***mm***

A **parent** egy kulcsszó, amely értelmessé és hasznossá teszi a :: műveleti jelet. A **parent** kulcsszó lehetőséget ad egy kiterjesztett osztály forrására való hivatkozásra az elődosztály megnevezése nélkül. Amikor a p-osztály kiterjesztésével létrehozod az s-t, az s-osztályban lévő kód hivatkozni tud a p-osztály függvényeire mint forrásra. Ha átnevezed a p-osztályt **p2-re,** ezt csak az s extends p-osztályutasításban kell megtenned. A $p-t már nem kell a $p->()-ban átnevezned minden egyes alkalommal, amikor a classp-ét átnevezed. Elegendő a parent::p() egyszeri megváltoztatása.

Szoftverterjesztés és dokumentáció

Amikor elkezdesz terjeszteni egy alkalmazást vagy egy alkalmazás-kiterjesztést, a nyers kódot terjesztheted beillesztett állományként, függvénykészletként vagy objektumként is. Egy objektum biztosítja a legmagasabb szintű kódelkülönítést, ugyanakkor a leginkább összetett felületet is. Gondold végig, hogy mennyire bonyolult a kódod, mielőtt egy objektumot választanál csomagnak.

A nyers kódok adják a legtöbb lehetőséget a kódok megtanulásához, így ez kiváló módja a kisebb kódkijelölések gyakorlásképpen történő becsomagolásához. A nyers kódok a legkevésbé sértetlenek, mivel mindenki változtat rajtuk, ezáltal a legkevésbé alkalmasak folyamatos támogatás ellátására.

A függvények sokféle bemenetet elfogadnak, és könnyedén alkotnak egy kimenetet, ezáltal sokféle kódhoz illeszkednek. A függvényeket könnyű megérteni, mivel szemléltethetsz egy bemeneti táblázatot, és minden kombinációra egy pontos kimenetet. Ha nem tudsz elmagyarázni egy függvényt egy egyszerű állapottáblával, akkor nagyon valószínű, hogy annyira összetett, hogy át kell gondolni. Ha egy függvényt többszörös kimenettel szeretnél összeállítani, a függvény hatásainak elmagyarázásával és működésének a tesztelésével kapcsolatban komoly problémákkal fogsz találkozni. Többszörös kimenetek vagy összetett utasítási követelmények esetén az objektumok illeszkednek a kódcsomagokhoz.

Többszörös adatelemek és állapot

Ha több adatelemet kell az egyik függvényhasználatból a másikba átvinni, tekintetbe kell venni egy objektum használatát, és az adatok abban való tárolását. Az adatelemek a kódhasználatok közötti folyamatok állapotait fejezik ki. Ha az adatok teljes mértékben tárolhatók az objektumban, vagy megoszthatók az objektum és a környező kódok között, az objektum irányíthatja a benne lévő adatokat. Az objektumban lévő elemekre ekkor rá lehet bízni az egyik kódból a másikba történő információátvitelt.

Ha van egy news-osztályból létrehozott $n-objektumod, amely hírcsoportokat kezel, nyugodtan elolvashatod a rendelkezésre álló hírcsoportok listáját az objektum létrehozásakor, és a későbbi használat során hivatkozhatsz erre a listára. Az objektum tartalmazza a listát, ellenőrzi a bővítéseket és törléseket, valamint elvégzi a frissítéseket a hírcsoportszerverről. Az objektumnak nem kell a listát minden alkalommal ellenőriznie, amikor azt használja. Ha ezután hozzáadsz a news-osztályhoz egy listát az előfizetői csoportokról, a $n-objek-tum az előfizetéseket is ellenőrizni fogja.

Képzeld el, hogy hozzáadod az előfizetői listát a $n-objektumhoz egy külső listaként. Ha elküldesz egy az előfizetői listán alapuló kérést a $n-objektumnak, a benne lévő kódnak minden alkalommal ellenőriznie kell, hogy a kérés érvényes-e, A $n-ben lévő hírcsoportlistán végrehajtott minden frissítés ki fogja hagyni az előfizetői listát a szinkronizálásból.

A legjobb megoldás a news-osztály kibővítése lenne egy másik osztállyal, amely magába foglalja az előfizetői listát és a vonatkozó kódokat is, így az előfizetői lista zárolva lehetne, szinkronban a hírcsoportlistával. Az objektumnak minden adatot befolyásolnia kell, ami hatással van az állapotára.

Többszörös kimenet

Amikor egy olyan kódot írsz, amely 22 kimenetet produkál, 22 függvényt használhatsz vagy jóval kevesebb objektumot. Az objektumok kínálják a legnagyobb megtakarítást, ha minimálisra tudod csökkenteni őket, amit a végfelhasználóknak meg kell tanulniuk. Ugyanakkor nagy erőforrás-pazarláshoz vezethet, ha számos nem kapcsolódó adatelemet zsúfolsz egyetlen objektumba. Az objektumok megértése nehézkessé válik, ha az objektum kapcsolat nélküli tulajdonságokkal rendelkezik, illetve sok módszert tartalmaz, amelyek nagy része kettős műveleteket hajt végre a kapcsolat nélküli adatokon.

Ha létrehoztál egy objektumot egy teljes weboldal elkészítéséhez, de az objektum az oldalt egy csomó kis részletre bontva adta vissza, például különálló bekezdésekben és fejrészekben, illetve 75 különböző adatot kért be, ráadásul mindegyiket más módszerrel, valószínűleg te sem fogod azt többet használni. Szerencsésebb megközelítés lenne egy olyan beágyazott szerkezetet létrehozni, ahol az objektumok vagy a függvények fejrészekhez hasonló elemeket hoznak létre, és egy általános objektum vagy függvény hozza össze az egészet.

Egy oldalkészítő osztály kifejlesztését én bekezdésekhez hasonló, különálló formázott függvényekkel kezdeném, ezeket összeállítanám oldalrészekre, és írnék egy átfogó függvényt az oldal megjelenítésére. Valószínűleg nem látnék hozzá az egy osztályba kerülő függvények elkészítéséhez, hiszen csak egy kimenet lehetséges, az elkészült oldal.

Próbáld meg először a dokumentációt megírni az objektumhoz a kód elkészítése előtt. Sorold fel a bemenő és kimenő adatokat, valamint az egyik objektum használatából a másikba áttárolt elemeket. Használ) objektumot, ha egynél több output van, vagy ha nagy mennyiségű adat képződhet az objektumban.

Gyors megoldások

Objektumok mentése sessiönökben és a__ sleepQ használata

Ez a megoldás egy egyszerű, műveleteket nyomon követő objektumot állít elő, az objektu mok és a __ sleep()-függvény működésének bemutatására. Ezt az objektumot egyedi látogatókövető rendszerré fejlesztheted, kiegészítve a látogatóknak szánt testre szabott megjelenítéssel, amely a böngészőjük azonosítására épül. Ha a látogató ügyfélként jelentkezik be, szükséged lehet bizonyos információkra és megjelenítési adatokra a látogatói profil alapján. Ez három, egymással együttműködő objektumot juttat eszembe.

Az első objektum nyomon követi a látogatók session-eit; mikor és hol léptek be az oldaladra, milyen oldalakat látogattak, mit gépeltek be a keresőbe és hol léptek ki. Ha bejelentkeztek, a nyomkövető azt is megmutatja, hogy hol léptek be és hol léptek ki, feltéve persze, hogy már kijelentkeztek.

A második objektum használatára akkor kerülne sor, amikor a látogató először lép az oldalra, és az objektum eldöntené, hogy az oldal megjelenítésén kell-e változtatni a látogató által használt böngésző konfigurációjának megfelelően. A legésszerűbb megoldás az oldal egy meghatározott böngészőbeállításához történő elrendezését egy böngésző objektum észlelésével kezdeni, olyasmivel, ami igényel néhány okosan együttműködő oldalt és némi JavaScript-et a böngésző teszteléséhez. Ez a példa kihagyja az intelligens megközelítést, a második objektumot és az oldalbeállítást, és csak a böngésző ügynöksztringjét rögzíti későbbi elemzés céljából.

A harmadik objektum akkor lép be, amikor valaki bejelentkezik, és információármalás történik a felhasználói profilján, például átírja a nyelvi hivatkozását. Ha a nyelvi hivatkozáshoz és a felhasználói profilhoz hasonló dolgokat csak később fogod hozzáadni, olvasd el a 12. fejezetet, és bővítsd ki ezt az objektumot a HTTP_ACCEPT_LANGUAGE és más fontos mezők vizsgálatával. Amikor már több nyelvet is hozzá tudsz adni, kiderítheted, melyek közülük a legnépszerűbbek.

Az objektum a session_logger-osztállyal kezdődik:

```
class   session_logger

   var $agent;
   var $log_file;
   var $messages;
   var $remote;

   function _ sleepf)

      if (isset($this->messages) and strlen($this->messages))

         if($file = fopen($this->log_file, "a"))
```

```
                fwrite ($file, $this->messages);
                fclose($file);


        return(array("agent", "log_file", "remote") }
        function add_message($text)

            if (! isset ($this->messages) ) ** f
-ou             $this->messages = "";

            5m = explodef" ", microtime()) ;
            $this->messages .= session_id() . " " .    date("Y-m-d          $m[i;
            substr ($m[0], 1) . " " . $text          "\n";

        function session_logger()

            if(!isset($this->agent))

                $this->log_file = "t:/sessionlog.txt"; $n =
                "HTTP_USER_AGENT";    ,.-r •■-,yC.ik/f
                $this->agent = getenv($n);
                $this->add_message($n . " "
                                                $this->agent)

            if(!isset($this->remote)

                $n = "REMOTE_HOST";
                $this->remote = getenv($n);
                if(!isset($remote) or !strlen($remote)

                    $n = "REMOTE_ADDR";       Jí,...--■',,
                    $this->remote = getenv($n);

                $this->add_message($n .
.Sí             1                               $this->remote)

    1
```

A session_logger a Sagent-, Slog_file-, Smessages-, $remote-változókat tartalmazza, és a $messages-t, amelyre csak a szknpten belül van szükség, illetve három egyéb műveletet, **amelyek az** egész **művelet során** szükségesek. Ezt az osztályt tartalmaznia kell, ha a műveleti kód elindul, így az osztály olvasható lesz, mielőtt a műveleti kód visszanyeri az osztályon alapuló objektumot. Ez azt jelenti, hogy a **session.auto_start** php.ini-ben

történő beállításával nem indíthatod el automatikusan a műveletet.

A **Sagent** megkapja a **HTTP_USER_AGENT**-mezőt, amikor a látogató belép az első **lapra.** A böngésző képességeit kibővítheted az ügynökmező dekódolásával, csökkentve ezzel az oldalmegjelenítések számát, például a képek kikapcsolásával a szöveg alapú böngészők esetében.

A $log_file a naplófájl nevét tartalmazza, amely nevet újra és újra továbbad és tulajdonságként tesz elérhetővé, így a tesztkód meg tudja jeleníteni a naplófájlt.

■ta.ie *cio-ZZSB*.

A $messages a napló számára veszi az üzeneteket az add_message()-függvényen keresztül, ami a Sthis->add_message()-módszerkém érhető el, illetve ellátja üzenetekkel a __sleep()-függvényt, hogy azokat a szkript végén átküldje a naplóba. Ezt oldalról oldalra nyeri vissza, mivel az összetevők a naplóban vannak.

A $remote összegyűjti a látogató gazdaszámítógépének a nevét és az IP-címét az első belépésnél, ezt oldalról oldalra elvégzi, amennyiben valamire használni szeretnéd. A valóságban azt, hogy a gazdaszámítógép neve tartalmaz-e az országra utaló információt vagy egyéb fontos tényezőt, az első belépésnél ellenőrződ, ezeket elmented, majd elhagyod a $remote-parancsot. Az IP-címre akkor lehet szükséged, ha kitüntetett bánásmódban részesítesz másik weboldalt vagy szervert. Célszerű az előjogot ismét az első belépésnél meghatározni, majd pedig az IP-cím helyett az előjogállapotot eltárolni.

A __sleep()-függvényt a session-kód hozza működésbe a változók besorolásakor a szkript végén, és egy tömböt ad vissza, amely tartalmazza a besorolandó változók neveit. Mivel a __sleep() egyszer fut le a szkriptben, annak a végén, minden naplóüzenet kivitelére hasz nálhatod, egy a naplófájlban végrehajtott hatékony találatként. Ha kritikus üzeneteket akarsz feldolgoztatni, a kóddal végeztesd el az azonnali kiíratást, ahogyan már más példák ban bemutattuk a könyvben. A fájlfüggvényeket a 8. fejezetben mutatom be.

Az add_message() egy üzenetet ad a $this->message-hez, kiegészítve egy előtaggal, ami tartalmazza a művelet azonosítóját, dátumát és a legközelebbi millimásodperchez képest eltelt idejét, így ütemezni tudod a kód részeit. Ha új kódot tesztelsz, írj rengeteg üzenetet, és ellenőrizd a kód új részeinek végrehajtásához szükséges időt. Amikor a kód stabillá válik, vedd ki belőle a tisztán időzítési üzeneteket, és tartsd meg a rendellenességeket visszajelző kódrészt:

Mieló'tt létrehozol egy objektumot, vagy mielőtt a műveleti kód dekódolhatna egy abba az osztályba tartozó objektumot, annak tartalmaznia kell az osztályt, ezért tagold az osztálykódot PHP-jelzőként, helyezd az osztálykódot egy session_logger.html nevű fájlba, helyezd a fájlt a beágyazó útvonalba, illetve írd be az alábbi require()-függvényt a szkript elejére, amit a session_start{) követ:

```
require("session_logger.html");
session start ();
```

i

Az oldal valahogy úgy fog kinézi, mint a következő lap, amely tartalmazza az osztálykódot, a művelet kezdetét, egy minden oldalra érvényes általános kódot és egy a jelen oldalhoz tartozó speciális kódot. Ha elégedett vagy az osztálykóddal és a művelet kezdetével, akkor egyesítsd mind a kettőt az általános HTML-fájllal:

```
<?php
require{"session_logger.html");
sessionstart ();
require("common.html"); //
Processing code here
```

IC i, fi: .V. £

Egy objektum létrehozásához és session-rekordba helyezéséhez használd a következő kódot, hogy ellenőrizd a Ssession_log-objektum meglétét, illetve hozd létre azt a session_logger-osztályból, ha esetleg nem létezik:

```
if(isset(Ssession log))
```

-i, print ("
Agent from session .
record: $session_log->add_message
("tnessage") ;
Ssession_log->agent ■>

```
else
```

'st Jípvbrí míi-'II KÍ

```
$session log = new session logger;
print("<br>Agent: " . $session_log->agent );
print("<br>Remote: " . $session log->remote );
session_register("session_log");
}
```

E .s'Ui.J^o^í

A session_log-objektum tesztelésének a segítésére a kód megjeleníti annak néhány tulajdonságát a létrehozás után, és egy tulajdonságot azon előfordulások közül, amikor az objektum már)elen volt a session-rekordban. A session_log automatikusan létrehoz egy üzenetet a definiálásakor, az osztály session_logger() konstruktőr-függvénye miatt, majd egy üzenet kerül hozzáadásra manuálisan a $session_log->add _message()-függvényen keresztül a későbbiek során, így láthatod az objektum működését.

Az első belépésnél a következő üzenet fog megjelenni az objektum működésének megerősítéseképpen. Természetesen az ügynöksztring tartalmazni fogja a böngésződből származó ügynöksztringet, és az IP-cím vagy a host-név a a saját host-odhoz vagy proxy szerveredhez tartozóval fog megegyezni. " T~ ' ~"

```
Agent: Mozilla/4.77 [en] (WinNT; U)-«~...—t————————————
Remote: 192.168.0.100
```

A későbbi belépések során a következő üzenetet fogod látni, jelezve, hogy a szkripted felvette a $session_log-objektumot a session-rekordba:

```
Agent from session record: Mozilla/4.77 [en] (WinNT; U)
```

A naplófájl összetevó'inek a nyomon követéséhez illeszd be a következő kódot a tesztoldalad aljára (jusson eszedbe, hogy ez minden oldalon meg fogja jeleníteni a naplófájlt az előző szkript végétől):

```
if(file_exists($session_log->log_file))
```

,,, $lines = file ($session_log->log
file) ; while (üst ($k, $v) = each
($lines))

ili

:£ xfi'JO»*3Í t bil

588

```
print("<br>"  .  htmlentities($v)
```

A naplófájlod a következőképpen fog megjelenni (a sorok a könyv elrendezésének , megfelelően vannak tagolva):

```
652ad60cllab3d7457e56c39aaa5c7fa  2001-07-16  17:16:01.74908700
  HTTP_USER_AGENT  Mozilla/4.77  [en]  (WinNT;  U)
652ad60cllab3d7457e56c39aaa5c7fa  2001-07-16  17:16:01.74932100
  REMOTE_ADDR  192.168.0.100                                    K
652ad60cllab3d7457e56c39aaa5c7fa  2001-07-16  17:17:12.47705700  message
652ad60cllab3d7457e56c39aaa5c7fa  2001-07-16  17:19:37.90405100  message
```

Mondhatni pikáns technikája ez a tesztelésnek, de *a pikáns* jelző nem minden kultúrában elfogadott a programozás jellemzésére, úgyhogy inkább a kifinomult szót használom. A függvények és egy tömb használatával bármit megtehetsz a bemutatott megoldásban, de meg kell gyó'ződnöd róla, hogy számos függvény és a tömb egyéni nevekkel rendelkezik-e a szkriptben. Egy objektum használatával csak két különálló névre van szükséged: egy osztálynévre és egy objektumnévre. A változókra (tulajdonságok) és a függvényekre (módszerek) történő hivatkozás már kissé unalmasabb a hosszabb nevek miatt, de egyszerűbb a neveket különbözővé tenni.

Most már bármivel kibővítheted a kódot, az működni fog az adott szerveren tárolt összes oldalon, mivel nincs szüksége a rendszernaplóhoz történő hozzáférésre vagy bármely egyéb különleges műveletre. Tényleg csak a legfrissebb PHP-re lesz hozzá szükség, mivel a _ sleep() körülbelül a PHP 4.0.6-ba került bele először. (Szerintem mindig a legfrissebb PHP-t kellene használnod, és egy olyan Internet-szolgáltató gazdagépén működni, aki fontosabbnak tartja a PHP-t, mint a Pearl-t vagy bármi egyebet, kivéve az Apache-ot és a biztonsági szempontokat.)

Objektumfüggvények használata

Az objektumfüggvények lehetővé teszik az objektumok, objektummódszerek és -tulajdonságok használatát az aktív szkript által hozzáférhető információ alapján. Esetlegesen meg kell nyitnod egy fájlt, be kell olvasnod belőle egy adatmintát, majd el kell döntened, hogy melyik osztályt fogod használni a megfelelő feldolgozási objektum, fájl számára történő létrehozásához. Ezek a függvények lehetővé teszik egy dinamikus környezet építését, és minden osztály, módszer és tulajdonság ellenőrzését a használat előtt.

call_user_method()

Előfordul, hogy egy objektumból végre szeretnél hajtani egy módszert, de az objektumban több módszer, sőt több objektum van, és az objektummódszer nevét bármikor változtatható módon szeretnéd felépíteni. A **call_user_method()** elfogadja mind a módszernevet, mind az objektumnevet mint sztring, így mindkettó't építheted adatokból.

Egyszerű HTML-példaként készíthetsz egy fejléc szintű számlálót annak eldöntésére, hogy l-es, 2-es, vagy 5-ös fejléc-szintet jelenítesz meg, illetve készíthetsz formázó objektumot is egy módszerrel minden egyes fejléc számára. Ekkor esetleg szeretnéd a fejlécszámlálót a módszer végére beilleszteni. A következő kód mutatja a formázó objektum használatát két feilécszint esetén:

```
$format->headingl("Portugál");
$format->heading2("Algarve");
```

Most állítsd be ugyanazt a fejlécet automatizált módon, esetleg egy adatbázisból egy tömbbe beolvasott adattal. Szimuláld a folyamatot a következő egyszerű tömbbel:

```
$headings[l] = "Portugál";
Sheadings[2] = "Algarve";
```

Most küldd át a tömböt a formázó objektumon, a while-ciklus, a list(), az eachQ és a call_user_method() használatával. A call_user_method() az objektumneveket és a módszerneveket, illetve a módszer által kért bármely számú paramétert elfogadja sztnngként:

```
while (Üst ($k, $v) = each ($headings) )

    call_user_method("heading" . $k, "formát", $v);
    }
```

call_user_method_array()

A call_user_method_array() egy kisebb módosítása a call_user_method()-nak, ami a PHP4.0.5-öt egészíti ki, így ez a példa egy korábbira épül. Képzelj el egy módszert, amely a következő példához hasonlóan a többszörös paramétereket egyszerre elfogadja:

```
$format->heading("Portugál", "Algarve");
```

A call_user_method() a következő, határozottan unalmas kód használatára is képes a helyzet megoldása érdekében, de képzeld csak el, hogyan nézne ki hat fejlécszinttel:

```
if(count($headings) == 2)

    call_user_method("heading", "formát", $headings[1],
    $headings[2]); }
elseif (count($headings) == 1) {
    call_user_method("heading", "formát",
    $headings[1]);
```

A következőkben a többszörös paraméterek kezelésének egy egyszerűbb módját mutatjuk be, feltéve, hogy egy tömbben vannak: / wo«.».,__"^*m"-

```
call user_method_array("heading", "formát", Sheadings);
```

Wi-

class_exists()

A class_exists() egy alternatív megoldást jelent az include_once() használatára, amikor az osztályokkal használod. A class_exists() elfogadja az osztályok neveit, és ha az osztály definiálva van, ugyanúgy visszaadja, ha nem, akkor hibásan. Ezt a kódot használhatod egy objektum létrehozására, ha ez még nem áll rendelkezésre a szkriptben. A legegyszerűbb módja egy osztály létrehozásának a fájlból történő beágyazás, include() használatával. A class_exists() használatával történő'vizsgálattal, győződj meg róla, hogy nincs-e kétszer beágyazva, és csak akkor ágyazd be az osztályt, ha még nem létezik. Ennek korábban volt értelme, de most a include_once() alkalmazásával a beágyazás már csak egyszeresen történik meg. Ez egy egyszerűbb módja annak, hogy meggyőződhessünk róla, hogy az osztálydefiníciók csak egyszer szerepelnek.

következő kód tartalmazza a session_logger-osztályt, a session_logger.html-fájlból:

```
include("session    logger.html);
```

A következő kód csak egyszer tartalmazza a session_logger-osztályt a session_logger.html-fájlból, az include_once() használatával:

```
include_onceCsession_logger.html");
```

get_class()

Ha már van egy objektumod, de szeretnéd tudni, hogy milyen osztályba tartozik, használd a következő kódot. A példa a session_log-objektumot használja, a session_logger osztály ból a "class_exists()" szakaszban létrehozva:

```
print("<br>"    .
get_class("session_log")) ;
```

Az eredmény a következő lesz:

```
session_logger
```

get_class_methods()

Ha egy objektumban minden osztályt vagy módszert szeretnél ismerni, használd a get_class_methods()-függvényt. Töltsd be a függvényt, egy osztályt vagy a PHP 4.0.6-ban már egy objektumnevet, és a függvény visszaad egy módszernevekből álló tömböt. A következő kód betölti a get_class_methods()-t, az osztály nevét az előző példából. A kód ugyanolyan jól fog működni, ha a session_log-ot, (az objektum nevét a "get_class"-ból) töltőd be:

```
$list    =    get_class    methods("session    logger");
while (üst ($k,    $v)    =    each ($üst) )

    print("<br>"    .    $v);
```

Az eredmény a következő módszernevekből álló lista: ' ir ! ■ ■ . ■ ■

```
__sleep
add_message
session_logger
```

get_class_vars()

Amikor egy osztályra vonatkozó minden tulajdonságra kíváncsi vagy, használd a get_class_vars()-függvényt. Töltsd be a függvénybe egy osztály nevét, és a függvény visszaad egy társított tömböt a változókról és az értékekről, az osztályra vonatkozó összes tulajdonságot ábrázolva. A tömb csak inicializált változókat tartalmaz, ezért próbáld ki az osztályaiddal is, hogy mit ad vissza.

A következő kódpélda betölti a get_class_vars()-t, az osztály nevét az előző példából, és megjeleníti az ebből eredő tömböt. A kód egy értékben egy tömbszintet fogad el:

```
$list = get_class_vars("sessíon_logger");
while(list($k, $v) = each($list))

    if (is_array() )          ■ '                    í,
                                                     -O.;3E
        {
        while(list($k2, $v2) =
        each($v))

            print("<br>" . $k . ": " . $k2 . ":
            " . $v2);


    else {
        print("<br>" . $k . ": " . $v);

     }
```

Amikor a kód a session_logger-rel szemben fut, nem hoz létre kimenetet, mivel egyik változó sincs inicializálva. A kimenet működőképességének igazolásaként cseréld a var Smessages-t, var Smessages = "test text"-ra, és futtasd le újra a kódot. A következő kimenetet fogod kapni:

```
messages: test text
```

[.

get_declared_classes()

A szkriptben szereplő összes osztálynév kiderítéséhez használd a get_declared_classes()-függvényt, ammt az a következő kódban látható. Ez létrehoz egy tömböt a függvényből és megjeleníti azt:

```
$list = get_declared_classes()
;                                                    '; (ve      "<  d>")
while(list($k, $v) =                                          Jr
    each(Slist))
    {
    print ("<br>" . $v) ;
```

Az eredmények a session_logger-osztály teszteléséhez használt szkriptet ábrázolják: *j>*

```
stdClass
OverloadedTestClass
Directory
COM
VARIANT
session_logger
```

Az stdClass és az **OverloadedTestClass** speciális osztályokként kerültek bevezetésre a PHP 4.0.1pl2-ben. Nálam a COM-kiterjesztés van telepítve, a másik kettő pedig egyelőre még kutatásra váró rejtély, de valószínűleg más, már betöltött kiterjesztések közül valók.

get_object_vars() «

A get_object_vars()-függvény úgy működik, mint a get_class_vars(), de ez egy objektum számára dolgozik. A következő kód betölt egy objektumot a get_object_vars()-parancsba, és megjelenít minden név-értékpár változót: .

```
51ist = get_object_vars($session_log);
while (üst ($k, $v) = each($üst))

    if(is_array())

        while (üst ($k2, $v2) = each($v))

            print("<br>" . $k . ": " . $k2 . ": " . $v2);


    else

        print("<br>" . $k . ": " . $v);
```

Ha nem számít, hogy a kiírás hogy néz ki, van egy alternatív megjelenítési lehetőség, a **print_r(),** amely kiírja a változókat vagy a tömböt egyenesen a böngészőbe, minimális formázással:

```
print_r(get_obj ect_vars($session_log));
```

get_parent_class() " , " :s

A **get_parent_class()**-függvény elfogadja egy osztály nevét, és visszaadja az eló'dosztály nevét.

Szintén elfogadja egy objektum nevét, és visszaadja annak az osztálynak a nevét, amely az objektum építése során az összeállításhoz használt elődosztály volt. A következő kód megjeleníti a Ssession_log-objektumból visszanyert osztálynevet:

```
print("<br>" . get parent_class($session_log));
```

Jí SS,

Líí-

is_subclass_of() *üu^sw-&zm>ij*

Tegyük fel, hogy az a-osztály, b-osztályra van kiterjesztve, és a b ki van terjesztve, hogy létrehozza a session_logger-osztályt. Azt szeretnéd tudni, hogy a $session_log-objektum az a-osztályban készült-e. Használd a következő kódot egy objektum és egy osztálynév is_subclass_of()-parancsba történő betöltéséhez, ami megbízhatóan visszaadja, hogy a Ssession_log-objektum elődosztálya az a-osztály, vagy 'hamis' ha ez nem igaz:

```
if(is_subclass_of($session_log,      "a"))

    print("<br>True");
```

method_exists()

Tartalmazza a Ssession_log-objektum a print_list-módszert? Egy objektumban megvizsgálhatsz egy osztályt a method_exists()-paranccsal, amint azt a következő példa is mutatja:

```
$method = "print_list";
if(method_exists($session_log, $method))      *        ,2    ....

    print("<br>Method " . $method . " exists."); }
else {
    print("<br>Method    "    .    Smethod    .    "    does    not
    exist.");
```

Ebben az esetben a print_list nem áll rendelkezésre.

"í .d

Honlap testreszabása objektumokkal

Ha felkészültél a weboldalaid testre szabására a látogató böngészőjéből származó információk felhasználásával, vagy a 12. fejezetben tárgyaltak alapján a következő kód generál egy objektumot, amelyet alapként használhatsz a speciális kódodhoz:

```
class    agent                - ;„                ■     -       "

    var $agent; var
    $browser; var $cookies
    = falsé; var $frames =
    falsé; var $images =
    falsé; var $language;
    var $version = 0; var
    $words; function
    agent()

        $this->agent    =    getenvf'HTTP    USER
        AGENT");
```

(Jaasto

/■'> ,:iv >n- mtíííísido

f

```
   $this->language = getenv("HTTP_ACCEPT_LANGUAGE");
   $this->decode_agent($this->agent);
   $this->get_language($this->language); global
   $HTTP_COOKIE_VARS; if(isset($HTTP COOKIE VARS))
```

Ánü? Á . H3S

```
      $this->cookies = true;
```

t

```
function decode_agent($agent) {
   $x = explode(" ", $agent); $y
   = explodeCV", $x[0] ) ;
   switch($y[0])

      case "Mozilla":
         $this->browser = $y[0];
         $this->version = $y[l];
         break;
      default:

   if (substr($x[l], 0, 1) == "[" and substr($x[1], -1)
                                                    == "]")
      if (!strlen($this->language)) { $this->language
         = substr($x[1] , 1, -1);



function get_language($language) { if
   (file_exists("./words" . $language . ".html"))
      {
      include_once("./words" . $language . ".html")
      $this->words = $words;
      }
   }
function local ($word) { if($this->words and isset
   ($this->words[$word]))
      {
      $word = $this->words[$word];
      } return
   ($word);
```

Az agent-osztály azért kapta az ügynök nevet, mert a böngésző ügynöksztringjének a be olvasásával indul. Az ügynöksztnng minden szkript számára elérhető, és a session élete so rán meg is változtatható, ha a látogató megváltoztatja a böngésző beállításait, például a nyelvet. viv

Az agent-osztály a változók egy listájával kezdődik, amely tartalmazza a **$cookies-,** a **Slanguage-** és a Swords-függvényeket is. Ezek közül néhány alapbeállításként üres, míg

mások rendelkeznek alapértelmezett beállításokkal. Miért van kétféle megközelítési mód? Vedd figyelembe, hogy hogyan vizsgálsz egy változót, illetve hogyan jutsz egy alapértelmezett értékhez. A sütik esetében hasznos lenne, ha csak a beállításokat kellene vizsgálni a sütik if(Sthis->cookies)-függvénnyel történő használatához, ami azt jelenti, hogy a változónak léteznie kell, illetve 'igaz' vagy 'hamis' készletként kell, hogy beállítva legyen. (Ha a PHP-figyelmeztetéseket kikapcsolod, kevésbé lehetsz pontos, ellenben több topográfiai hiban átjuthatsz.)

Az agent()-függvény akkor megy végbe, amikor az osztály egy objektummá válik, és begyűjti a böngésző ügynöksztringjét. Bejövő süti esetén a PHP a $HTTP_COOKIE_VARS-ban tárolja azt, tehát egy változó létezésével kapcsolatos vizsgálat azt jelenti, hogy a böngésző elfogadja a sütiket.

Az decode_agent()-függvény az ügynöksztringből történő fontos információk kinyerésére használt kódok prototípusa, így a böngésző típusa alapján dönthetsz. A kód keresi a nyelvmezőt az ügynöksztringben, és az alapján elvégzi a $this->language-beállítást. A HTTP_ACCEPT_LANGUAGE-ben lévő nyelvérték felülír mindent, amit az ügynöksztringben talál. Ezt a kódot elfogadja a agent()-parancs, de eléggé megnehezítené az életedet, ha az agent dekódoló kódot le szeretnéd cserélni egy részletesebb kódra. A dekódoló kód különálló függvényben történő megtartásával könnyedén kiterjesztheted az agent-osztályt egy újra, ugyanakkor lecserélheted a decode_agent()-függvényt is.

Szintén választhatod a get_browser()-függvény használatát is az ügynöksztring objektumba történő dekódolásához, ami elméleti információk tömegét tartalmazza egy böngészőről. A get_browser()-függvény könnyedén használható az agent-osztály kiterjesztése során. Ha nagyon sok többletmunkát jelentene a get_browser() használata, végezd el a dekódolást egyszer és mentsd el az eredményt a session-rekordban.

Hivatkozás:	oldal:
Böngészőalapú kód	245

A get_language()-függvény egy másik kódblokk, amelyet átadhatsz az agent()-függvény-nek, de jobb inkább önállóan meghagyni, mivel így az osztály kiterjesztésekor lecserélheted a függvényt. A get_language() beolvassa a nyelvi információkat a beágyazott fájlokból. Egy sok dinamikus szöveget és nyelvet tartalmazó oldalon én először a get_language()-függvényt cserélném le egy adatbázisolvasással. Másik előnye a függvény elkülönítésének az, hogy teszteléshez megváltoztathatod a $language-ben lévő értéket, és ismét végrehajthatod a get_language()-függvényt a teszteléshez szükséges nyelv beállításához.

A localQ-függvény lefordítja az angol mondatokat és kifejezéseket a get_language()-függ-vény által betöltött táblázatban használt nyelvre. Az angolt lecseréli a helyi, a weboldalon használt nyelvre, illetve a fordítás a böngésző által megjelölt bármely helyi nyelvre fog megtörténni.

Az osztály teszteléséhez használd a következő kódot, amely egy objektumot hoz létre az osztályból:

```
Sa = new agent;
```

Próbáld ki az objektumot a következő kóddal:

```
print("<br>Agent: " . Sa->agent);
if ($a->cookies)

    print ("<br>Cookies true.");

if ($a->language)
    {
    print("<br>Language:    "    .
    $a->language);
```

Az ügynöksztring mindig jelen van, tehát a sztringet csak ki kell íratnod. A cookies-változó csak annak a jelzésére szolgál, hogy a cookie-k használatban vannak-e, tehát a kód tesz teli, hogy a változókra vonatkozó **állítás** igaz-e, és megjeleníti az üzenetet. A nyelv fordítá sa akkor fordul elő, ha ezt a nyelv kéri, tehát egy nyelvterület használata tartalmaz egy tesztet is annak kiderítésére, hogy a nyelvterület be van-e állítva.

A következő kimenet a kód vizsgálatát tartalmazza:

```
Agent:    Mozilla/4.77    [en]    (WinNT;
U) Cookies    true. Language:    en
```

Az alábbi teszt a nyelvi beállítás megváltoztatására alkalmas. Annyit kell csak tenned, hogy az alábbi kódot hozzáadod a szkript végéhez, hogy a **get_language()**-függvényt az új nyelv fordításának beolvasására kényszerítsd:

```
$a->get_language("de");
```

Próbáld ki a fent megjelenített teszt nyelvi beállításának módosítását:

```
print("<br>"    .
$a->local("agent") if($a->cookies)                $a->agent);

    print ("<br>"    .    Sa->local ("cookies")    .    "    "    .„ ia-^lqg

if ($a->language) {print("<br>"    .    Sa->local("language")    .    ":
    "    .    $a->language);
```

Minden szöveges sztnng olyan módon van tördelve a $a->local()-függvényben, hogy a szöveget a helyi nyelven jelenítse meg. Az ezen az oldalon használt szövegfájlok csak szavakat tartalmaznak, így a *cookies true* kifejezés két szóból áll, amelyeket külön-külön fordít le. Egy élesben működő rendszerben érdemes teljes kifejezéseket beadni a fordító rendszerbe a jelentés pontosságának kipróbálásához.

Az eredmény a következő (ha a német fordítás rosszul sikerül, a Bábel Fish fordítási lehetőséget okold az AltaVista-ban):

```
mittel: Mozilla/4.77 [en] (WinNT; U)
plStzchen zutreffend. sprache: en
```

Hogy néz ki a fordító fájl? A következő fájl a német nyelv használatához van betöltve, és csak néhány, a Bábel Fish-szoftverrel lefordított tesztszót tartalmaz (a Bábel Fish a *cookies true*.-*x*. egy kifejezésként fordítja, a *cookies true-t* pedig egy másikként, ami jó példája a számítógépes fordítás problémáinak): ;.'.,: § g,> /ifi;

```
<?php                                    ■   »--        íi
$words["agent"] = "mittel";     '~! f ¡. ■ ■■ts
$words["cookies"] = "plStzchen";
$words["dog"] = "hund";                  :<-,-2   " --c s- ıÉ :-'id^
$words ["true" ] = "zutref fend" ;    ^ -f 'íí '^'íH'v'ir;!1 °r'í-:e1aii sifí Swords
[ "language" ] = "sprache";   'r -:.'■. e*^ .on ¡ PH)1* a
```

A fordításokat igen sokféleképpen elrendezheted, beleértve az XML-fájlokat és az adatbázisokat is. Az a fontos, hogy úgy rendezd el az objektum felépítését, hogy végrehajthass változtatásokat anélkül, hogy újra kellene írnod az eredeti objektum szerkezetét. Az agent-osztálynak megfelelő a struktúrája ahhoz, hogy kicserélhess a fordításhoz hasonló részeket az osztály kiterjesztése során.

Hírcsoportok olvasása

A hírcsoportokból származó információk olvasásához szükséged lesz hírszerverek neveire, a szervereken található hírcsoportok listájára, a hírcsoportokban lévő cikkek felsorolására és a cikkek fejléceinek, szövegtörzsének olvasási módjára, beleértve a csatolt állományokat is. Ha ez az egész túl bonyolultan hangzik, ne aggódj, nem kell az egészet egyedül bevállalnod.

A hírszolgáltatás az RFC 977-ben leírt (www.faqs.org/rfcs/rfc977.html) Hálózati hírátviteli protokollon (Network News Transfer Protocol (NNTP)) és az RFC-k egy levelezéssel megosztott készletén alapul. Néhány hírcsoportszerver könnyen elérhető a PHP IMAP-függyényeivel, ezekkel azonban nem láthatod, hogy mi zajlik a háttérben, így nem tudod megérteni a hírcsoportok belső működését. Használhatod a PHP kísérleti csatolófüggvényeit is a hírcsoportszerverek elérésére, de ez azért már túlzásnak tűnne. A PHP-ban megszokott fsockopen()- és normál fájlfüggvények (1. 8. fejezet) megfelelnek a hírcsoportok olvasásához és az NNTP megtanulásához, ezért ebben a megoldásban az fsockopen()-t használjuk.

Bármilyen nyilvános hírcsoportszervert elérhetsz, de némely esetben valós problémákkal találkozhatsz a szerver konfigurációja vagy az általa használt szoftver miatt. Tölthetsz le PHP-objektumokat a hírek és a levelezés kezelésére, de lehet, hogy nem fogod megérteni bonyolult kódjaikat mindaddig, amíg nem szerzel tapasztalatokat saját, kisebb objektumokkal. Teljes hírcsoport-alkalmazásokat is letölthetsz, de ebben az esetben is a kód megértésének problémájával leszel kénytelen szembesülni, ha változtatásokat akarsz végrehajtani, vagy ha egy problémás hírcsoportszerverrel találkozol.

A kód ebben a megoldásban lehetővé teszi az egyszerűbb olvasást a hírcsoportszerverről és a hírcsoportokból, végül pedig a hírelemek feldolgozását is. A csatolt állományokkal itt nem foglalkozunk, mivel azok MIME-kódolásban vannak tárolva, ami a 15. fejezetben szerepel.

A következő kód definiálja a news-osztályt, egy hírcsoportlista $groups-ba történő beolvasásához a Sservers-ben szereplő szerverek listájából. Az érthetőség végett egyszerre csak egy szervert fogok használni, és az 50.000 elérhető hírcsoport számát 10-re korlátozom:

```
class news
    {
    var $servers;
    var $groups;
    var $limit = 10;
    function get_groups()

        Slimit = 0;

        if(isset($this->groups))

            unset ($this->groups);

        if (isset($this->servers))

            while (üst ($k, $server) = each ($this->servers) )        '* A

                if($file = fsockopen($server, 119, $errno, $error, 2))

                    $x = fgets($file, 10000); if
                    (substr($x, 0, 3) != 200)

                        print("<br>fgets not 200: " . $x );

                    fputs($file, "list\r\n"); $x
                    =    fgets($file,    10000);
                    if(substr ($x, 0, 3) != 215)

                        print("<br>fgets not 215: " . $x );

                    while(substr($x, 0, 1) != "." and !feof($file) and
                    $limit < $this->limit)


                        $a    =    explode("    ",    fgets($file,    10000));
                        $this->groups[$server][]    =    array("name"    =>
                        $a[0], "start"    =>    intval($a[2]),    "end"    =>
                        intval($a[1]));

                    fclose($file);

                else

                    print ("<br>Cannot fsockopen server " . $server ); }
```

A korlát a Slimit-ben határozható meg. A get_groups()-függvény a $servers-en keresztül olvas, megnyit minden szervert, és begyűjt minden hírcsoportot a szerverről. Egy valós rendszerben szükséged lenne valamilyen módszerre a különleges előtagokkal bíró vagy megadott szavakat tartalmazó hírcsoportok kiválasztásához vagy letiltásához, hogy az embereknek lehetőségük nyíljon csak azokra előfizetni, amelyek valóban érdeklik őket.

Az előfizetési lista elhelyezhető a felhasználói profilban, majd a könnyű elérhetőség érdekében bemásolható a session-rekordba. Kiterjesztheted az osztályt a __sleepQ és a __wake() használatához, hogy az objektum alkalmas legyen a felhasználói session-rekordban történő tárolásra, de légy óvatos az ott tárolt adatok mennyiségével. Ha hírcsoportok ezrei és cikk bejegyzések tízezrei vannak az objektumban tárolva, a session-rekord nem lesz megfelelő. Nézd csak meg a hírcsoportok és cikkek adatbázisban történő tárolását, rejtett fájlként tárolt nagyméretű mellékletekkel. Ha sok hírcsoportokra előfizető felhasználód van, használhatod az egyedi azonosítót minden egyes cikk esetén azok tárolására és az előfizetőkkel történő megosztására.

A fsockopenQ megnyitja a hírcsoportszervert a szervernév, a 119 port, két hibamező és egy másodpercekben mért időtúllépési érték használatával. Minden szerver esetén az időtúllépési értéket és a port számát külön kell tárolni, mivel nem minden szerver használja a 119-es portot, illetve a válasz tovább tart, mint az itt beállított 2 másodperc. (A megállapított két másodperc a szélessávú elérés és a reggeli csendes időszak figyelembevételén alapul; ez az értékmodem használata esetén napközben 10-60 másodperc szokott lenni.)

Az fgets() beolvas egy sort a hírcsoportszerverről, mivel a belépéskor a hírcsoportszerverek elküldenek egy azonosító sort. A sor első három karakterének 200-nak kell lennie, ezzel jelezve a sikeres belépést; bármi más hibát jelent. Az üzenetet azonosító számok az RFC 977-ben vannak leírva.

Az **fputs()** elküld egy sort, amely egy list-lekérést tartalmaz. A list szó bármely karaktere tetszés szerint írható kis- vagy nagybetűvel. Néhány kérés után megadhatsz paramétereket, a paraméterek között és után pedig használhatsz szóközt és tabulátort is. Minden sort kocsivissza és **újsor,** \r\n jellel kell befejezni. Minden bejövő sor végén a \r\n szerepel, tehát ne felejtsd el ezt eltávolítani a sztring adattá történő konvertálásakor. Az fgets()-t azonnal követő fputs() a listakérésre adott válasz megszerzésére szolgál, de győződj meg róla, hogy a válasz értéke 215, jelezve, hogy a válaszsort követi a hírcsoport lista is. A **while()**-ciklus beolvassa a további sorokat, mindaddig, amíg a végén egy ciklussal kezdődő sort talál, jelezve a lista végét. A ciklus sornak .\r\n-nak kell lennie, de nem vizsgáltam meg minden szerverről érkező választ; az egyik szerver hibát okozott, azt jelezve, hogy a szerver néhány sort \n-el küldött el, az RFC 977-ben megszabott, teljes \r\n helyett.

Az explodeQ felosztja a cikk sorait egy tömbre, a szóközöket alapul véve, a $groups-ba helyezi a sorok első három részét, és a sztringből származó két értéket egész számokká konvertálja. Az utolsó lépés a kapcsolat fclose() használatával történő lezárása, illetve ha a függvény hibásan érkezik vissza, a fennmaradó kódnak hibát kell jeleznie. Több lehetőség is van hibaüzenetek és visszajelzések beágyazására, mint amit ebben a kódban megvalósul.

A hírcsoport sor a következőképpen néz ki:

```
php.announce    0000000029    0000000001    •5. fi.
m\r\n php.beta    0000000161    0000000001
n\r\n
```

```
php.cvs    0000006251    0000000001                    '"' PJ90-
n\r\n php.db    0000010539    0000000001
y\r\n php.dev    0000060198    0000000001                ab
y\r\n php.doc    0969336294    0969332401
y\r\n
```

Az első oszlop a hírcsoport neve, amit egy szóköz követ. A következő oszlop az utolsó cikk sorszáma, a harmadik oszlopban szerepel a kezdőcikk sorszáma, és az utolsó oszlopban az m, n vagy az y jelzi a hírcsoport elérhetőségét. Az RFC 977-ben az y jelzi, ha küldhetsz cikkeket a hírcsoportba, az n jelzi, ha nem. Az m nincs definiálva, de valószínűleg azt jelenti, hogy a hírcsoport moderált, tehát küldhetsz cikkeket, de azok mindaddig nem jelennek meg, amíg a moderátor jóvá nem hagyta a megjelenésüket. Az \r\n egy manuális fordítása az egyébként nem látható sorvége és újsor karaktereknek. Vedd észre, hogy egy számsor rendszerint nagy értékekkel rendelkezik, ami valószínűleg hibákat fog okozni a hírcsoportszerver szoftverében.

Próbáld ki az osztályt a következő kóddal:

```
$a = new news;
$a->servers [ ] = "news.php.net";
$a->get_groups ();
```

Az osztályt a $a-objektum létrehozására használjuk, ami a news.php.net hírcsoportszerver használatára van beállítva, a kód pedig a get_groups()-módszeren keresztül olvassa a hírcsoportlistát.

A következő kód egyszerűen listázza a $a->groups-tömbben tárolt hírcsoportokat, és egy HTML-táblázatba tördeli az adatokat az oszlopok rendezett megjelenítéséhez:

```
if (isset ($a->groups) )

    print ("<tablextrxtd>Server</td><td>Group</td>"
        .   "<td>Start</td><td>End</td></tr>") ;
    while (Üst ($s,    $vs)    =   each ($a->groups) )
    ;

       while (üst ($k, $v) = each($vs)) {
    print("<tr><td>" . $s . "</td><td>" . $v["name"]
     . $v["start"] . "</td>"
     . $v["end"] . "</tdx/tr>" );
    }
    }
    print("</table>");
```

A következő lista tartalmazza az összes visszanyert hírcsoportot a news.php.net-oldalról. Vedd észre, hogy a hírcsoportból visszanyert sorokból a második szám volt a kezdő érték, amely most az első helyen szerepel. Ha kiterjeszted ezt a kódot a cikkek hírcsoportba történő elküldéséhez, be kell ágyaznod a y/n/m-jelzőt, hogy a levelet fogadó hírcsoportok láthatóak legyenek:

```
Server          Group           Start      End
news.php.net    php.announce  1            29
```

```
news.php.net  php.béta        1           161
news.php.net  php.cvs         1           6251
news.php.net  php.db          1           10532
news.php.net  php.dev         1           60193
news.php.net  php.doc         969332401   969336293
news.php.net  php.generál     1           58379
news.php.net  php.gtk         1           2255
news.php.net  php.il8n        1           164
news.php.net  php.install     1           3598
```

A következő news2-osztály kiterjeszti a news-osztályt a get_articles()-függvénnyel, a hír-csoportban szereplő minden fejezet beolvasására:

```
class news2 extends news
    {
    function get_articles(Sserver, $group)  . - »■.....
        {                                          .
        $limit = 0;
        if($file = fsockopen($server, 119, $errno, $error, 2))

            $x = fgets($file, 10000);
            if(substr($x, 0, 3) != 200)

                print("<br>fgets not 200: "    $x);
                }
            fputs($file, "group " . $group    "\r\n")
            $x = fgets($file, 10000); if
            (substr($x, 0, 3) == 211)          ;

                do                                    i ( ■-- I 0-Í-- -B l) .

                    fputs($file, "head\r\n");
                    $head = fgets($file, 10000);
                    print("<br>head: " . $head );
                    $y = fgets($file, 10000);
                    while(substr(Sy, 0, 1) != "." and ! feof($file))
                        {                e
                       print("<br> Snbsp;Snbsp; " . htmlentities($y) );
                        $y = fgets($file, 10000);
                        }
                    fputs($file, "next\r\n"); $head =
                    fgets($file, 10000); }
                    while(substr($head, 0, 3) == 223
                        and $limit < $this->limit);
                }
            else                        \-■
                {
                print("<br>fgets not 211:       $x);
                }
            fclose($file);

        else {
```

u 0:

```
print("<br>Cannot fsockopen server      $server);
```

Néhány hírcsoport akár 10.000 cikkel is rendelkezhet, ezért ez a kód korlátozza a lekért mennyiséget. Ne feledd, hogy a hírcsoportszerverek csak korlátozott ideig tárolják a cikkeket. Egy aktív hírcsoportba naponta akár 1 000 cikk is érkezhet, az Internet-szolgáltatód pedig 10 napra korlátozhatja a hírcsoportok megőrzését. Egy felnőttek számára készülő audio- és video-fájlokat tartalmazó hírcsoport esetleg csak néhány órára őrzi meg a fájlokat azok nagy terjedelme miatt.

A kódok négy sora kiemelt jelentőséggel bír, mivel ezek különböznek a get_groups()-függényben levő kódtól. Az első, az fputs() egy lekérést küld a hírcsoport kiválasztására vonatkozóan, és a hírcsoportszerver a kapcsolat teljes ideje alatt emlékszik a kiválasztott hírcsoportra. A második, az fputs() egy head-lekérést küld a szervernek a választott hírcsoport fejlécsorainak letöltésére. A cikk törzsének megszerzéséhez küldhetsz egy body-leké-rést, vagy ha az egész cikket meg szeretnéd kapni fejléccel és törzzsel együtt, akkor használd az article-lekérést. A print()-utasítás segít a fejlécsorok áttekintésében, amíg te összeállítod a kívánt értékek kinyeréséhez szükséges kódot. Az utolsó, az fputs(), a next-kérést küldi el a szervernek a következő cikk hírcsoportból történő letöltéséhez.

Próbáld ki az osztályt a következő kóddal:

```
$b = new news2;
$b->servers[] = "news-server";
$b->limit = 1;
$b->get_articles($b->servers[0] , "alt.php");
```

Cseréld ki a news-server-t az általad használt hírcsoportszerver nevével, és manuálisan határozz meg egy érdekes hírcsoportot, mint amilyen például az alt.php. Vedd észre, hogy az osztály még nem tartalmazza az összes szerver összes hírcsoportjának átnézéséhez és az összes fejléc kinyeréséhez szükséges kódot, mivel ez a fejlécmennyiség nehezen lenne kezelhető. Ehelyett illessz be egy kódot a $groups-tömb megjelenítésére, adj minden sorhoz egy gombot, hogy az emberek előfizethessenek rá, majd pedig tölts le az előfizetett hírcsoportokhoz tartozó fejléceket.

íme a megjelenített első sor, a hírcsoport első cikkéből visszanyerve (a 211 az üzenet száma, ami azt mutatja, hogy a fejléc folytatódik, a szövegtörzs viszont nem):

```
head:  221  19081  <3b4240al@news . kos . net>                    -IOI
```

Az alábbiakban láthatók a hírcsoportban található fejlécek bejegyzései az első cikkből (a sorok a könyv szerkezetének megfelelően vannak tördelve):

```
Path:   news-server.bigpond.net.au!intgwpad.nntp.telstra.net!
        news.stealth.net!news.maxwell.syr.edu!newsfeed.slurp.net!
        news.kos.net
   From: "Online Creator Inc" <jobs@theonlinecreator.com>
   —•—*-Newsgroups: alt.php,ca.jobs,can.jobs,comp.jobs,
        kingston.jobs,linux.jobs,ont.jobs,ott.jobs,tor.jobs
```

```
    Subject: Web Programmer / Web Designer  B'^'IC-: '')■■.-. I'JC:
    Lines: 61                                                  í
    X-Priority: 3                              ■       .       :
    X-MSMail-Priority: Normál
    X-Newsreader: Microsoft Outlook Express 5.50.4133.2400
    X-MimeOLE: Produced By Microsoft MimeOLE V5.50.4133.2400
    X-Original-NNTP-Posting-Host: 216.13.93.203 Message-ID:
    <3b424 0al@news.kos.net>
    X-Original-Trace: 3 Jul 2001 18:01:05 -0500, 216.13.93.203
    Organization: Kingston Online Services Date: Tue, 3 Jul 2001
    18:09:55 D0400 NNTP-Posting-Host: 216.13.25.103
    X-Trace: newsfeed.slurp.net 994197876 216.13.25.103 (Tue, 03 Jul
2001
        17:04:36 CDT)
    NNTP-Posting-Date: Tue, 03 Jul 2001 17:04:36 CDT Xref:
    news-server.bigpond.net.au alt.php:19081 ca.jobs:153181
        can.jobs:505434 comp.jobs:276301 kingston.jobs:11067
:'"*-  " ont. jobs:436593 ott. jobs : 208118 tor . jobs : 392964
                                                «2p : !, íd -■■
```

Az RFC 977-től kaphatsz valamennyi információt a fejlécekről, néhányat pedig a hírcsoportszervereket fejlesztő emberek is összeállítottak. A X-MimeOLE-fejléc például gyanúsan úgy néz ki, mint egy Microsoft-hirdetés. A Path-fejléc segít lenyomozni a news-szerverekkel kapcsolatos problémákat. A Newsgroups elárulja, hogy a cikk többször is szerepel különböző hírcsoportokban, a Message-ID pedig ad egy azonosítót, amely az összes hírcsoportban egyedinek számít, és segítségével kiküszöbölhető a többszörözés. A Subject megjeleníti a weboldaladon a cikk nevét, a From megadja a válasz e-mailcímet. A Lines megmondja, hány sor van a cikkben, így a telefonvonalat használók eldönthetik, hogy elolvassák-e a cikket vagy sem, de a sorok számának ismerete nem szükséges a sorok elolvasásához. A régi levelek törléséhez vagy kihagyásához használhatod a Date- vagy a NNTP-Posting-Date-uusítást. Gyűjts ki több hírcsoportból származó fejléc összeállításokat, hogy láthasd a hasznos és konzisztens fejléceket.

A hírek és cikkek visszakeresése hasonlít a fejlécek letöltéséhez, az ezekre vonatkozó lekérések az RFC 977-ben vannak leírva. A mellékletek is hasonlóak a levelezésben használt mellékletekhez (1. 15. fejezet), a cikkek küldése pedig nagyjából a fordítottja a lekérésnek. Kísérletezz az itt leírt kódokkal, majd látogass a http://freshmeat.net- vagy a http://sourceforge.net-oldalra, ahonnan letölthetsz egy teljes értékű hírcsoport alkalmazást. Egy hírcsoport egy levelezési alkalmazás teljes kifejlesztésének - beleértve a csatolt állományok kezelését is — időszükséglete meghaladja a legtöbb ember lehetőségeit. A saját hírosztály kifejlesztésének lényege az NNTP használatának megtanulása, úgyhogy inkább szerezz be egy kész kódot, és azt boncolgasd tapasztalatszerzés végett. A kísérletezés segít majd megérteni az előre megírt PHP alapú híralkalmazásokat.

Hivatkozás:	oldal:
Levél küldése csatolt állománnyal	527

```
. 3io ,ei' ,íno , .ic{ . y.u-.tl l ..£■■{;■:(;. r.
```

18. fejezet

Keresés

1oJXJd

|lobnorj>

gn;ríai£'jéí„
A

Áttekintés

A World Wide Web meglehetősen sokféle dokumentum és adatbázis keresésére nyújt lehetőséget, de a legtöbb keresőprogram csak a weboldalakra korlátozódik. Ha bármi mást szeretnél megkeresni, ismerned és értened kell a speciális keresési technikákat és eszközöket.

Amennyiben képeket szeretnél keresni, próbáld meg az altavista.com szolgáltatást. Sokszor megpróbálkoztam az AltaVista képkeresőjével, de csak kevés használható eredményt kaptam. A szoftver nem tudja, hogy mi van egy képben, tehát a képet körülvevő szöveg alapján tippel, így ez a technika csak a képek kis hányada esetén működik.

Az információforrások széles választéka érhető el az Interneten, de nem feltétlenül csak weboldalakról, és vannak források, amelyek speciális elérési technikát igényelnek. A Z39.50 protokoll a különleges adatbázisok nagy választékához biztosít hozzáférést, amelyek közül sok könyvtárban van, és hagyományos, papír alapú forrásokat vagy azok tárgymutatóit tartalmazza. Ha régi újságokban szeretnél keresni, használd a PHP YAZ-függvényeivel a Z39.50 forrásokat.

Keresőprogramok felkutatása

A PHP fsockopen()-függvénnyel, a fájlfüggvényekkel vagy a csatolófelület-szolgáltatásokkal (socket services) úgy kereshetsz, mint a yahoo.com-on. Minden Internet-könyvtár és keresőoldal elérhető a szervereddel böngésző emulálásával. Próbaképpen írhatsz egy kódot, amely a keresést a google.com- vagy az excite.com-oldalakra továbbítja, majd az eredményeket visszakeresve megjelenítheted azokat. Ha ezzel csak egyszer próbálkozol, senki sem fogja leszedni a fejed. Ha a saját gépedet egy webszerverhez állítod fel, és a webszervert pusztán csak egy személyes keresőnek használod, az semmiben sem különbözik egy hagyományos kereső alkalmazásától. Ha továbbítod a visszakeresett információkat egy harmadik személynek, megszeged a szerzői jogokat, és hamarosan számíthatsz az FBI vagy a helyi illetékes szervek hajnali 5 körüli látogatására.

Ha egy olyan kódot írsz, amely ciklikusan látogat meg egy szervert, számíthatsz rá, hogy a szerver jelentést fog küldeni az eseményről. Egy szerver nem megfelelő használata DoS (denial of service - szolgáltatásblokkoló támadás) támadásként is értelmezhető, ami 500 000S büntetést is vonhat maga után. De az is előfordulhat, hogy néhány évre hűvösre kerülsz. Gondolj csak a biztonsági mentéseidet tartalmazó floppyra, amit egy öreg, bűzös cipősdobozban tartottál a ruhásszekrény mélyén 10 évig, majd képzeld el, hogy egy ugyanilyen helyen kellene eltöltened a következő 10 évet.

Ha ezek után is végre akarod ezt hajtani, és úgy gondolod, hogy üzleti érdeked fűződik ehhez a vállalkozáshoz, lépj kapcsolatba a kívánt keresőoldallal, hogy fel szeretnéd használni az információikat. A Google és más keresők egy speciális interfészt fognak elküldeni cserébe az anyagi hozzájárulásodért. Néhány keresőoldal-elhelyezésért fizető hirdetéseket kínál, és cserébe a reklámok más helyeken történő megjelenéséért megosztja veled a bevétel egy részét. A „Searching google.com" nevű megoldás megmutatja, hogy hogyan állíts be a ke-

resési teóriáid teszteléséhez egy gyors csatlakozást a google.com-oldalhoz. Bővítsd a kódot az új ötleteiddel, és add el az új interfészt a Google-nak. (A meglévő interfészűk szerintem működhetne a kifejezések keresésének egy logikusabb módján.)

LDAP

A 14. fejezetben bemutatott PHP LDAP-függvényekkel kereshetsz az LDAP-adatbázisokban. Az LDAP-ban az embereket a telefonkönyvhöz hasonló módon listázzák, de sok telefonkönyvszerű alkalmazás használ a háttrében hagyományos relációs adatbázist és SQL alapú keresést, így az LDAP kevés olyat nyújt, amit ne tudnál közvetlenül SQL-ben elvégezni.

Z39.50

A Z39.50 egy az Amerikai Nemzeti Szabványügyi Intézet (American National Standard Institute - ANSI) által hozott szabvány, amelyet a Nemzeti Információ Szabályozási Szervezet (National Information Standards Organization - NISA) hagyott jóvá 1988-ban, a szabvány betartatását pedig a Kongresszusi Könyvtár (Library of Congress) Z39.50-es Nemzetközi Szabványkezelő Ügynökségre bízta (International Standards Maintenance Agency; http://lcweb.loc.gov/z3950/agency/). A Nemzetközi Szabványügyi Szervezet ' (International Standards Organization) ISO 23950 szabványa - „Információ visszanyerése (Z39.50): alkalmazási szolgáltatás és protokoll meghatározás" - technikailag ugyanaz, mint a ANSI/NISO Z39.50.

A Z39.50 egy protokoll, amely lehetőséget ad a számítógépnek egy másik gépen való keresésre, a másik számítógép által használt keresési szintaktika ismerete nélkül. Mind a két számítógépen telepítve kell, hogy legyen a Z39.50 szoftver.

A Könyvtárközi Kölcsönző-alkalmazási Szabványkezelő Ügynökség (Interlibrary Loan Application Standards Maintenance Agency - www.nlc-bnc.ca/iso/ill/main.htm) kiterjesztést fejlesztett ki a Z39.50-hez a könyvtári kölcsönzési kérelmek kezelésére. A bővítések a www.nlc-bnc.ca/iso/ill/stanprf.htm-oldalon vannak PDF-dokumentumok formájában leírva. Nézd meg a "Profilé for the Use of Z39.50 Item Order Extended Service to Transport ILL Protocol APDUs (Z39.50/ILL Profilé l)-et" és a "Profilé for the Use of Parameters from the ILL-Request APDU in Z39.50 Item Order (Z39.50/ILL Profilé 2)"-t. A YAZ yaz_itemorder()-függvényét a kiterjesztések kezelésére tervezték.

YAZ

A YAZ- (Yet Another Z39.50 Toolkit) szoftver és a PHP YAZ-függvények az Index Data-tól (www.indexdata.dk) származnak. Az Index Data a Z39.50 alapú alkalmazásokra szakosodott, weboldalán a PHP-t használja, így a PHP és a Z39.50 házassága logikus lépésnek tűnik.

A YAZ telepítése

A YAZ a PHP egy új kiterjesztése, amelyet a PHP4.0.6 már tartalmaz, ezért telepítsd a legújabb PHP-t: először telepítsd a legfrissebb PHP-t, teszteld vele a meglévő kódot, majd telepítsd a YAZ-t.

Windows 98 ME és újabb verziók

A YAZ Windows 98 ME vagy újabb verziója alá történő telepítéshez a következőket tedd:

1. Másold a c:\Program Files\php\extensions\php_yaz.dll-fájlt a c:\windows\system\-könyvtárba.
2. Állítsd le a webszerveredet.
3. A php.ini-ben távolítsd el a pontosvesszőt az alábbi sor elejéről:

```
extension=php_yaz.dll
```

4. Indítsd el a webszerveredet.

WindowsNT és Windows 2000

A YAZ Windows NT és Windows 2000 alá történő telepítéséhez a következő lépéseket vé-lgezd el:

1. Másold a c:/Program Files/php/extensions/php_yaz.dll-fájlt a c:/winnt/system32/-könyvtárba.
2. Állítsd le a webszervert.
3. A php.ini-ben távolítsd el a pontosvesszőt a következő sor elejéről:

```
extension=php_yaz.dll
```

4. Indítsd el a webszervert.

Unix

A YAZ Unix alá történő telepítéséhez kövesd az alábbi lépéseket:

1. Töltsd le a legfrissebb YAZ-szoftvert az IndexData-oldalról (www.indexdata.dk/yaz/).
2. Fordíts és telepítsd a YAZ-t.
3. Fordítsd be a PHP-t a —with—yaz-zel.

Adatforrások

A YAZ-adatforrás egy kereshető adatbázis - az adatbázis rendszerint egy könyvtárban van, az pedig sokféle adatforrással rendelkezhet. A BookWhere rendelkezik egy adatforráslistával, amelyet a www.bookwhere.com/library.htm-oldalról tölthetsz le, de ez elavultnak fog tűnni. Az IndexData is fenntart egy listát a www.indexdata.dk/targettest/-oldalon, és a források elérhetőségét rendszerint tesztelni is szokták.

Az IndexData tesztelési céljai listájának egy része a 18.1 ábrán látható, egyik eleme a Bell Labs. A Bell Labs olyan elsődleges kísérleteket végez, amelyek technológiánkat mozgásban tartják, így adatbázisukban rengeteg lebilincselő dokumentum kell, hogy legyen. A Bell Labs olyan hasznos szerkentyűket talált fel, mint a tranzisztor, a lézer, és amennyire a fizikához értek, szerintem az elektront is ők fedezték fel.

Target Db Name Name	Access	Address **Port**	Record Syntaxes	Services	Explain Categories	Bib lüse Attributes
Bavanan State Libranes SS	**:o%**	193 174.96.24 31310	SUTRS Ummarc	search prezent delSei, resourceReport, scan,' namedResultSets		37-41,44,46, 48.53,59, 1003-1006, 1008-1009, 1018
Belqorod 'State übrary *	97%	liberbgunbru i 210	SUTOS USmarc	search, present. scan	**Nőne**	1-4,7-8,21. 30-32,59.63. 1003, 1011-1012. 1018,1035
Bell books \ Laboratories **Library Network** i		z3950.bell-labs.com 210 USmarc GRS-1	I *SUmS*	search, present, accessCtrl, scan, exlendedServices, level-2Segmentation, ramedResultSets	CategoryLísl, TargetInfo, DatabaseInfo, SchemaIrfo, AtiributeSetInfo	1-63,1000-1028 , 1030-1036

nos

18.1 ábra Elérhető Z39.50-szerverek index-adatlistája

Tesztelés

ion

A fejezetben szereplő YAZ-függvények nagy része újdonság a PHP 4.0.6-ban, így kevés a dokumentáció, és átfogó példa van rájuk. Többször kell majd a kódodat tesztelned, mmtha régebbi és jobban ismert függvényeket használnál.

A **http://lcweb.loc.gov/z3950/agency/resources/testport.html-oldalon** találsz egy listát a tesztoldalakról. A következő minta leír egyet az ebben a fejezetben általam példaként használt forrásokból. Egyrészt azért választottam ezt a forrást, mert a kereshető információk széles skáláját nyújtja, másrészt mert Dél-Ausztrália meglehetősen érdekes része a világnak, és néhány remek bornak a hazája. Olyan embereket fogok keresni, mint Dr. Christopher Rawson Penfoldot, a világhírű Penfolds-borok termelőjét (www.penfolds.com.au/), illetve Max Schubertet, a pompás Penfolds Grange termelőjét:

```
The State Library of South Australia
   Address : 143.216.21.3                    dl •• -9 Iт«
   Port: 210

Databases available for testing:
   innopac arch pict                                    írn

Searching:
   MAIN : author,title,subject,word
   PICT : word,title,name ARCH :
   word,title,name
```

■*yA* XAY

.AY

```
MAIN : Books, newspapers, magazines, periodicals, non-print and
electronic library materials held in the Bray Reference Library,
Mortlock Library of South Australiana and Rare Books and Named
Collections

PICT: Photographs and other pictorial matériái from the Mortlock Library's
collections, and references to articles and other citations from selected
South Australian published matériái held in the Library

ARCH: This database contains Personal, business and society archives and the
J.D.Somerville Oral History Collection

Contact: Elvio Pederzolli                                          ;
```

RPN

Fordított előtag jelölést (Reverse Polish Notation - RPN) a korai HP-számítások során használtak egyes számítások gyorsabbá tételére. Az RPN-ben az A+B-t AB+ formában tudod bevinni. A YAZ-függvényekben az RPN-hivatkozás valójában az RPN-jelölés fordí tottját jelenti, ahol az A+B bevitele +AB formában történik. Megjegyzem, hogy a művelet kerül az első helyre, amit a műveleti elemek követnek. A YAZ a fordított RPN-jelölést használja a kereső sztringre, és ez az egyetlen logikátlan része a szoftvernek, ami hibákat okozhat. . ,-■ .....„„.

89-

Keresés YAZ-val

A YAZ lehetővé teszi a Z39.50 protokoll használatával a hivatkozások adatbázisokban való keresését. Egy időben több adatbázisban is kereshetsz vele, illetve az eredményeket több formátumban is lekérdezheted. A keresési típusa attól függ, hogy mit fogad el az adatbázis, az eredmények formátuma pedig az adatbázistól függ. A Z39.50 protokoll nem szünteti meg a keresési típusok egyezésének és az eredményformátum adatbázissal történő illesztésének igényét. A YAZ- és a Z39.50-protokoll nem ad módot a keresési típusok és az eredményformátumok intereaktív kialakítására.. Ez a YAZ-en elvégzendő következő fejlesztés.

Nem támogatott keresés - *

Az *Unsupported search* egy általános hibaüzenet, amivel az adatbázisokon történő első keresések során fogsz találkozni. Ennek egy érvénytelen keresési típus vagy szintaktika lehet az oka. Van lehetőség a hibákkal kapcsolatos többletinformáció megszerzésére, bár az általam megvizsgált források nem használták ezt a támogatott keresési típusok listázásához.

YAZ-függ vények

Ebben a részben a PHP 4.0.6 YAZ-függvényeket mutatom be. Az újkeletű YAZ-kiterjesztés és az „önvizsgáló" függvények hiánya a YAZ-függvények skálájának bővülését sejteti. Az IndexData, a YAZ kifejlesztője segíthetne a világon, ha a saját online Z39.50 adatforrás listáját YAZ által kereshető adatbázissá tenné, így a felhasználók a YAZ-forráslis-tákat a YAZ-en keresztül sajátíthatnák el.

yaz_connect()

A yaz_connect() egy ellenőrző struktúrát állít be, amit egyéb függvényekkel kell megtölteni, majd egy szerverhez kapcsolódik a yaz_wait()-függvénnyel történő kereséshez. A második paraméter egy opcionális hitelesítő sztring a függvény használatának néhány példájában a többszörös sztnngek neveket és jelszavakat tartalmaznak. Az itt bemutatott első sztring egy szerver nevét vagy IP-címét tartalmazza, amit a port száma és az adatbázis neve követ. A port száma opcionális, alapértelmezésben 210; mivel a legtöbb szerver a 210-es portot használja, ritkán lesz rá szükséged. Amennyiben kihagyod a port számát, a kettőspontot is hagyd el. Az adatbázis is opcionális paraméter, a kapcsolat alapértelmezésben az oldal alapértelmezett adatbázisát fogja használni.

A yaz_connect() egy csatlakozásazonosítót ad vissza, ha a kapcsolat sikeres, illetve egy nullát, ha nem. Ezért mentsd el a yaz_connect()-ból származó eredményt, és készíts egy hibaüzenetet arra az esetre, ha a kapcsolat sikertelen. Minden egyéb YAZ-függvény számára a csatlakozásazonosító az első paraméter:

```
if($connection = yaz_connect("143.216.21.3:210/pict", "auth"))
```

íme ugyanaz a kapcsolat a 210-es portszám felesleges megjelölése nélkül:

```
if ($connection = yaz_connect("143.216.21.3/pict", "auth"))
```

yaz_close()

A yaz_close() bezárja a yaz_connect() által létrehozott kapcsolatot, ahol a yaz_connect() által visszajuttatott egyetlen paraméter a csatlakozásazonosító:

```
yaz_close($connection) ;
```

yaz_syntax()

A yaz_syntax() csatlakozásazonosítót és a keresési eredményekhez a szintaktikát megnevező sztnnget fogadja el. Amennyiben érvénytelen keresési sztringet adsz meg, az alapértelmezés usmarc-ként jelenik meg. A többi opció a grsl-et, sutrs-t, és xml-t tartalmazza is. A függvény ugyanazt az eredményt adja vissza, függetlenül attól, hogy mi történik, ezért nincs értelme az eredmények tesztelésének:

```
yaz_syntax($connection, "xml");
```

yaz_search()

A yaz_search() egy csatlakozásazonosítót, egy rpn-t tartalmazó sztringet és egy kereső M. sztringet fogad el a következőképpen:

```
if(yaz_search(Sconnection,    "rpn",    $search))
```

Az eredmény megfelelő működést vagy hibát jelez. A második paraméter a lekérdezés típusát határozza meg, de jelenleg csak az rpn-t fogadja el. Az RPN keresési formátum leírását az www.php.net/manual/en/function.yaz-search.php-oldalon találod.

A teszteléshez használhatod a penfolds vagy a "new york"-hoz hasonló egyszerű kereső sztringeket:

```
$search = "@and penfolds V'new york\"";
```

A szóközt tartalmazó keresési értékeket idézőjelbe kell tenni. A többszörös keresési kritériumokat összekapcsolhatod az RPN-jelöiés használatával. A **@and** azt jelenti, hogy a sorron következő kritérium az and-hez van kapcsolva, illetve a @or-kritérium az or-ral, a (Snot pedig az and not-tal kerül összekapcsolásra.

yaz_wait()

Az alábbiakban bemutatásra kerülő, paramétereket nem igénylő **yaz_wait** megvárja az összes keresés befejeztét. Létrehozhatsz többszörös csatlakozásokat, indíthatsz összetett kereséseket, ezután csak a keresések végét kell kivárnod. A yaz_wait()-ben nincs olyan lehetőség, hogy megvárd egy keresés befejeződését, miközben a többit működni hagyod.

```
yaz_wait
```

yaz_error()

A keresésre való várakozás után a hibákat a yaz_error()- vagy a yaz_errno()-függvényekkel ellenőrizheted (lásd "yaz_errnoQ"). A yaz_error() a csatlakozásazonosítót fogadja el, majd a legfrissebb hibára vonatkozó hibaüzenetet adja vissza. Egy zéró hosszúságú sztring jelzi, hogy nem történt hiba, bármi más esetén igen. Mentsd el a hibasztringet, teszteld a hosszát, majd jelenítsd meg, ha az nagyobb nullánál:

```
$error = yaz_error($connection);
if (strlen(Serror))

   print("<br><font color=\"red\">" . $error . "</font>");
```

yaz_errno()

A **yaz_errno()** csatlakozásazonosítót fogad el, és a legutolsó hiba hibaszámát adja vissza. A nulla nem jelöl hibát, a pozitív szám olyan egyszerű hibát jelent, mint például a szintaktikai hiba, a negatív szám végzetes hibát, például a kapcsolat elvesztését jelenti. Mentsd el a hiba számát, teszteld, és ha nem nulla, jelenítsd meg. Jelenítsd mega a hibás sztringet a **yaz_error()-ból** a hiba számával:

```
$errno = yaz_errno ($connection) ;
if($errno != 0)

      print("<br><font color=\"red\">"    $errno          Serror
          . "</font>");
```

yaz_addinfo()

A **yaz_addinfo()** feladata további információ megjelenítése hiba esetén, vagy egy nulla hosszúságú sztring visszajuttatása. Úgy vettem észre, hogy a **yaz_addinfo()** időnként meg ismétli, amit a yaz_error() visszajuttat. A következő kód által a yaz_addinfo() egyszerűen hozzáadható a yaz_error()-sztringhez azon szerverek számára, amelyek a yaz_addinfo()-n keresztül juttatnak vissza használható információkat:

```
$error = yaz_error($connection); if
(strlen($error))

   $error .= "<br>" . yaz_addinfo($connection) ;
   print ("<br><font color=\"red\">" . Serror .
                                                </font>"
```

yaz_hits() ^ w v

A yaz_hits() csatlakozásazonosítót fogad el, és a kijelölt adatbázisokban szereplő találatok számát vagy nullát ad vissza, ha nem volt találat. A yaz_wait() és a hibaellenőrző kódod után használd a következő kódot a találatok számának a megjelenítéséhez (ha egyáltalán vannak):

```
$hits = yaz_hits($connection) ;
if($hits > 0)

   print("<br>Hits:          $hits)
                          ;
```

yaz_record()

Ha egy keresésből találatokat eredményez, használd a yaz_record()-függvényt, hogy meg kapd az eredményeket. A yaz_record() csatlakozás-azonosítót, a találatok számát, és egy sztringet fogad el, amely az eredmény formátumát jelöli ("string" vagy "array"). A találatok 1-től a yaz__hits() által adott értékig vannak számozva, és a következő for()-ciklussal vé gigmehetsz az eredményjegyzéken. Nem minden találat tartalmaz eredményeket, ezért ellenőrizd a jegyzékben szereplő eredményrekordok hosszát, majd hagyd el a nulla hosszal rendelkezőket:

```
for{$h = 1; $h <= $hits; $h++)
   $hit = yaz_record($connection, $h, "string");
```

yaz_range()

A yaz_range() csatlakozásazonosítót, a keresési eredményrekord kezdő számát és a , visszakeresendő rekordok maximális számát fogadja el. Használd a yaz_range()-et a yaz_search() után és a yaz_present() előtt, a yaz_present() alapértelmezett beállításainak -kezdő szám 1, visszajuttatott rekordok száma 10 - megváltoztatására. A következő példa közli a yaz_present()-tel, hogy a az eredményeket a 25. rekorddal kezdje, és összesen 20-at adjon vissza:

```
yaz_range($connection, 25, 20);
```

yaz_present()

A yaz_present()-et a yaz_range() után és a yaz_wait() előtt szokás használni, egy eredménykészlet előkészítéséhez. A yaz_range() és a yaz_present{) kombinációját csak abban az esetben kell használni, ha meg szeretnéd változtatni a kijelölés hatáskörét a yazjrangeQ-függvényen keresztül:

```
yazpresent();
```

yaz_database()

A yaz_database() lehetőséget ad a yaz_connect()-ben meghatározott adatbázis felülírására. A yaz_database() csatlakozásazonosítót és sztringet fogad el, utóbbi az adatbázis neveket (+) jellel elkülönítve tartalmazza. A következő kód egy tömbben tárolt adatbázislista alkalmazásával végrehajtja a yaz_database()-t. Ez a függvény igazat ad siker, hamisat hiba esetén:

```
$db[]  =  "books";
$db[]  =  "newspapers";
if(!yaz_database($connection, implode("+", $db)))   * \
    f
    print("<br><font color=\"red\">yaz_database failed. " . $error
        . "</font>");
```

yaz_element()

A yaz_element() csatlakozásazonosítót és egy elemkészletjelzőt fogad el, utóbbit az F (teljes) és B (rövid) jelölésekkel. A következő kód az elemkészletet teljesre állítja. Használd a yaz_element()-et a **yaz_search()** után és a yaz_present() előtt:

```
if (!yaz_element ($connection, "F"))

    print ("<brxfont color=\"red\">yaz_element failed. " . $error .
        "</font>");
```

yaz_scan()

A yaz_scan()-függvényt hasonló módon használhatod, mint a yaz_search()-t, az eredményeket pedig a **yaz_scan_result()** segítségével kapod meg. A yaz_scan() csatlakozásazonosítót, egy vizsgálati paramétertípust, ami jelenleg csak az **rpn** lehet, egy vizsgálatkezdő paramétert, amelynek ugyanaz a formátuma, mint a yaz_search() keresési paraméternek, illetve opcionális jelzőket fogad el. Az opcionális jelzők használatára nem találtam példát. A következő vizsgálat a *cat* szóval kezdődik:

```
yaz scan ($connection, "rpn", "cat");
```

yaz_scan_resuit()

Ez a függvény csatlakozásazonosítót és egy opcionális tömböt fogad el, majd egy a **yaz_scan()** által létrehozott, az eredményeket tartalmazó tömböt ad vissza. Az opcionális tömb kap néhány mezőt a vizsgálatból: a **number** tartalmazza a belépési eredmények számát, a **stepsize-,** a **position-** és a status-mezőket:

```
$array  =  yaz_scan__result ($connection) ;
```

yaz_ccl_conf()

Ez a függvény konfigurálja a CCL lekérdezésértelmezőt, de a CCL-értelmezővel kapcsolatban nincs túl sok részlet, és én sem találtam példát a használatára, így nyugodtan átugorha-

tod ezt a függvényt és a yaz_ccl_parse()-t, amíg a fejlesztők kidolgoznak néhány példát. A CCL (Common Command Language) Általános Parancs Nyelv az ISO 8777 szabvány sze rint van meghatározva; néhány Z39.50 szoftver a CCL-t az RPN Z39.50 változatára fordít ja le:

```
$config[]  =  "";
yaz_ccl_conf($connection,  $config);
```

yaz_ccl_parse()

A yaz_cd_parse() a CCL formátumú find-lekérdezést RPN-formátumba konvertálja a yaz_search() részére:

```
$array  =  arrayO;
if(yaz_ccl_parse($connection,  "query",  $array))
    {
    yaz_search($connection, "rpn",$search);
    }
else
    {
    print("<br>ccl parse error.");
```

yaz_itemorder()

A yaz_itemorder() egy kiterjesztést végez el a Z39.50-en a könyvtárközi kölcsönzésekre. Egy halom paraméter van elhelyezve egy tömbben, majd a tömb a yaz_itemorder() függvénybe van betáplálva. A példa csak egy a tömbbe kerülő könyv ISBN-számát tartalmazza. Ha szükséged lesz ennek a használatára, valószínűleg a könyvtár által szolgáltatott dokumentumhivatkozásokhoz fogsz jutni:

```
                          "1588800539";
$order["item-id,ISBN"] =
yaz itemorder($order) ;
```

Adatbázis alapú keresések

Hogyan állítanál össze egy keresőmotort, ha az alapoktól kellene kezdened: Itt van néhány, több honlaphoz használt keresőmotorra és a web létrehozása előtti online rendszerekre épülő ötlet.

Adatok megőrzése eredeti formájukban

Néhány ember szeret mindent egybepréselni, hogy egy adatbázishoz jusson, és mindent átnevez, hogy illeszkedjen az elképzeléseihez. Ha adatokat kapsz, lehetőség szerint tartsd meg őket eredeti formátumukban és osztályozásuknak megfelelően.

Vegyük például a CRC32 adatokat. A 32 bites CRC-k előjel nélküli 32 bites egész számok, vagy 32 bites előjel nélküli hexadecimális számok formájában vannak tárolva. A PHP 32 bites előjeles egész számokat használ és CRC32-adatokat jelenít meg, ami eltér attól, amit a legtöbben a CRC32-adatoktól várnak. A CRC32-adatok feldolgozásának az eredménye

nem megjósolható, még akkor sem, ha a kódot személyesen ellenőrződ. Meg kell győződnöd róla, hogy a beérkező adatok adatbázisban történő tárolása az eredeti formátu--í mukban történik, akkor is, ha olyan mezőtípust kell használnod, ami több lemezterületet használ, mint az adatbázisra vonatkozó elméleti minimális mezőméret.

Most nézzük az eredeti adat nevét. Valami olyasmi volt a név, mint a CRC32h? Aki az ada tot rögzítette, valószínűleg hozzáadott egy *h-x,* annak jelölésére, hogy az adatot hexadeci mális formában tárolták. Mind a mező neve, mind a formátuma jelentheti azt, hogy az ada tokkal együtt kell tárolni. ' -3 '. a ■ . ■

Hogyan tudsz több forrásból származó CRC32-adatot tárolni az eredeti formátummal ki-egészítve? Az egyik megközelítés az adatok sztnngben történő tárolása és formázása. Ezt a megközelítést alkalmazzuk a serialize() esetében is (lásd 17. fejezet), ami az egyik módja a rendezett adatbázisokba nem illeszkedő adatformátumok másolásának. Egy hexadecimális CRC32-mező h:ef014ec2-formában is tárolható, ahol a h: jelenti a hexadecimális formátumot. Ha a bemenet egész szám, akkor azt i:12345678-formában lehet tárolni. Másik megoldást jelent, ha két mezőt használunk, nevezetesen CRC32h-t és CRC32i-t. Ha a bemenet egész szám, CRC32i-ként, ha hexadecimális, CRC32h-ként tárold.

A CRC32 elég gyenge megoldásnak látszik, mivel két különböző formátumban kell tárolni az adatokat. Miért pazarolnád a helyet? Találkoztam már a hexadecimális kódba történt nem megfelelő konvertálásnál előfordult hibákkal, amelyet a PHP 32 bites előjeles egész számai, illetve 32 bites egész számok ábrázolása okozott az adatbázisokban. Az ilyen ábrázolási problémák sokkal összetettebbé váltak a telefonszámok esetén, ahol minden ország különböző formátumot használ. Az érvényes e-mailcímeket visszautasítják azok a rendszerek, amelyek az általuk ismert e-mailcímformátumot feltételezik helyesnek. A személyneveket azok a rendszerek tekintik hibásnak, amelyek korlátozzák a nevek hosszát, illetve megszabják azok rendjét és számát. *["UüZl* ,fc.: 71733 :

Úgy kell tervezned az adatbázisodat, hogy a tárolt adatok a lehető legjobban megközelítsék a formát, amelyben ezekhez az adatokhoz hozzájutottál, és bármely fordítást vagy egyszerűsítést új, elkülönített adatok formájában tárolj.

,¥< Rugalmas adatok ,,01om6?91,ivsa3S,tó n mi

Mikor kérdőívet töltesz ki, milyen gyakran találkozol olyan pozícióval vagy munkakörmeghatározással, amely valóban fedi a te munkádat? A nyomtatványon kell, hogy legyen egy „Egyéb mező", ahol az emberek új és pontosabb értékeket adhatnak hozzá a felsoroláshoz. Egy eredményül előálló adatbázisban a kiválasztott munkakör meghatározást és az Egyéb mezőt is tárolnod kell, így bárki elemezheti az alternatív javaslatokat, hogy javítsa kérdőíved pontosságát.

Hogyan határozol meg egy olyan mezőgazdasági motorbiciklit, amelynek hat kereke van, j egynél több embert szállít, esőálló összecsukható teteje jobb a többi átalakítható autóénál, és ugyanolyan jól közlekedik vízen, mint szárazföldön? Ha valaki az adatbázisban keres, motorra vagy hajóra klikkeljen? j A(

El kell gondolkoznod az adatbázisodba beáramló adatokon, valamint azon, hogyan nyerhetik ki a látogatók ezeket az adatokat. Vegyél rá néhány embert, hogy egy tesztadatbázisban

végezzenek kereséseket, és vezess naplót minden keresésről. Adj lehetőséget az embereknek, hogy jelezhessék a sikeres kereséseket. Ulj oda, és nézd te is, hogy dolgoznak. Találd ki, milyen adatjellemzőket szeretnének a felhasználók, és te hogyan tudod ezeket adatbázisodba bevinni.

Rugalmas keresések

Sok weboldal keresőmotorja nem veszi figyelembe a HTML-t és más kiegészítő jelölést, és csak a tartalmat veszi ki. Ez megnehezíti olyan oldalak keresését, amelyek JavaScript-et, Flash-t vagy más technológiákat tartalmaznak. Néhány keresőmotor használja a meta-adatokat, mások a címek miatt elhagyják ezeket. De miért nem tartjuk meg mindkettőt, és tesszük diszkrét mezőkként vagy aggregált oldalszöveg tartalomként elérhetővé őket?

Mikor még nőtlen voltam, és ajándékba selyem fehérneműt kerestem, ki akartam zárni a metaadatokat a keresésből, mert több millió pornográf weboldal tartalmazta vagy a *fehérnemű* (hngene) vagy a *selyem* méta tag-eket. Mikor speciális technológiával létrehozott vagy különleges emberek által tervezett weboldalakat keresek, a metaadatokat is be akarom venni, különösen az olyan tag-eket, mint a <meta name="author>. Nem ismerek olyan keresőoldalt, ahol én választhatnám ki az általam bevenni kívánt méta tag-eket.

Az adatok oztályozása

Végtelen számú próbálkozás történik arra, hogy az adatokat egy jelentéktelen informá-ciópöffenetre szűkítsük, majd sztringről sztringre párosítva kinyerjük a bejegyzéseket. Az eljárás működik, de nem olyan jól, mint a teljesen osztályozott adatok párosításánál, és egyik megközelítés sem olyan jó, mint a kettő együtt. Mikor valaki egy szerzőt keres, hogy könyveit megtalálja, a tény, hogy a keresés célja az adatbázis szerzőmezője, már segítség. A szerző foglalkozása viszont nem fog segíteni, ha a szerző életrajzát keressük, mert a szerző nevére való keresés a Szerzőmező alatt nem találja meg az életrajz szerzőjét.

Mikor a beérkező adatokat osztályozod, hajts rá és osztályozz. Nem kell, hogy az osztályozás tökéletes legyen, és nem kell minden adatelem számára megtalálnod az egyetlen üdvözítő osztályozást. Egy önéletrajzban a szerző megjelenhet a mű szerzőjeként, szereplőjeként, és a kutatások listájában is. A szerzőnek ezenkívül ugyanebben a műben lehet megegyező nevű apja vagy fia. Mikor valaki név alapján keresi a szerzőt, küldd vissza a keresés eredményeit, és ha nincs vagy csak kevés találat van, adj lehetőséget a keresőnek, hogy ugyanarra az értékre minden sztringadatra elvégezzen egy általános keresést. Kezdd a meghatározottakkal, és a kevésbé pontosak felé haladj, míg a kereső sikert nem jelez.

Csökkenteni, de meghagyni

Mikor autókra vagy motorkerékpárokra állítunk fel adatbázist, a tapasztalatlanabbak kihagyják a Kerekek száma részt, mivel minden autónak négy és minden motornak két kereke van. Mikor az adatbázist már széles körben használják, és nehéz rajta változtatni, valakinek be kell vinni a 3, 4 és 6 kerekű motorokat és autókat egészen 10 kerékig.

Lecsökkentheted a szervereden átáramló információkat, de ne akard eltávolítani azokat az adatokat, amelyekről úgy tűnik, hogy történetesen nem használnak. Mikor az információk - a járművekről bekerülnek az adatbázisodba, állíts fel egy adatbázist, amely mindent tartalmaz, és egy másikat is, amely csak azokat a mezőket és adatokat tartalmazza, amelyeket szeretnél online elérhetővé tenni. A teljes adatbázis egy olcsó, lassú szerverre kerül, nagy, lassú és olcsó lemezeken. Az online alcsoport egy gyors szerverre megy, drága, gyors lemezekkel. Ez a látogatóid weboldalait szolgálja, míg az offline adatbázis segítségével új weboldal-lehetőségekkel kísérletezhetsz, olyan információkat használókkal, amelyekről te azt gondoltad, feleslegesek.

' Szabad szövegasszociációk használata

Ételnek számít a krumplicukor? Egy online élelmiszerbolt felállításánál valaki biztosan kimutatja, hogy némely krumplicukornak olyan magas a cukortartalma, hogy az már édességnek számít. Meg akarod tartani az élelmiszerek általad kialakított formális osztályozását és egy szabad szövegmezőt is, ahol az emberek megjegyzéseikkel segíthetnek megtalálni a megfelelő termékeket. Valaki majd panaszkodik, hogy a paradicsom a zöldség kategóriában szerepel, ezért adj olyan felsorolást, hol a paradicsom a gyümölcsöknél és zöldségeknél is megtalálható. Van Só hozzáadása nélkül kijelölő doboz az élelmiszerkeresőben? Ha nincs, tegyék ezt munkatársaid a szabad szövegmegjegyzés mezőbe, és add hozzá azokhoz a sztringekhez, amelyeket az emberek kereshetnek.

Vevőszolgálati munkatársaidnak pótlólagos szabad formájú inputként kellene használni a felhasználói megjegyzéseket a termékekről és kategóriákról az adatbázisodban, míg nem alakítasz ki egy formális adatbázis-szolgáltatást az extra információk keresésére.

Készíts nagy tárgymutatót

Nincs rá ok, hogy az adatbázisod indexe kisebb legyen, mint maga az adatbázis. Egyszer felállítod az indexet az oldalra, aztán az emberek többször felkeresik azt, így az index frissítésére fordított idő nem anyagi természetű. Ha egy *PHP és Weboldal architektúra* című oldalt akarsz indexelni, azt szeretnéd, hogy az emberek közvetlen indexkereséssel találják meg az oldalt, és ne indexvizsgálással. Ez a lassú, költséges SQL like-paraméter elhagyását jelenti. Ne térj vissza az oldal címének SQL általi keresésére, ami a where title Üke '%php%'-ot tartalmazza.

f A *PHP és Weboldal architektúra* címet így helyezd be az indexbe:

```
php and web site architecture
php and web site
php web site
php
web site architecture
_ web site
architecture php
architecture web site
```

Gvűjts össze minden elképzelhető kombinációt, hogy pontos egyezéssel megtalálhatók legyenek. Az SQL lassú **liké** scan-jét hagyd meg utolsó lehetőségnek, ha esetleg nincs pontos egyezés. Jegyezz minden esetet, amikor nem sikerül a találat, majd próbáld kitalálni, mit keresett a felhasználó, és hogy segíthetnél megtalálni. Finomíts az indexépítés rutinján annak érdekében, hogy minimalizáld az SQL like-ra szorítkozó keresések arányát.

Gyors megoldások

Keresés egy szerveren

A YAZ segítségével egy időben egy vagy akár több szervert vagy adatbázist is kereshetsz vagy vizsgálhatsz. Ez a megoldás egy adatbázist kezel és a search()-utasítást alkalmazza, **mert néhány** adatforrás csak a **search()** használatát engedi meg, a scan()-ét nem.

search() nh fc[ru fb; , ^ ; ,1 i b.?

Az első lépés, hogy létrehozunk egy függvényt vagy objektumot az adatbázis keresésére a kereső sztring, a kereső sztring szintaktikai típusa, a szerver neve vagy IP-címe, a port és az adatbázis használatával. Az egyetlen eredmény, amit várunk, a keresés kiírásra kész eredménye vagy egy kiírható hibaüzenet lesz. Az egyetlen output-elvárás **miatt objektum** helyett én **inkább** egy függvényt választottam. A következő kód az eredmény, amit a **Dél-Ausztráliai Állami Könyvtár és** a **Bell** Labs adatbázisán is teszteltem:

```
function search($site, $database, $search, $syntax = "", $port
= "")

    $result = "";
    Starget = Ssite;
    if(strlen($port))

        $target .= ":" . $port;

    $target .= "/" . $database;
    if (! strlen ($syntax) )

        $syntax .= "sutrs";

    if ($connection = yaz_connect (Starget) )

    yaz_syntax ($connection, $syntax) ;
    if(yaz search(Sconnection, "rpn", $search))

    yaz_wait();
    $errno = yaz_errno ($connection) ; | ..'
    if ($errno == 0)

        $hits = yaz_hits ($connection) ;
        $result .= "Hits: " . $hits;
        if ($hits > 0)

            $record = "array";
        if (Ssyntax == "sutrs"
    or $syntax == "xml")

            $record = "string";
```

```
$result .= "<table cellspacing=\"O\" cellpadding=\"O\">" .
    "<trxtd>Hit</tdxtd> </tdx/tr>\n";
for($h = 1; $h <= Shits; Sh++)

    $hit = yaz_record(Sconnection, $h, $record);
    if(is_array($hit))
        {
        $result .= "<tr><td align=\"right\"" *3 .
            " valign=\"top\">" . $h . "</td><td>" . .
            array_display ($hit) . "</tdx/tr>\n";

    elseif (strlen($hit))

        $result .= "<trxtd align=\"right\" "
            . " valign=\"top\">" . $h . "</tdxtd>" .
            htmlentities($hit) . "</td></tr>\n";


$result  .=  "</table>\n";


else {
    $result .= "Wait failed. Error: " . $errno . "
        . yaz_error($connection) ; $add =
    yaz_addinfo($connection); if
    (strlen($add))

        $result .= "<br>  " . $add;


    }
else

    $result .= "Search failed.";

yaz_close($connection);

else

    $result .= "Connection failed.";

return($result) ;
```

A kód első része a szerver nevét, a port számát, és az adatbázist kombinálja egyetlen sztringbe, a Starget-be, a yaz_connect()-ben történő felhasználás végett. A port száma opcionális, mert alapértelmezett értéke a 210, a legtöbb adatforrás pedig éppen a 210-es portot használja. A **yaz_connect()** eredménye a **Sconnection,** amely az első paraméter lesz minden további YAZ-függvényben.

A $syntax-mező szintén opcionális, alapértelmezett értéke sutrs, mert a YAZ-dokumentá-ció szerint ez a legelterjedtebb szintaktika. Ha végignézünk számtalan forrást, tényleg az usmarc tűnik népszerűbbnek. A yaz_syntax() közli a kapcsolattal a helyes szintaktikát.

A yaz_search() megadja a keresés típusát, az rpn-t, és a search sztringet a kapcsolathoz. A yaz_wait() vár a keresés befejezésére, a yaz_errno() jelez mindenfajta hibát, a yaz_hits() pedig a keresés találatainak számát mutatja.

A találatokat úgy kapod meg, hogy az eredményeket végigolvasod a yaz_record()-dal. A yaz_record()-ot egy for()-ciklusba kell ágyaznod, hogy az eredményeket 1-től a találatok számáig mind megkapd. A keresés alapértelmezésben az első 10 találatot adja vissza, hacsak nem növeled meg ezt a számot a yaz_range() használatával, amivel ebben a megoldásban nem próbálkoztunk. Figyeld meg, hogy a yaz_range() és a yaz_present() együttes haszná latával létrehozhatsz egy olyan eredménymegjelenítési rendszert, ahol az első 10 találatot az első oldalon látod, a többi eredmény között pedig előre-hátra mozoghatsz oldalról oldalra. Olvasd el a 19. fejezetet, és annak alapján tárold el a látogatók keresési profiljait egy session-rekordban, hogy megbízható keresési lehetőségeket kínálhass. A felhasználók adat forrás-választásait is eltárolhatod a profilokban, kiküszöbölve ezzel a gyakori ismételt be adásokat,

A yaz_error() kiírja a yaz_errno() által kiadott hibaszámokhoz tartozó hibaüzeneteket, a yaz_addinfo() pedig plusz hibainformációkat közöl. Nem mindegyik szerver ad extra hiba-információt, néhány pedig egy plusz információs mezőben jeleníti meg a a hibaüzenetet. Teszteld az általad használt forrásokat, vagy adj meg egy összehasonlítást, hogy figyelmen kívül hagyhasd a pótlólagos információt, ha az pontosan megegyezik a szabványos hibaüze-nettel.

A yaz_close() bezárja a kapcsolatot, amikor készen vagy. Normál weboldalak esetében lefuttatnál egy keresést, megjelenítenéd az eredményeket, és rögtön bezárnád az kapcsola-tot. Ha ismétlődő keresési folyamatot szeretnél kialakítani, az összes eredmény bekérésére használd a yaz_range()-t, mentsd el az eredményeket a session-rekordba, és dolgozz az el-mentett eredményekből ahelyett, hogy a hálózatról minden oldalhoz újra és újra lekérnéd ugyanazt az információt.

array_display()

Néhány információ sztring helyett tömbben jelenik meg, ezért a következő függvény átala-kítja a tömbböt sztringgé. Ha a tömb egyik eleme sztring, ez a függvény meghívja magát a tömb megformázása végett. Minden tömb egy táblázatba helyeződik át, a kulcsok a bal ol-dali, az értékek pedig a jobb oldali oszlopba. Nincsenek szóközök, sem formázás, ezért te tetszés szerint rakhatsz bele rácsokat és színezést:

```
function array display($array)

   if(is_array($array)

      $return = "<table cellspacing=\"0\" cellpadding=\"O\">\n";
      while (üst ($k, $v) = each ($array) ) í
```

```
■■ ■'  •      $return  .=  "<trxtd  align=\"right\"  valign=\"top\">"
.  $k
-. b.»;;.      .  "</td><td  valign=\"top\">"  .  array_display ($v)
               .  "</tdx/tr>\n";                    tt,rr(DOJl íniílJA sí'.

    return(Sreturn . "</table>\n");              ;

else                                                          '          _;
    '                                                                    X
    return(htmlentities ($array));
```

Dél-Ausztráliai Állami Könyvtár

Az első teszt az Ausztráliai Állami Könyvtár innopac-adatbázisát és egy usmarc típusú keresést használ. A szintaktika mezőt a usmarc-kal nagybetűvel, kisbetűvel és keverve is kipróbáltam, ugyanolyan eredménnyel. A @attr keresési kulcsszó attribútumok szerinti keresést jelent, az 1=4 pedig cím szerinti keresést. A keresés olyan könyvekre és dokumentumokra való hivatkozást ad vissza, amelyek címében szerepel a penfold szó:

```
$site = "143.216.21.3";
$database = "innopac";
$search = "@attr 1=4 penfold";
$syntax = "USmarc";
print(search($site,  $database,  $search,  $syntax));
```

Az alábbiakban az eredmények bejegyzéseiből válogattam, de a bejegyzéseknek számomra egy az usmarc-szabvány szerinti részleteket felsoroló dokumentum nélkül nincs jelentősége:

```
Hits:     12                                                  ""  "   ■■■'•
Hit                                                            '     :  "
  1     0  0                                          (3,001)      ;       .       ;    .
        1  0(3,001)(3,0)                                          .              ,
           1 000022215008                                         (              .  Jtvfi*
     23   0(3,245)(3,04)(3,a)                          .I  i r  *.        .-
           1 The  Penfold  Cottage  story.
     41   0(3,600)(3, 10)(3, a)                                                    ,\jl '
           1 Penfold,  Christopher  Rawson,
  3  23    0(3,245)(3,14)(3, c)              ; "-;r.-.€C£i
           1 Story  by  Oswald  L.  Ziegler.          ;
     36   0(3,500)(3,   )(3,a)                '        v"/iov  w-n"  -  ÍÍDTBSS?
           1 Wines  industries.  Australia.  Penfolds  Wines  Australia
             Ltd,
             1844-1974   (ANB/PRECIS  SIN  0037818)
```

A szöveg első sora valószínűleg a cím. A *Story by-]A* kezdődő sor a szerzőt adja meg, más sorokban a szerző neve önállóan szerepelt. Az utolsó sorok úgy tűnik, a könyv kategóriáját tartalmazzák.

A kimenet szintaktikáját és a különböző azonosító mezők jelentését nagyon)ól kell ismerned. Ezen adatok valós alkalmazásakor az outputot valószínűleg olyan hasznos mezőkre szűkítenéd, mint a cím, a szerző, és az ISBN.

Bell Labs

Az Állami Könyvtár oldala nem fogadja el a sutrs-szintaktikát, ezért azt a Bell Labs-nél próbáltam ki, a következő paraméterekkel (a szintaktika alapértelmezésben sutrs):

```
$site = "z3950.bell-labs.com";
$database = "books";
$search = "new york";
print(search($site, $database, $search));
```

Itt következik az eredmény, amely sokkal inkább alkalmas emberi olvasásra, de a szkripteddel való feldolgozáshoz nem eléggé tagok. Hogyan szednéd ki a kiadás dátumát ebből az adattömegből?

```
Hits:
1375 Hit
  1 Conference on Optical Fiber Communication (7th :. Digest of
    technical papers of t 1984. 621.38275/O62f 1984 v.l 100112T
  2 Electronic Components Conference (23rd :. 1973 proceedings : 23rd
    Electronic Components Con 1973? 621.3815/E38cl 1973 100553X
  3 International Electron Devices Meeting . Technical digest of the
    1976 IEDM. 1976. 621.3815/I614e 1976 101177A
  4 Product Liability Prevention Conference (9th : 1978 : Philadelphia.)
    Proceedings 1978. 658.026/P49 1978 101185J
  5 IEEE Power Engineering Society. Winter Meeting (1977 : New York.)
    Conference papers 1977. 621.319/121 1977 103080S
  6 IEEE Vehicular Technology Group. Annual Conference (22d : 1971 :
    Detroit.) Technical digest 1971. 629.2/111 22d/1971 104463X
```

A tanulság az, hogy mindig érdemes kísérletezni az adatforrásokkal és a szintaktikákkal. Egy sok különálló mezőt kínáló formátum több keresési lehetőséget nyújt, úgy mint cím, kiadó, kiadási dátum, viszont formázott megjelenítés nélkül elég nehéz az eredményeket elolvasni.

XML

Ahogy itt is látszik, a Bell Labs-nél lefolytatott próba az xml-szintaktikával pontosan ugyanazt az eredményt adta, mint a sutrs-szal. A Bell Labs szoftverei nyugodtan figyelmen kívül hagyhatják a szintaktikai beállítást:

```
$site = "z3950.bell-labs.com";
$database = "books";
$search = "new york";
$syntax = "xml";
print(search($site, $database, $search, $syntax))
```

GRS1

A Bell Labs-nél a grs l-re állított szintaktikával lefuttatott próba eltérő eredményt hozott. A Bell Labs szerverét úgy kell beállítani, hogy csak azokat a szintaktikatípusokat hagyja figyelmen kívül, amelyeket nem ismer fel:

bar

```
Ssite = "z3950.bell-labs.com"; —>--»■■-, ■»-■■• ■-
$database = "books";
$search = "new york";
$syntax = "grsl";                    !"__ -**■*-__'  * ~~
print (search ($site, $database, $search, $syntax) ) ;
```

A grs 1-gyel kapott eredmények kicsit hasonlítanak a usmarc-eredményekre, a hullámjelka rakter és függőleges vonal (~ |) hozzáadásával a sztringek sztringeken belüli elhatárolására, íme egy rész az eredményekből: „ . ,

```
10 0 (3,388)
   1 ~|+ 20 ~|a Conference on Optical Fiber Communication ~|n
     (7th : 1984 : -|c New Orleans, La.)
```

USMARC

A Bell Labs a következő kódban a usmarc-ot mint szintaktikát fogadta el:

```
$site = "z3950.bell-labs.com";
$database = "books";
$search = "new york";
$syntax = "usmarc";
print(search($site, $database, Ssearch, Ssyntax)) ;
```

Itt következik egy válogatás az eredményekből, hogy összehasonlíthasd a grs l-es eredményekkel:

```
21 0 (3,111) (3,20) (3,a)
   1 Conference on Optical Fiber Communication
22 0 (3,111) (3,20) (3,n)
   1 (7th :
23 0 (3,111) (3,20) (3,d)                                  í  ;  :
   1 1984 :                    '                           ..
24 0 (3,111) (3,20) (3,c)
   1 New Orleans, La.)                                     <fínöj-lbSÉ9ü£ A  "
```

Ismeretlen

Hogy kipróbáljam, mit csinál a Bell Labs szervere egy teljesen ismeretlen szintaktikával, a zzzz-t adtam meg:

```
$site = "z3950.bell-labs.com";   • '£■■■  .
$database = "books";              ■     ■
Ssearch = "new york";                                  ()fÍ316S3
$syntax = "zzzz";
print (search($site, $database, $search, Ssyntax) ) ;   jHaiW? OX'j>ií37Ö;J A
```

íme az eredmény egy része, amelyből látszik, hogy a szerver alapértelmezésben az usmarc-ot használja. Jobb lenne, ha a YAZ-függvények és a protokoll egy része együtt működnének, hogy lekérdezhessük a szerverről az elfogadott szintaktikák listáját:

```
21 0 (3,111) (3,20) (3,a)      ! %      '      :9D:CÍK^S = zl
   1 Conference on Optical Fiber Communication
```

Keresés több szerveren

Ez a megoldás az előző folytatása, egy keresést végez több forráson. A kód a keresési igé-nyekhez illeszkedő extra formázást is tartalmaz. Mindent elolvashatsz az űrlapokról a 9. fe-jezetben, és az ott látottak segítségével létrehozhatsz egy oldalt, ahol az emberek kitölthet-nek olyan mezőket, mint például a szerző neve, te pedig a beírt értékeket áttöltheted ebbe ■*■ a megoldásba.

Adatforrás

A search()-függvényt kiterjesztettük, hogy tömbben is elfogadjon adatforrásokat. A következő példa az első megoldásban bemutatott két adatforrást tartalmazó tömböt mutatja:

```
$source["State Library of South Australia"] =
   arrayC'site" => "143.216.21.3", "database" => "innopac",
   "syntax" => "usmarc");
$source["Bell Labs"] = arrayC'site" => "z3950.bell-labs.com",
   "database" => "books");
```

A Ssource-tÖmb a forrás nevével van ellátva, így név alapján csoportosíthatod a tömböket, látogatóidnak pedig egy szép választási lehetőség listával kedveskedhetsz. Az eredményeket ezután az ő forrásválasztásuknak megfelelően prezentálhatod.

A tömb a forrás szintaktikáját is tartalmazza. Néhány forrás elfogad többszörös szintakti-katípusokat, ezért egy kód hozzáadásával a látogatónak lehetősége lesz a szintaktikát a forrás alapján kiválasztani. Ez a kód feltételezi, hogy a $source a látogató választását tartalmazza, és nem a teáltalad a weboldalon kínált többféle lehetőséget.

Keresési paraméterek

A $search-tömb különféle fajtájú többszörös keresési paramétereket is elfogad. Kapd elő az egyik Z39 50-dokumentumot, vagy a Z39 50-oldal bibliográfiai referenciáit, és adj hozzá annyi mezőt, amennyit csak akarsz. Mindössze annyit kell tenned, hogy hozzáadsz egy összehasonlító kódot a search()-függvényhez, hogy minden mezőtípust feldolgozzon:

```
$search[]  =  array("title"  =>  "penfold");
```

searchf)

A következő search()-függvény két tömböt fogad el, és a többszörös eredményt abban a sorrendben adja vissza, ahogy a források a $source-ban fel vannak sorolva:

```
function  search ($source,  $search)
   if(is_array($source))
      $s  =  $source;
```

```
elseif (is_string (Ssource) and strlen ($source) )

    $x = explodeC'/", $source) ;
    $Y = explode (":", $x[0]);
    $s[$y] = array("site" => $source);

else

    $s["Bell Labs"] = arrayC'site" => "z3950.taell-labs.com",
        "database" => "books");
```

A kód első része ellenőrzi, hogy a $source egy tömb-e, és tömbbé alakítja, ha esetleg nem lenne az. Ennek segítségével a tesztelés megkönnyítésére megadhatsz egy adatforrást egy előre formázott sztringként. A Ssearch tartalma végül a $s-be kerül. A kódnak ezt a részét módosíthatod, hogy például olyan szolgáltatásokra is képes legyen, mint a forrás oldalának vagy IP-címének kikeresése egy adatbázisból, így az információt ezen a függvényen kívül máshol nem kell ismerni.

A kód második része a $search-tömböt RNP-sztringgé alakítja. Mérnöki hátteremnek köszönhetően a kódot úgy alakítottam ki, hogy az minden körülmények között működjön, beleértve egy előre formázott *lekérdező sztnng* fogadását is. Miután kiépítetted a kereső beviteli oldalát, kidobhatod a felesleges kódot, és hozzáadhatod azokat a finomításokat, amelyek segítségre lehetnek a keresésben. Az elterjedt mezőneveket, mint a cím- (title) és az attribútumreferenciákat, mint az 1 = 1003 közötti átfordítást egy tömbbe viheted, így kis erőfeszítéssel megadhatsz plusz mezőtípusokat, s majd a tömb használatával alakítod ki a kérdéseket a kereső beviteli részére:

```
if(is_array($search))

    $t = "";
    if(count{Ssearch) > 1)

        $t .= "@and ";

    while (üst ($k, $v) = each (Ssearch) ) {
        if(isset($v["author"]))

            $t .= "Sattr 1=1003 " . $v;

        if(isset($v["title"]))

            St .= " 1=4 " . Sv; }
        }

elseif(is_string(Ssearch))

    St = Ssearch;

else
```

K Jt---,

A következő kód létrehozza az eredménysztringet, az azt követő pedig ehhez kapcsolódik:

```
$result = "<br>Search string: " . $t; '"*"■•'  ' »'-'  ■'■
```

A kód kétszer halad végig a Ss-tömbön, egyszer a keresések összegyűjtésére, másodszor az eredmények összeszedésére. Az alábbiakban az elsőt mutatom meg, valamint a tömbben való lépegetést a while()-ciklus segítségével:

```
reset ($s);
while(list($k, $v) = each($s))

    $s[$k]["source"] = $v["site"]; A
if(isset($v["port"3) and strlen($v["port"]))

    $s[$k]["source"] .= ":" . $v["port"]; if

(isset($v["database"]) and strlen($v["database"] ))

    $s[$k] ["source"] .= "/" . $v["database"] ;

if(!isset($s[$k] ["syntax"]) or ! strlen($s[$k] ["syntax"]))

„;      $s[$k] ["syntax"] = "sutrs";

if($s[$k]["connection"] = yaz_connect($s[$k]["source"]))

    yaz_syntax($s[$k]["connection"], $s[$k]["syntax"]);,
    if(yaz_search($s[$k]["connection"], "rpn", $t))    ;


    else :Sí rarvssiéíe^c :.\i . is ■ MÁ ad.
        {$result .= "Search failed for " . $s[$k] [ "source"]
            . " and search " . $t;


else
    {
    $result .= "Connection failed for " . $s [$k] [ "source"];
```

A cikluson belül néhány a $v-tömbre utaló referencia a $s-tömbből származik, néhány pedig közvetlenül a Ss-re utal. A $s-re utaló referenciák a $s-be mentik az értékeket, a $s-en való második *áthaladásra.* Sok ember számára könnyebben követhető a kód, ha a $v referenciáit kicseréled a $s[$k]-ra. Mások akkor értelmezik könnyebben a kódot, ha az két részre van osztva: az egyik a mezők megformázására, a másik a mezők kapcsolatbeli használatára.

■3 £ .. 3

A yaz_connect () elfogad egy forrást, és létrehoz egy kapcsolatot ahhoz a forráshoz. Többszörös kapcsolattal is rendelkezhetsz, egy szerver pedig több kapcsolatot is elfogad, így egyetlen szerveren keresztül több adatbázishoz is kapcsolódhatsz. A yaz_syntax() meghatározza az eredmény szintaktikáját a forrás számára, a **yaz_search()** pedig továbbítja a kereső sztringet a kapcsolatnak. A kód további része hibaüzeneteket küld, ha a kapcsolat nem működik, de nem állítja le a keresést.

A yaz_wait(), mieló'tt a feldolgozás elkezdődne, megvárja, hogy minden forrás befejezzen minden keresést. Segítségével több keresést is végezhetsz párhuzamosan, de így egyetlen adatforrás lelassíthatja az egész keresési folyamatot. Ha hosszú, egymást átfedő informáci ókat tartalmazó listád van a forrásokról, előnyös lehet, ha olyan szkriptet hozol létre, amely egyszerre egy forrásból keres, és a keresések idejét is felsorolja, így kiiktathatod a túl lassú forrásokat:

```
yaz_wait ();
```

A következő kódszakasz mindegyik adatforrást áthurkolja, és kigyűjti az eredményeket:

```
reset ($s) ;
    whiledist ($k, $v) = each($s)) {
        $errno = yaz_errno($v["connection"] ) ;
        if($errno == 0)
            {
            $hits = yaz_hits($v["connection"] );
            $result .= "<br>" . $hits . " hits for
                . $v["database"] ;
            if($hits > 0)

                $record = "array";
                if($v["syntax"] == "sutrs" or $v["syntax"] == "xml")
                    {
                    $record = "string";
                    } $result .= "<table cellspacing=\"0\" cellpadding=\"O\">"
                    . "<tr><td>Hit</td><td>" . $v["syntax"]
                    . "</tdx/tr>\n";
                for($h = 1; $h <= $hits; $h++)
                    {
                    $hit = yaz_record($v["connection"] , $h, $record);
                    if (is array ($hit))

                        $resuit .= "<tr><td aiign=\"right\""
                    . " valign=\"top\">" . $h . "</td><td>"
                    . array_display ($hit) . "</tdx/tr>\n" ;
                    >
                    elseif (strlen($hit) )

                        Sresult .= "<tr><td align=\"right\""
                            valign=\"top\">" . $h . "</tdXtd>" .
                            htmlentities ($hit) . "</tdx/tr>\n";
```

```
-dr '           $result .=
                "</table>\n";

else  f.i v< düiiidír.bií ddftj
      $result .= "Wait failed. Error: " . $errno
{       . yaz_error($v["connection"]); $add =
u     yaz_addinfo($v["connection"]);
      if(strlen($add))

          $result . = "<br> Snbsp;" . $add;

yaz_dosc ($v [ "connection" ] ) ;          >A iliáti-no"
```

A függvény utolsó része az eredménysztringet küldi vissza, megjelenítésre készen:

```
return ($result) ;                                    tirn
```

A yaz_errno() jelez, ha valamilyen hiba van a kapcsolatban, a **yaz_hits()** pedig az adott kapcsolat találatainak számát adja vissza. A találatokat úgy gyűjtsd össze, hogy az eredményeket végigolvasod a **yaz_record{)** segítségével és egy **for()** ciklussal, amely egytől a találatok számáig fut. Hiba esetén a **yaz_error()** küld hibaüzenetet, a **yaz_addinfo()** pedig plusz hibainformációt ad. A yaz_addinfo()-ról az előző megoldásban olvashatsz. A **yaz_close() lezár** minden kapcsolatot.

E kód legnagyobb része táblázatformázás, amely helyettesíthető a többi fejezetben ismerte tett táblázatformázó függvényekkel. Úgy alakítsd ki saját táblázatkészítő függvényeidet, hogy azok valóban illeszkedjenek az igényeidhez, mert ezeket fogod a leggyakrabban használni. ü

array_display() '%!^^:T.:.^

Néhány eredmény sztring helyett tömbben jelenik meg, ezek az előző fejezetben mutatott **array_display()**-függvény segítségével tömbből sztringgé alakíthatók.

A keresés tesztelése .-íí £ < i i \ ~ i

A search()-függvény teszteléséhez nem kell mást tenned, mint begépelni a kód következő sorát. Az adatbázisok és keresési paraméterek minden változása az input tömbökben történik:

```
print {search ($source,     $search) ) ;          ■ .«, Sv : >n-.»^ö  j árir;    .-.ni.
```

Az eredmények w ?

íme a Dél-Ausztráliai Állami Könyvtár eredményeinek néhány sora:

```
Search string: @attr 1=4 Array
1 hits for State Library of South Australia innopac
```

```
Hit    usmarc                                                    x iq
   10   0    (3,001)                          IJJOV
     1  0   (3,001)                           is.íns                  s'ri,
        (3,@) 1
        flu00032526
```

Ez pedig a Bell Labs-eredmények első pár sora:

```
2 hits for Bell Labs books
Hit sutrs                                                          °*
   1 cn 621,395/l613a 1990 ti Application specific array processors :
     proceedings of the international conference, September 5-7, 1990,
```

A többféle forrásból származó adatok megjelenítésével egyetlen alapvető probléma van. Ha mindegyik forrás más formában adja ki az adatokat, azzal a kihívással kerülsz szembe, hogy az összes eredményt egyetlen használható listában kell megjelenítened.

Keresés a google.com-on ,«A o

Ez a kód a *Google* keresőmotor segítségével saját webszerveredet böngészővé alakítja. A kód alapul szolgálhat egy méta keresőmotor kifejlesztéséhez, amely egyszerre több JJ keresőmotor használatára is képes, de alapját képezheti egy forradalmian új tó keresőfelületnek, amely leváltja a már meglevőket.

Hogyan javíthanánk a Google jelenlegi kezelőfelületén? Az alapfelület meglehetősen korlátozott, a fejlettebben pedig túl sok választási lehetőség van, és egyik sem tesz lehetővé lépésről lépésre történő keresési finomításokat. Ami nekünk kell, az egy olyan kereső, amely automatikusan beszűkíti a keresési kérést, ezt teszteli egy kereséssel, kiszélesíti a keresést, míg nincs elegendő eredmény, majd a szűktől a tág felé haladva megmutatja az eredményeket.

A kód, amit mutatok, a Google alapértelmezésével végez el egy keresést, majd némi szer kesztést hajt végre a megjelenítés alapbeállításain. Ha tökéletes keresést szeretnél, az ered ményeket úgy szerkesztenéd, hogy szám szerint jelenjenek meg, és hogy ha nincs elegendő számú eredmény, a program másképp is próbálja meg a keresést. A teszt kereső sztring le gyen a *„PHP és weboldal architektúra"*. A Google kidobja az és szót, a többit pedig nem kacsolódóként kezeli. Egy jó keresőprogram először a pontos kifejezést keresné, azután az azt követő, a logikai és-sel kapcsolódó szavakat, majd megismételné a keresést, jobbról mindig egy-egy szót elhagyva, míg el nem éri a kívánt minimumeredmény számot. Ha a kereső tökéletes, először a phptect.com-ot hozná fel, de ez csak az elfogultság mondatja velem. ifi

Az űrlap

A következő kód a 9-es fejezetből való kód továbbfejlesztése:

```
$x   =   "";
if(isset($keywords))
    {
    $x   =   "   value=\""   .   $keywords   .   "\"";
```

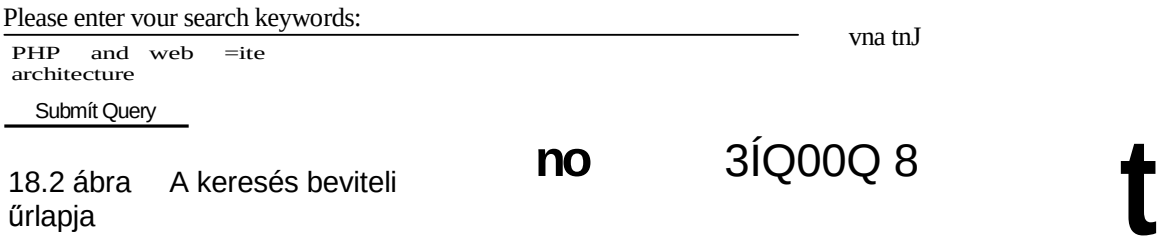
t;-v:-;$v-

t

```
print("<form action=\""  . $PHP_SELF . "\" method=\"post\">" .
    "Please enter your search keywords:"
    . "<br><input type=\"text\" name=\"keywords\" size=\"60\"" . $x . ">" .
    "<brxinput type=\"submit\" name=\"submit\ value=\"Submit\">"
    . "</form>" ) ;      . r,     ..
```

A keresés sztring a $keywords-mező'be megy, és az űrlap ciklust zár be magával, vagyis a kód a Skeywords meglétét használja annak eldöntésére, hogy először használod-e az oldalt. A $keywords-ben található érték lesz az alapértelmezett érték az egyetlen input-mezőben.

A 18.2 ábra mutatja a formátumot a keresett kifejezéssel, amit e függvény tesztelésére használtunk.



18.2 ábra A keresés beviteli űrlapja

Egy google.com-os egyszerű keresés lefuttatásához a következő kódon kívül nincs is másra szükség:

```
if(isset($keywords))
    {
    Ssearch = "http://www.google.com/search?q=" . urlencode($keywords);
    print ("<br>" . htmlentities(Ssearch) );
    $page = file(Ssearch);
    }
```

A URL hosszabb lenne, ha az oldal fejlettebb keresőjét használnád. A **file()**-függvény a URL-t az oldalra küldi, és a Spage-tömbben a keresés eredményeit tartalmazó oldalt küldi vissza:

íme a Google-hoz küldött URL:

```
http://www.google.com/search?q=web+site+architecture+and+PHP
```

Ugyanezt a keresési kifejezést a Google fejlettebb keresőjével az Exact Phrase-mezőben is kipróbáltam, és a következő URL lett az eredmény. Sok extra mező kihagyható, ha azok alapértelmezettek. Annak ellenére, hogy pontos kifejezési egyezést kértünk, a Google így is kiírta a figyelmeztetést: „Az AND operátor nem szükséges": :

```
http://www.google.com/seareh?as_q=&num=10&btnG=Google+Search&as_epq=PHP+
and+web+site+architecture&as_oq=&as_eq=&lr=&as_qdr=all&as_occt=any&as_dt
=i&as sitesearch=&safe=off
```

Hivatkozás:

Űrlap létrehozása 307

A nyers eredmények t^; -f J ^ < Ir';)

Mondjuk szét akarod szedni a keresett oldalt, és csak azokat a részeket kívánod használni, amelyekre tényleg szükséged van. Az alábbi kód a nyers eredményoldalt jeleníti meg, így magad jelölheted ki azokat a részeket, amiket szeretnél:

```
if (isset ($page) )                                   *  (v?   >*'£)
                                                       *  ,0                  ítw

    reset(Spage);
    while(list ($k, $v) = each ($page)
                                                                              .t;

       print("<br>k: " . $k . ", "      htmlentities($v)
                                      );
```

íme néhány kivonat az eredményoldalról, az oldal kezdetével, a 0 sorral, a stílus kezdetével a 3. sorban, annak végével a 13. sorban, az eredmények kezdetével a 18. sorban, és az alsó navigációs rész kezdetével az 53. sorban:

```
                                                                          A
k:   0,   <html>
k:   3,    <style><!--                                                     "
k:   4,   body    { f ont-f amily:    arial, sans-serif}                  t/'
k:   5,   div.nav    {margin-top:    lex}                                 ^|
k:  12,    II —>                                                          >
k:   13,
</style> k:   14,
</head>
k:   15,  <body   bgcolor=#ffffff   text=#000000   ünk=#0000cc
vünk=#551A8B k:   16,  <form  name=gs  method=GET  action=/searchXINPUT
TYPE=hidden
   valign=middle><input   type=text   name=q   size=31   maxlength=256
   value="web
   site   architecture   and   PHP">   <input   type=submit   name=btnG
   value="Google
   Search"xinput   type=submit   name=btnl   value="I'm   Feeüng
   Lucky"xbrx/form>                                                  ,  „. .,   .-.,
k:   17,   </td></tr></table>                                        ■ti -P    %_    ■A
k: 18, <table border=0 cellspacing=O cellpadding=2><tr><td ...^  ^ ..^ k: 53,
<div class=nav>
k: 54, <pxtable border=0 cellpadding=O width=10% cellspacing=O><tr
k: 71, <p><center><table width=100% cellpadding=2 cellspacing=0
Google</fontx/center></bodyx/html>                                       /,
```

Eredmények szerkesztése H -

A következő kód ismét a nyers eredményeket jeleníti meg, néhány sor elhagyásával: jg*-'

```
$nav   =   "<div   class=nav>";
if(isset($page) )                                                        ^

    reset ($page) ;                                                      „J
    while (Üst ($k,   $v)   =   each($page)   and   substr($v,   0,   7)   !=
    "<style>")


    print("<br>k:   "   .   Sk   .   ",   "   .   htmlentities($v));
```

```
wh
ile
(üs
t
($
k,
$v)
=
eac
h($
pa
ge)
an
d
su
bst
r($
v,
0,
8)
!=
"</
sty
le>
")

v ,:
prin
t
("<
br>
k:
"
.
$k
.
",
"
.
htm
lent
ities
($v)
);
```

```
print ("<br>k: " . $k . ", " . htmlentities ($v) ) ;
print("<br>") ;
while(üst <$k,    $v)   =   each($page)   and   substr($v,   0,   6)   !=   "<table")


print("<br>k: " . $k . ", " . htmlentities($v) ) ;
while (üst ($k, $v) = each($page)
    and substr($v, 0, strlen($nav)) != $nav)

                                                htmlentities($v));
    print("<br>k:    "    .
```

Az első sor a navigációs rész jelölőjét definiálja, ezért azt könnyen kicserélheted. Bármely más jelölőnél használhatod ezt a megoldást. A kód ellenőrzi, hogy a $page létezik-e, újraindítja azt, ha esetleg más kód is használta volna, majd egy sor ciklust hajt végre a Spage-en, hogy a kívánt részeket kiírja, a feleslegeseket pedig kihagyja.

Az első while()-ciklus a <style> -lal kezdődő bejegyzéseket az első sorban átugorja. Mivel a <style> -lal kezdődő sorokra az outputban van szükség, az aktuális bejegyzés egyszer kiíratásra kerül. A következő while()-ciklus addig végez kiírást, amíg a <style> záró tag-jét meg nem találja. Kiíratja a zárótag-et, majd szünet következik, és a while()-ciklus a <table>-tag elejét kezdi keresni. Elkezdődik a kiírás, míg a következő while()-ciklus megtalálja a tag-et a $nav-ben. Az egyik előnye annak, hogy változókban tároljuk a jelölő tag-eket az, hogy a strlenQ segítségével megkaphatjuk a tag hosszát, így az összehasonlítások a pontos hossz alapján történhetnek. A tag-eket akkor is használhatod a változókban, ha a hiányzó tag-ek jelölésére hibaüzeneteket veszel be.

A következő eredmény a $page-ből a kiválasztó ciklusok által visszaadott első sorokat mutatja. A stílust is tartalmazza, bár itt nem minden eredményt jelenítek meg:

```
k: 3,   <style><!--
k: 4,   body {font-family: arial,sans-serif}
k: 5,   div.nav {margin-top: lex}
k: 10,  A.Írünk {color: #6f6f6f}    Eq.'.'K  ^sfciocí oLún :><q>
k: 11,
k: 12,  A.u:ünk {color: green}
k: 13,
        </style>                          -■■       '                              ■

k: 18, <table border=0 cellspacing=O cellpadding=2xtr><td width=200>
```

Eredmények megjelenítése

A megszerkesztett eredményeket valószínűleg a weboldaladon szeretnéd látni, hogy könnyen össze tudd kapcsolni az eredményeket a szerkesztéssel. Az alábbi kód a te honlapodon mutatja a Google-ról származó HTML-kódot ugyanazoknak a kiválasztó ciklusoknak a használatával, de most a HTML-t az oldal részeként jelenítjük meg a htmlentitiesQ által előállított formátum helyett:

```
_. if (isset (Spage) )                          ^     _  •  -"-  •   _;   ■•     j>

    reset ($page);                                                          }
    print("<br>");       «
```

```
while (list ($k, $v) = each($page) and substr($v, 0, 7) != "<style>")


print($v);
while (üst ($k, $v) = each($page) and substr($v, 0, 8)   !=
    {                                                    "</style>")
    print($v);

print ($v);
while(list($k, $v) = each($page) and substr($v, 0, 6) != "<table")
    í
    }
print($v); while (list($k, $v) =
each($page)
    and substr($v, 0, strlen($nav)) != $nav)

    print($v);                                                    :w
```

A megjelenített eredmény első része a 18.3. ábrán látható.

--- **rr-----** The **"AND"** operator is unnecessary- we indude all search terms by default [details|

iarched the web for **web site architecture and PHP**

Categoiy. Computers > Proqramminq > Lanquaqes > PHP > **Tutorials**

Search Tools - Information. Guides and News

... a free service to the **web** development community and is not ...New version of the Inktomi 1■
site and enterpnse search engine ... page with SSI or **PHP** layout commands. ... ,bZ
iPescnption SearchToots reports on **WBh site** search tools, provides news about local site search engines and indexes,../;
Category Computers "-• Softwaic - Internet - *Semrs* > Search .
wwwsearchtook. com/- 16k.- Cached- Sirmlarpages '

18.3 ábra A Google keresési eredményei

További kódok hozzáadásával finomíthatod a keresés beviteli lehetőségeit, hogy a Google fejlettebb keresését is tudd használni, hogy egyszerre több keresőmotorral is kereshess, esetleg hogy az eredményekből kinyert információt saját stílusodban jelenítsd meg, vagy hogy az eredményeket más adatforrásokkal vegyítsd. Oszd meg másokkal fejlesztéseidet, vagy találj ki akár egy egészen új keresőt. A Google az egyik legjobb keresőmotor, de még mindig van rajta finomítani való, és a pálya előtted is nyitva áll.

Adatok indexelése

Mikor a keresésre felállítod saját adatbázisodat, egy keresőindexet is fel kell építened. Minél jobb ez az index, annál több látogatódnak sikerül majd az közvetlen keresés, és annál kevesebbet kell közülük a lassú, az SQL üke operátorát használó keresőkhöz visszaküldened. Az alábbi megoldás egy egyszerű indexépítést mutat végig, néhány általános példa használatával. Ezt a példát az adataidban található egyedi név és a leírási hagyományok alapján ki is bővítheted.

<' A következő lista néhány sztringet tartalmaz, amelyek könyv- és filmcímek megjelenítésének szokásos módját mutatják. Ezek tesztelik az indexépítő kódot:

```
$data[] = "PHP and web site architecture";
$data[] = "Test with Punctuation, The";
```

A következő lista olyan szavakat és kifejezéseket tartalmaz, amelyeket kizárunk, ha - és csak akkor, ha - a potenciális indexsztring pontos találat:)

```
$exclude["and"] = true;
$exclude["the"] = true;
```

Az alábbi kód a Sdata bejegyzésein halad végig, indexbejegyzéseket hozva létre a $index-ben, amelyek a Sdata-beli bejegyzésekre mutatnak vissza. Egy weboldal indexelő funkciójánál a Sindex-beli bejegyzések az input adatokat tartalmazó oldalakra mutatnának vissza.

```
while (list ($k, $v) = each($data))
    $v = trim($v);
    $index[$v] = $k;
    if (strtolower($v) != $v)
        {
        $index[strtolower($v)] = $k;
        }
    $x = str_replace("I", " ", $v) ; $x =
    str_replace(" and ", "|", $x); $x =
    str_replace(",", "|", $x); $y =
    explode("|", $x); while(list($yk, $yv)
    = each($y))
        {
        $yv = trim($yv);
        if(Istrlen($yv))
            unset($y[$yk]) ;
    if (count ($y) > 1)
        {
        reset($y);
        while(list($yk, $yv) = each($y)) 1
            $yv = trim($yv);
            if(! isset(Sexclude[strtolower($yv)])) {
                $index[$yv] = $k;
                if(strtolower($yv) != $yv)
                    $index[strtolower($yv) ] = $k;
```

Az első lépés a pontos, a kezdő- és záró szóközöktől megszabadított kifejezés áthelyezése a $data-ból a Sindex-be. A második lépésben a kifejezést kisbetűsre alakítjuk, és a kisbetűs variációt visszük át az indexbe. így lehetséges lesz a rövidítések pontos kis- és nagybetű egyezéssel való keresése, ma|d a kis- és nagybetűtől független párosítás is. sít

A kód az str_replace()-függvény segítségével az összes előforduló elválasztójelet egyetlen * közös | karakterré alakítja. Ezután az explode() a $array-tömbbe választja szét a sztring ■ részeit. További kódok hozzáadásával a sztringet szétbonthatod az idézőjelek, pontosvesszők vagy bármi más alap|án, amit a szövegedben találsz.

Ha a $y-ban csak egyetlen bejegyzés van, az alapvetően ugyanaz, mint az indexben már benne levő eredeti bejegyzés. Ha a $y egynél több bejegyzést tartalmaz, mindegyik hozzáadódik az indexhez. Én csináltam egy $y-on végigfutó ciklust, hogy az még a számolás előtt eltávolítsa a nulla és az üres bejegyzéseket, arra az esetre, ha az adatokban valahol kettős elválasztójelek szerepelnének.

A $y-on végigfutó while()-ciklus kihagy minden olyan bejegyzést, ami a $exclude-ban van, a maradékot pedig a $index-be teszi. Mielőtt egy bejegyzést a $exclude-ban tesztelünk, lerövidítjük és kisbetűsre alakítjuk. A $index-be kerülő bejegyzések is megszabadulnak a kezdő és záró szóközöktől, és vegyes vagy kisbetűs változatban kerülnek be.

Itt az ideje, hogy teszteljük az indexet. A következő kód a $index-et táblázat formájában jeleníti meg, a bal oldali oszlopban az index bejegyzéseivel, a jobb oldaliban pedig az index bejegyzései által azonosított sztringekkel:

```
print("<table>");
while (üst ($k,          =    each
$v)                      ($index) )

   print ("<trxtd>" . 5k . "</tdxtd>" . $data[$v] . "

print("</table>");                                          . '
```

íme a felépített index alapján előállt output:

```
PHP and web site architecture PHP and web site architecture php
                        and web site architecture PHP and web
PHP                     site architecture
php                      PHP    and   web    site    architecture PHP
web     site                        and   web    site    architecture
architecture                   PHP    and   web    site
Test    with            architecture
Punctuation,            The                                  Test
test    with            with    Punctuation,    The
                         the       Test    with    Punctuation,    The
                                    Test    with    Punctuation,    The
                                    Test    with    Punctuation,    The
```

Az igazi kihívás az, hogy megtaláld az azonosítókat az adataidban, és úgy rendezd el őket, hogy látogatóid igényeit ki tudd elégítem. Ha utcanevek megtalálásában akarsz segíteni, és a Rózsakertből elveszed a Kert szót, a Rózsát az emberek számos sugárút, utca stb. között megtalálhatják. Ha véletlenül a Rózsa utcát gépelnék be, mutass meg nekik minden Rózsa utcát, és ha nincs ilyen vagy csak kevés van, csak akkor hozd elő a Rózsakerteket.

637

Ha az adatbázisodban éttermek, virágüzletek, növénytermesztők is szerepelnek, a Kert elhagyása több ezer Rózsát tartalmazó virágüzlet nevet adna ki. Talán te is egyetértesz azzal, hogy az SQL hasznos vonása a limitelőírás, ami az adatbázisból kiadott eredmények számát határolja be. Adj meg egy 10—50-ig terjedő eredménytartományt, és az 50-es határt add meg az SQL-ben. Ha a látogató keresése 10-nél kevesebb találatot ad, próbálj meg egy szélesebb keresést, mint például a grove elhagyása a rose grove-ból. Ha az eredmények túllépik a határt, a keresőnek adj konkrétabb, pontosabb keresést, és mutasd meg a látogatónak az általad kínált plusz választási lehetőségeket, mint például az állam vagy város kiválasztása. Hogy az ilyen emlékeztetés valóban hasznos legyen, ellenőrizd, hogy az eredményeid tartalmazzák a helyes mezőt a látogatónak kiadott bejegyzéseknél. Ha a látogató látja, hogy államonként több bejegyzés is szerepel, megpróbálja kiválasztani a megfelelő várost. Ilyenfajta vizuális támogatás hiányában a látogatók maguktól talán csak az államot választják ki, és még mindig több, mint 50 találatuk lesz. A Google megmondja, mennyi bejegyzést talált. Ha az eredményeken végighaladsz l-l 0-ig, és látod, hogy összesen 23 találat van, persze végigpörgeted az oldalakat. Mégis egy 29 858 as számú találat talán nagyobb keresési pontosságot hív elő az emberekből.

Gyakran gépelnek be látogatóid kerítéslécet mikor léckerítést keresnek? Hány boltos emlékezne a citrom-, lime-, narancs- és gyömbér-lekvár helyes sorrendjére? Egy kód egyszerű hozzáadásával az azonosítókat szavakra bonthatod, majd új sorrendben ismét összeállíthatod őket. Ha egy szó-sztring szavait az explode() segítségével egy tömbbe szétbontod, ciklusokkal végigmehetsz a tömbön, megkapva a szavak mmden lehetséges kombinációját.

Elterjedt szavak szinonimáit is bevetheted. Add hozzá a dzsemet, mikor a lekvárt látod. Győződj meg róla, hogy nem cseréled le a szavakat, mint néhány buta keresőmotor; csupán extra hivatkozásokat kínálsz a meglévőkön felül.

19. fejezet

Session-ök

í

..a-i

kin .í.irnii; i

Áttekintés

A session-ök a World Wide Webet a véletlenszerű oldalbeolvasásból koordinált, a felhasználó és a honlap közötti interaktív eszközzé változtatták. Tegyük fel, hogy egy szupermarket természetgyógyászati terméket kínáló polca előtt állsz. A C-vitaminon van egy rövid termékismertető, amely tájékoztat arról, hogy a C-vitamin a korpától a rákig mindent meggyógyít. Az ugyanitt megtalálható cinktabletta dobozán azt olvasod, hogy a cink a ráktól a korpáig mindenre jó. Az információ elszigetelt és koordinálatlan.

Most menj el egy természetgyógyász rendelésére, aki egy rendszerezett tájékoztatót tart, és elmondja, hogy milyen eló'nyei vannak, ha C-vitamint szedsz, de arra is figyelmeztet, hogy a C-vitamin elrágása esetén az aszkorbinsav kilukaszthatja a fogad. A természetgyógyász azt is elmondja, hogy a cink jó a bőrödnek és segít a korpa ellen is, de az étrended elég cinket tartalmaz, és hogy a cinket a kávé kivonja a szervezetedből. Minden egyes alkalommal, amikor felkeresed, a természetgyógyász feljegyzi, amit mondasz, és a következő' látogatásod előtt elolvassa, hogy mi történt az előző alkalmakkor. Hasonlóképpen működnek a -—weboldalakon a session-ök.

Az Internetet véletlenszerű, egymással kapcsolatban nem álló események sorozataként tervezték meg, így egy esemény meghiúsulása nem érinti a többit. Az FTP fájltovábbítása egy egyedi *esemény*. Esemény történik akkor is, amikor a böngésző megkér egy oldalt, és a weblap szervere elküldi azt. Ha a weblap képeket tartalmaz, és a böngésző a képeket kéri meg, minden kép megkérése és elküldése egy külön esemény. Ugyanúgy, ahogy a szupermarket egyik termékcímkéjének sincs semmi köze a másik termékéhez. Az Internet szabályai biztosítják, hogy csak minimális szabvány és formátum vonatkozik az adatra, így az adat nem vész el. Ugyanígy a hivatalos szervek is csak azt írják elő, hogy a termékcímkén fel kell tüntetni a töltési tömeget, de azt nem, hogy a termék gyógyítja a rákot.

A World Wide Web örökölte az Internet véletlenszerűségét, és megpróbált egy kis rendet vinni bele az URL-ekhez adott lekérdezési sztringekkel. Ha a *korpa* szóra keresel egy honlapon, a szó belemegy az URL-be, vagy egy GET- vagy POST-fejrészbe, és az továbbítódik a honlapra. Általában a keresési kifejezés a weboldalon belül visszamegy a böngésződbe az oldal URL-hnkjeihez adott lekérdezési sztrmgként. Amikor újra keresel az oldalon, a *korpa* szóval előre betöltődik a keresési bemenő doboz, így innen kezdheted és olyan szavakkal finomíthatod a keresést, mint a *viszketés*, *súlyos* vagy *hóvihar*.

Lehet, hogy a weblapnak 300 szerver válaszolt a kérdéseidre, és a lekérdezési sztringet leszámítva egyik sem tudja, hogy mire kerestél az előző oldalon. Mi van, ha 20 keresési kifejezést kipróbáltál, és unottan otthagyod az oldal, mert nem kaptál választ vagy megfelelő gyógymódot? Ilyenkor segítenek a session-ök.

A session-ök egy azonosítót adnak a böngésződnek és a szervernek, hogy egy oldal lekérését az ugyanabban a session-ben történt többi lekéréshez kapcsolják. Ha az azonosítót intelligens módon használod, a rák meggyógyításán kívül bármit elérhetsz vele.

A honlap tulajdonosának haszna

A session a honlap tulajdonosának és a látogatónak is rengeteg hasznot nyújt. A tulajdonos használhatja arra a session-rekordokat, hogy megnézze a 20 kérdést, amellyel a korpára rákerestél, és lássa, hogy mégsem vettél C-vitamint (vagy bármit, amit árul). Ennek a visszacsatolásnak a birtokában rájön, hogy javítania kell a keresőket, hogy megtaláld a megfelelő oldalakat. Ha az oldal kerékpáralkatrészeket árul, dönthet úgy, hogy a C-vitaminnal kell az oldal választékát bővíteni. A tulajdonos utánanézhet azoknak az oldalaknak, amelyeket azelőtt látogattál meg, hogy a korpára vonatkozó információkra rákerestél, és elkezdheti * törni a fejét, hogy vajon egy kerékpáralkatrészeket forgalmazó oldalon mi késztetett arra, hogy a korpa gyógymódjaira rákeress.

Azok a csicsás webelemző segédprogramok, amelyek több ezer dollárba kerülnek, csak session nélküli díszek. Csak azokat az oldalakat mutatják, amelyeket az emberek felkerestek, de azt nem, hogy melyik oldalakról jött a látogató a honlapodra, és melyik miatt ment tovább. Használj session-t, és az adat információvá alakul. Megtudhatod, melyik a session első oldala, utolsó oldala, és átlagosan mennyi ideig olvasnak egy oldalt.

A látogató előnyei

A látogató egy jobb minőségű oldalt kap, ha a tulajdonos session-ökkel elemzi és javítja azt. A látogató ekkor be tud jelentkezni, automatikusan a testre szabott profilját használja, bevásárlókosarat használhat, megtekintheti a kumulált keresési kifejezéseket, és számtalan egyéb hasznos lehetőséghez jut.

A session kapcsolódhat egy fájl kis rekordjához vagy akár a felhasználó preferenciáira vonatkozó információk egész adatbázisához.

Sessionazonosító

A session-ök közötti kapcsolatot az *azonosító* (vagy kulcs) teremti meg, amely minden egyes oldallal elküldésre kerül a böngészőbe, amelyet az a következő oldallekéréssel visszaküld a szervernek. Az azonosítót létrehozhatod saját magad, vagy generálhatja azt a PHE Az azonosítót lehet a session-információk adatbázisának kulcsaként vagy a session fájlné ven belüli egyedi sztringként használni. Az azonosító cookie-ban vagy link URL-ekben ke rül a böngészőbe és vissza. - .i. Vi,■'.-,.;

Cookie-k

Rengetegen összekeverik a cookie-kat a session-ökkel. A cookie a session-azonosító oldalról oldalra továbbításának egyik eszköze, de semmi több. A session-ökhöz nem feltétlen kell cookie, és session-azonosítón kívül semmit nem is tárolnak benne.

Van olyan bevásárlókosár-szoftver, amely a kumulált bevásárlólistát cookie-ban tárolja. Telepítsd a szoftvert az oldaladra, és kérd meg az embereket, hogy teszteljék a cookiealapú

bevásárlókosarat. Meg fogsz őrülni. Lesznek, akik megváltoztatják az árakat a cookie-ban, így a 9990 Ft-os könyv 990 Ft-ba fog nekik kerülni. Lesz, aki a kosár tartalmának a felét elveszíti, mert a cookie túllépi a böngészőben neki beállított 4KB-os cookie-méretet. Ahogy vásárolgatnak, és információért megnyitják a terméket gyártó cég honlapját, a cookie számlálója továbbmegy, és végül a böngésző eldobja az első cookie-t - amelyikben a bevásárló-kosár van.

Ha a cookie-ban információ van, akkor az az információ annyira van biztonságban, mint a hópihe a mikrohullámú sütőben. Ha cookie-ból származó információt olvasol, úgy bízz benne, ahogy a "nem lesz adóemelés" vagy a "pénzét már elutaltuk" szövegekben bízni szoktál. A jó cookie-alapú session-ök nem csak távol tartják az információt a cookie-ból, hanem nagy erőfeszítéseket tesznek azért, hogy megbizonyosodjanak arról, hogy a session-azonosító valódi-e. Mit tehetsz annak érdekében, hogy megbízz a cookie-ban? A session-azonosítót egyszerű szövegként tedd a cookie-ba, hogy a visszatérő cookie-t a szerverkezelő szoftver automatikusan az elsődleges szerveredre vagy egy intelligens routerbe irányítsa. Minden egyes oldallekéréshez adj egyedi, rejtjelezett, titkos kulcsot, amelyet megfejtve ellenőrizheted, hogy a bejövő oldallekérés a webszerver ugyanerre a session-re adott válasza-e.

Bármi, amit cookie-ban küldesz ki, legyen rögzítve a szervereden, hogy ellenőrizni tudd, amikor a cookie visszajön. Ennek eredményeképpen a kapcsolatod biztonságos és megbízható lesz.

Más dolgokat is el kell kerülni, amikor a session-ökhöz cookie-t használsz. A session cookie-n belül rejtjelezheted a látogató IP címét, vagy tárolhatod egy session-rekordban, így minden olyan kérést visszautasíthatsz, ami más IP címről jön. Azonban a WebTV-felhasználók és bizonyos proxy-k minden lekéréshez különböző IP címet adnak meg, így ők nem tudnak az olyan oldalon vásárolni, ahol az IP címet használod a session igazolására.

Amikor a látogató a Visszagombra kattint a böngészőben, a szerverre visszajuttatott információ attól függ, hogy a böngésző készítői hogyan értelmezték a Visszagomb írásának szabályait, de akár jól, akár rosszul, majdnem biztos, hogy az információ nem megfelelő a vásárlási alkalmazások számára. Egy ilyen alkalmazásban azt akarod, hogy a vásárló úgy tudjon az előző oldalra visszatépni, hogy a bevásárlókosár tartalma ne változzon. Ez azt jelenti, hogy a session-rekordban változatlanul kell tartanod a bevásárlókosarat, és olyan régebbi oldallekéréseket kell elfogadnod, amelyek megzavarhatják a cookie tartalmára kialakított ellenőrzési szabályaidat.

H T T P S ■-.-■■*■•■

A visszalépések engedélyezése esetén a cookie-k ellenőrzésének egyik lehetséges megközelítése a HTTPS használata, amely a HTTP biztonságosabb változata. A bejelentkezés alatt HTTP-ről HTTPS-re váltasz, amely a böngésző és a honlap közötti adatfolyamot rejtjelezi.

A biztonságtechnikai szakértők sokkal többet el tudnak neked mondani a HTTPS-ről, mint én a korlátozott tapasztalataimmal. Egyszer segítettem egy barátomnak a weboldalában, és kiderült, hogy sokkal többet tudok a HTTPS-ről és a biztonságról, mint a barátom oldalát kezelő Internet-szolgáltató vezető technikai szakértője. Ciki. Ellenőrizd az oldalad biztonságát az Internet-szolgáltatóddal, majd kérjél még egy véleményt.

Számtalan nagy online-rendszer biztonsági adminisztrátoraként dolgoztam, és tudom, hogy a HTTPS mögötti összes koncepció létezett már a web elő'tt is, ezekeket már alaposan áttanulmányozták és kellő mértékben feltörték. Egy szakértő a honlapod biztonságossá tételének összes lépését megmutatja, a főbb problémák pedig - a jelenlegi tapasztalatok alapján -a proxy-szerverek és a tűzfalak lesznek.

Cookie-k vagy URL-ek

A cookie-k távol tartják a zavaros session-azonosító sztringeket az URL-ektől, és lehetővé teszi, hogy az emberek anélkül vegyenek fel oldalakat a könyvjelző közé, hogy a session-azonosítót a könyvjelzőkbe zárnák. Amíg tesztelsz, az URL-ek a zavaros session-azonosító sztringet láthatóvá teszik az URL-ben, és az emberek akkor is tudják a honlapodat használni, ha a cookie-kat kikapcsolják.

A cookie-kat el lehet küldeni a látogatónak, de ő visszautasíthatja ezeket, és ezt a szkripted fogja észlelni. Amikor egy látogató az oldaladra érkezik, a $HTTP_REFERER mező egy másik oldalra mutató URL-t tartalmaz, a $REMOTE_HOST pedig egy másik szervernevet. Ezt a különbséget felhasználhatod az első látogatásuk azonosítására és mind a cookie, mind a link URL-ek által küldött információ értelmezésére. Amikor a látogató a honlapodon a következő oldalra megy, keresd a bejövő cookie-t annak megállapítására, hogy a böngészője be van-e állítva a cookie-k fogadására. Állíts be a session-rekordjában egy mezőt cookie-ra vagy URL-re, majd a következő oldalakat a session-rekordban meghatározott módon tárolt session-azonosítóval küldd el.

Adatok vagy adatbázisok

A webszerveren a session-információkat fájlok vagy adatbázisok használatával tárolják. Az adatbázisokban session-önként egy rekord van; fájlok esetén pedig egy fájl. Fájlokkal könnyebb a session-ök beállítása, így az első PHP-beli kísérletezéshez jobban illenek. Én inkább az adatbázisokat használom a fájlok helyett, mivel már tudom, hogyan kell őket használni. A választásod függhet attól, hogy melyik fejezetet olvasod először, ezt vagy az ötödiket.

MySQL-t használok a session-ökhöz, mert könnyen telepíthető, minden olyan operációs rendszeren fut, amelyeket a weboldalakhoz használok, könnyű felügyelni, végezhetek session-elemzéseket, és mentes a tranzakcióalapú adatbázisok minden fölösleges jellemzőjétől. Választhatsz más adatbázist, vagy megmaradhatsz a fájloknál is, de nagyobb oldalaknál mérlegeld a következőket, mielőtt a megfelelő módszert kiválasztod.

Az olyan fájlrendszerekben, mint a Linux ext2, a kis mennyiségű adatok sok többlethelyet foglalnak el, míg a területhatékony Reiser- vagy NTFS-fájlrendszereknek csak bizonyos méretű fájlok esetében optimálisak. Az adatbázisok a session-rekord méretétől függetlenül csak kis többlethelyet foglalnak el.

A fájloknál alacsony ugyan a feldolgozási többletráfordítás, de ha meg vannak osztva szerverek között, problémáik lehetnek. A feldolgozási többletráfordítás magasabb az adatbázisoknál, de jól megoldott a többszörös szerverek hozzáférése.

«■ Ili-?:

nnv ..-■■* p-i"í

t

A MySQL lehetőséget ad a tranzakciós támogatással felépített táblázatok vagy az egyszerűbb és gyorsabb, tranzakciós támogatás nélküli táblázatok választására, amelyek ideálisak a session-ökhöz. Minden adatbázisodat meghagyhatod PostgreSQL-ben (az 5. fejezetben mutatom be) vagy DB2-ben (a 6. fejezetben mutatom be), és állíts fel egy MySQL-adatbázist kizárólag a session-kezelésre. A MySQL-táblázatok könnyen kezelhetők a phpMyAdminnal, a MySQL PHP alapú adminisztrációs rendszerével. Az 5. fejezetben megnézheted, hogyan írj saját MySQL-szkripteket. A PostgreSQL-ben megvan a phpMyAdmin megfelelője, de abban még nincs meg a táblázatok tranzakciós támogatását kikapcsoló opció. A választásod függ az operációs rendszeredtől és a többi alkalmazásod követelményeitől.

Ha az aktuális session-öket listázni vagy elemezni akarod, a session-rekordokat adatbázisba kell tenned. A rekordok fájlból való elemzéséhez be kell olvasni egy könyvtárat, illetve annak minden fájlját, és azokat is, amelyek aközben tűnnek el, hogy a könyvtárat és az egyedi fájlokat beolvasod. Adatbázis használatakor csak egy sor SQL és egy lekérdezés kell.

PHP-szolgáltatások

/ÍÜ Jílíl

A PHP-ban vannak beépített session-kezelő szolgáltatások, amelyek a legtöbb esetben működnek, illetve olyan esetekben, amikor ennél jobban akarod kontrollálni a session-öket, ezek kikapcsolhatok vagy mellőzhetők. Kezdd azzal, hogy megpróbálod megérteni a php.ini ' beállításait, a session-rekordokat pedig a session-függvényekkel való gyakorlás idejére hagyd fájlokban. Ha a session működik, kísérletezz a php.ini változtatásaival, majd rakd át a session-rekordokat adatbázisba. Jegyzeteld le a php.ini változtatásait és azok eredményeit.

php.ini

A következő kódpélda az alapbeállításokkal telepített PHP 4.0.5 php.ini Session része:

```
[Session]
session.save_handler = files
session.save_path = /tmp
session.use_cookies = 1
session.name = PHPSESSID
session.auto_start = 0
session.cookie_lifetime = 0
session.cookie_path = / C
session.cookie_domain =
session. serialize_handler = php
session.gc_probability = 1
session.gc_maxlifetime = 1440
session.referer_check = •'
session.entropy_length = 0
session.entropy_file =
;session.entropy_length = 16 "'■ ;
session. entropy_f ile = /dev/urandom
session.cache_limiter = nocache
session.cache expire = 180
```

```
session.use_trans_sid = 1 '          '                                    ■
„rl rewriter.tags =
   "a=href,area=href,frame=src,input=src,form=fakeentry"
```

Ebben a részben röviden elmagyarázom azokat a beállításokat, amelyeket lehet, hogy meg kell változtatnod. A php.ini legújabb verziójában sokkal több beállítás van, mint amennyi a session-ök megjelenésekor volt, így feltételezhetjük, hogy a hozzáadott beállításokat a felmerülő problémák megoldására tették be, de ezek nem olyan problémák, amelyek a. session-ök első használatakor jelentkezhetnek.

A **session.save_handler** alapértelmezésben fájlban tárolja a session-öket, és használatával **user-t** határozhatsz meg. Ha ezt megteszed, használd a **session_set_save_handler()-t** olyan függvények definiálására, melyek a session-rekordok írását és olvasását kezelik. A következő kód azt mutatja, hogy mit kell a php.ini-ben megváltoztatnod, ha a session-öket adatbázisba, például MySQL-be helyezed:

```
sessíon.save_handler    =    user
```

A session.save_path azt határozza meg, hogy a PHP hova írja a session-fájlokat, amikor fájlokat használsz. Unix és Linux **alatt /tmp-t,** Windows NT/Windows 2000 alatt **c:/temp**-et vagy c:\temp-et, Windows 98 vagy későbbi verzió alatt pedig **c:\temp-et** használj. Ha a session-ök működnek, hozz létre a session-rekordoknak egy meghatározott könyvtárat, korlátozd az ehhez való hozzáférést, hogy a rendszer többi felhasználója ne érje el a session-rekordokat, mert akkor lehetősége lesz mókából vagy nyerészkedésből feltörni a session-t.

A session.use_cookies-zal kikapcsolhatod a cookie-kat, de a cookie-k első tesztelésénél ne próbáld még ki. A **session.name** a cookie nevét adja. Állítsd a **session.auto_start-ot l-re,** hogy minden oldalra cookie-t indítson, és amikor a cookie-k már működnek, kapcsold ki az automatikus üzemmódot a manuális szabályozás kikísérletezésére, hogy például csak akkor kezdj egy cookie-t, ha valaki bejelentkezik.

Ne nyúlj session.cookie_lifetime-hoz; alapértelmezésben addig futtája a session-öket, amíg a böngésző **aktív.** A **session.serialize_handler-ben** hagyd meg a **php-t,** illetve minden más beállítást hagyj meg az alapértelmezett értéken, míg a cookie-k nem működnek.

A **session.gc_probability** beállítja, hogy az idő hány százalékában fusson a PHP session-hulladékgyűjtő. Akárhányszor egy oldallekérés PHP session-kezelő rutint indít el, a PHP ellenőrzi, hogy szükség van-e a hulladékgyűjtésre. Hagyd a **session.gc_probabüity-t** l-re állítva, hogy a hulladékgyűjtő csak 100 oldalanként egyszer fusson. Vannak ugyan elméleti indokai az érték megváltoztatásának, de pont annyi érvet lehet az érték megnövelése mellett felhozni, mint a csökkentés mellett. A változtatás eredménye nehezen mérhető és függ az adatbázisod típusától, a szerverkonfigurációtól és a forgalom mértékétől. Egy olyan honlapon, amely a nap egy részében rendkívül leterhelt, a nap többi részében viszont elég csendes, lehet, hogy van értelme a tisztító kódnak a hulladékkezelőből való eltávolításával kivenni a hulladékgyűjtőt a PHP ellenőrzése alól, és a tisztító kódot egy olyan kötegelt feladat futtatóba tenni, mint a cron ütemező.

A **session.gc_maxlifetime** meghatározza, hogy a session-rekord az utolsó használat után hány másodpercig használható. Az alapértelmezés 1440 másodperc, így 24 perccel az után,

hogy valaki megnyomja az Entert, a PHP session-hulladékgyűjtő rutinja törli a session-rekordot. Lehet, hogy az oldalad valamelyik felhasználója panaszkodni fog, hogy ez túl rövid, és ezért gyakran kell visszajelentkeznie, így addig-addig növeled a beállítást, amíg valaki azt fogja reklamálni, hogy a session-jét eltérítették, miután ő hazament. Nincs olyan megoldás, amely mind a 225 000 felhasználót kielégíti, így a felelősséget dobd vissza az oldal tulajdonosának, ő legyen az, aki kihúzza a gyufát.

A session.referer_check-kel korlátozhatod a látogatókat bizonyos oldalak elérésében, de az ellenó'rzés nem megbízható, mert maga a forrásadat, azaz hogy ki látogatja az oldalad, nem megbízható. Hagyd ki ezt a pontatlan ellenőrzést a session-ökből. Ehelyett minden látogatónak legyen session-je, és a hozzáférést a hagyományos biztonságtechnikai megoldásokkal korlátozd. Ugyanakkor hagyd meg nekik a lehetőséget, hogy az oldalad olyan részére jussanak el, ahol informálódhatnak, kitölthetnek űrlapot, így nem veszítesz el potenciális vásárlókat.

A session.entropy_file-lal meghatározhatsz egy fájlt, amely további adatokat biztosít a session-azonosító generáló rutinnak. Több Unix-rendszeren használhatod a /dev/random-ot vagy a /dev/urandom-ot. Még nem találtam olyan oldalt, amely ezt a beállítást igényelte volna. Amikor az oldalak olyan nagyra nőnek, hogy több szerveren vannak, és a PHP véletlen azonosító generálóját a határig lökik, az oldalak gyakran intelligens routerekre váltanak, amelyek elvégzik a session-azonosító elosztását és vezérlését. A PHP azonosítóját felcserélheted egy adatbázis önnövelő mezőjéből generálttal, ami biztosítja, hogy az azonosító egyedi lesz az adatbázist használó szervereken. A session.entropy_length a külső fájlból beolvasott bájtok számát határozza meg, így 0-val letilthatod a külső' fájlokból való beolvasást.

A url_rewriter.tags határozza meg, hogy melyik HTML-tag kapja a session-azonosítót, amennyiben a cookie-k ki vannak kapcsolva. Az alapbeállítás tökéletesen megfelel a kezdeti teszteléshez, és lehet, hogy soha nem kell megváltoztatnod.

PHP session-függvények . • - -.

A következő függvényekkel manuálisan lehet szabályozni a session-öket, így a php.ini-beál-lítások felülírását is. Amikor a PHP alapértelmezett session-fájljait adatbázisra cseréled, session-rekord kezelő-függvényeket kell definiálnod, ahogy azt a „Session indítása MySQL használatával" című Gyors megoldásban láthatod,

session_cache_limiter()

A php.ini tartalmazza a session.cache_limiter-beállítást,a mellyel a HTTP-fejrészekkel vezérelhető a cache-elés. A session_cache_limiter()-rel megjeleníthet az aktuális beállítást, és megváltoztathatod az aktuális szkriptednek megfelelően. Természetesen ennek a fejrészek elküldése előtt le kell futni, a fejrészek pedig akkor lesznek elküldve, amikor a session_start()-ot vagy session_register()-t használod, illetve amikor a böngészőnek adatot küldesz. A következő kód megjeleníti az aktuális értéket, majd private-re változtatja azt (az opciók a priváté, a nocache és a public):

```
                                                    ,'iJi/ noij E ínirn .innsi ed
print("<br>"    .    session_cache_limíter());
session_cache_limiter ("priváté") ;                 ta'jm ■9rnÍJ3liíxErn_3§.not«íffí
.rií;ju                                  _>f»i aavjínismq/Js xÁ .c.-i.AÍEns.^.d yi;mqí)O2í;rrr re
```

session_decode()

Ha a session-ök tesztelésekor a session-rekord-információkat manuálisan akarod kikódolni, használd a session_decode()-ot. A következő kód kikódolja a sztringet, és jelzi az esetleges hibát:

```
if(!session_decode($data))

   print("<br>Decode failed for " . htmlentities($data) ) ;
```

Hova kerül az adat? A globális változóösszesítőbe. Ha használod is a session_decode()-ot egy függvényben, az adat a globális változóösszesítőbe kerül, és éppen ezért a session kikó-dolást a tesztelésen kívül semmi másra ne használd.

session_destroy()

A session_destroy() eltávolítja az összes, a session-höz kapcsolódó adatot.
A session_destroy() előtt használnod kell a session_start()-ot, és a session cookie a böngészőben marad. A következő kód egy session-t indít, hogy lerombolhassa azt, majd ennek megfelelően le is rombolja:

```
session_start();
if(!session_destroy())

   print ("<br>session destroy failed.");
```

Mi történik, ha egy szkriptben kitörölsz egy session-t és azt követően újat hozol létre? A következő hibaüzenetek jelennek meg, ha a session_start()-ot vagy a session_register()-t másodszor használod ugyanabban a szkriptben:

Warning: Constant sid already defíned
(Figyelem: Konstans session-azonositó már definiálva van)
Warning: Cannot send session cache limiter - headers already sent
(Figyelem: A session cache limiter nem küldhető el - a fejlécek már el vannak küldve) Fatál error: Failed to initialize session modulé
(Végzetes hiba: A session modul elindítása nem sikerült)

A duplikált session-azonositó elkerüléséhez az új session-t manuálisan kell létrehoznod, a cookie-t és az új session-rekordot magadnak kell létrehoznod. A fejrészes hibát elkerülendő, addig kell a session-t törölnöd és újra létrehoznod, amíg az oldal még a HTTP-fejrész állapotban van. A session törlése és újbóli létrehozása helyett hagyd inkább érintetlenül azt, és a session_unset() manuális behívásával töröld a session összes változó-ját. A következő oldalon a látogatónak van session-azonosítója, de a session-ben nincsenek változók, és a szkript új változókat fog felépíteni, mint amikor a látogató először jön oda.

session_encode()

A session_encode() az összes regisztrált változót megragadja, a PHP-nak megfelelő módon kódolja őket, és egy a kódolt változókat tartalmazó sztringet ad vissza. Használhatod tesztteléshez, a session_decode()-ot pedig kikódoláshoz. A következő kód egy változó gyors tesztelését tartalmazza:

```
$title = "Articles for sale";
session_register("title") ;
$string = session_encode();
print("<br>Session string: "    htmlentities($string)) ;
```

session_end()

A session feldolgozása automatikusan befejeződik, amikor a szknpt véget ér, de előfordulhat, hogy a session hamarabb történő befejezésével csökkenteni akarod a fölösleges ráfordításokat. Ez különösen fontos frame-ek használatánál, mert a frame-frissítések-nek ugyanarra a session-rekordra kell várni. A frame-ekhez hagyd a session_start()-ot az utolsó percig a szkriptedben, majd végezz el minden session-frissítést, amilyen gyorsan tudsz, ma|d a session-rekord használatának befejezésére futtasd a session_end()-et. A szkript ezután szabadítja fel a session-rekordot a többi frame számára. Ez a függvény a PHP 4.0.6-os verziója után érhető el:

```
session  end();
```

session_get_cookie_params()

Amikor a session-kódodat cookie-kkal tesztelet, használd a session_get_cookie_params()-t a bejövő cookie-k értékeinek ellenőrzésére, ahogy itt látod:

```
$p    =    session_get_cookie_params();
while (Üst ($k,    $v)    =    each($p))

    print("<br>Cookie paraméter:                value:        $v)
```

A

A függvény egy tömböt ad vissza, amely tartalmazza a **lifetime-t,** ami a cookie lejáratának ideje, a domaint, amelynek egyezni kell a weboldalad domainjével, és a **path-t,** ha a cookie-t a weboldalad egy alkönyvtárára használod. A cookie tartalmazhatja a secure-t, ha Secure Sockets Layer-en (SSL) keresztül kell azt elküldeni:

session_id()

A session_id() az aktuális session-azonosítót adja vissza, és opcionálisan egy újat állít be. A következő kód megjeleníti az aktuális azonosítót, a **microtimeQ** használatával újat hoz létre, illetve beállítja és megjeleníti azt:

```
print("<br>" . session_id());
$x = explode(" ", microtime())                                    tű sfc
$id = session_id("id" . $x[l]    substr($x[0], 2)
print("<br>" . $id);
```

Ha nincsen cookie, akkor a session-azonosító elérhető egy állandó session-azonosítón keresztül is, és a session-azonosítóval lehet URL-eket létrehozni, ha az automatikus URL-létrehozás ki van kapcsolva:

```
if (definedC'SID") )                    i;5 ..-•sékc:. i.-.xw

    print("<br>Session id:      SID);
```

Rengeteg kód van az egyedi session-azonosítók létrehozására. A legtöbb a fantázián alapul. A session-azonosítótól egy dolgot vársz el: az egyediséget. A véletlen számok legtöbbször működnek, de éppen mivel véletlenek, akár két egymás után következőnek is lehet ugyanaz az értéke. Az egyedi szám létrehozására jó módszer a session-rekordok adatbázisba való mentése, és az egyes session-ök új kulcsait az adatbázis auto_increment szolgáltatásával lehet létrehozni.

Ha nem olyan biztonságos hálózatot használsz, mint például a Virtuális Magánhálózat (VPN), azt várod a session-azonosítótól, hogy nehezen kitalálható legyen, és az emberek ne tudják próbálgatásos technikákkal feltörni. Ehhez az egyedi azonosítót másmilyen szöveggel kell keverni, és az eredmény egy részét vagy egészét rejtjelezni. A 4. fejezetben számos rejtjelezési technikát találsz.

session_is_registered()

A session_is_registered() sztringként fogadja el a változó nevét, és igazat ad vissza, ha a változó regisztrálva van, és hamisat, ha nincs: *m*

```
$reg    =    "title";
if (session_is_registered($reg))

    print("<br>"    .            is
    $reg                      registered.");

else

    print("<br>"    .    $reg    .    "    is    not    registered.");
```

session_module_name()

A session_module_name() használatával kideríthetted, hogy a session.save_handler vajon files-ra vagy user-re van-e állítva, ahogy azt a következő kód első sora mutatja:

```
print(session_module_name());
print (session_module_name ("user") ) ;
```


Ezt meg is változtathatod, ahogy a második sorban látod. A változtatást csak teszteléskor lehet így elvégezni, mert az előtt kell megtörténnie, mielőtt bármely, a session-t elindító kódot használnál, és a session használata alatt ugyanolyannak kell maradnia. Egyszerűbb egy tesztoldal használatával tesztelni, mint dinamikusan megváltoztatni a session-t kezelő modult.

session_name()

A session_name() a php.ini-ben a session.name által definiált session-nevet adja vissza. Ha paraméterként új nevet adsz neki, a session_name() új nevet állít be. A nevet az előtt kell beállítani, hogy a szkriptedben megkezdenéd a session feldolgozását, mielőtt a session_start()-ot vagy a session_register()-t használnád. Amikor megváltoztatod a nevet, új cookie lesz beállítva. Az új cookie beolvasásához a session összes szkriptjében be kell állítanod az új nevet. Egy lehetséges ok, amiért egy speciális nevet állíts be az, hogy egy második session-t kezdj, amikor a felhasználó bejelentkezik. Amikor pedig kijelentkezik, eldobhatod az új session-nevet, és visszatérhetsz az eredeti cookie-hoz és session-höz.

A következő kód kiíratja az aktuális nevet, majd beállítja és kiíratja az újat:

```
print("<br>" . session_name());
print("<br>" . session_name("supersession"));
```

Az eredmény:

```
PHPSESSID
supersession
```

session_readonly()

Amikor frame-eket használsz, az összes frame-frissítés ugyanazon a session-rekordon osztozik és kizárják egymást. A zavart csökkentheted azzal, ha a szkriptet a session_readonly()-val kezded, hogy visszakeresse a session-változó értékeit. Amikor minden helyben van, nyisd meg gyorsan a session-rekordot a frissítéshez, majd zárd be a session_end()-del. A session_readonly() azáltal javítja a frame-eken belüli session-kezelést, hogy némely frame-nek megengedi a session-rekord frissítés nélküli beolvasását.

A session_readonly() még nem jelent meg, amikor ezt írom, így csak tippelek, hogy hogyan működik. A függvény benne volt a PHP-dokumentációban, majd eltávolították. A read only-opciót a session_start()-hoz adva ugyanezt az eredményt lehet elérni. A session_readonly()-hoz szükség van a hozzáillő session_update()-re, vagy arra, hogy a frissítéseket egy normális session_start()-tal szinkronizáld:

```
session readonly();
```

session_register()

A session_register() a session-rekordokban elmentésre kerülő változókat regisztrálja. Ha a session_register() előtt nem futtatod a session_start()-ot, akkor a session_register() indítja a session-t. A függvény korlátlan számú paramétert elfogad. Mindegyik paraméter lehet egy változó nevet tartalmazó sztring, egy változó neveket sztringekben tartalmazó tömb, vagy egy változó neveket tartalmazó tömbök tömbje:

```
Stitle = "Articles for sale";
$reg = "title";
if (!session_register($reg))

   print("<br>Register failed for " . $reg) ;
```

Ha egy változó regisztrálva van, ellenőrizheted azt a session_is_registered() használatával, a session_unregister()-rel pedig törölheted a regisztrálást.

session_save_path()

A session_save_path() a session-fájlok tárolásánál használt elérési utat adja vissza, illetve opcionálisan újat állít be. A következő kódban a session_save_path() ideiglenes könyvtárat állít be a session-fájloknak Windows NT és Windows 2000 alatt. Windows 98-ban használd a c:\temp\session-t:

```
print(session_save_path("t:/session") ) ;
```

A könyvtárnak a session_save_path() használata előtt már léteznie kell, és ha hatékony akarsz lenni, az elérési utat azelőtt kell beállítanod, mielőtt a session_start()-tal vagy a session_register()-rel elindítanád a session-t:

session_set_cookie_params()

A session_set_cookie_params() session cookie-t állít be, de mivel a session cookie-k automatikusan vannak beállítva, mire jó ez a függvény? A cookie-k manuális vezérlésével adott könyvtárakhoz adott cookie-kat állíthatsz be, és felülírhatsz olyan cookie-paramétereket, mint a lejárati idő:

```
session_set_cookie_params(time()+90000);
session_set_cookie_params(time()+90000,      "/test");
session_set_cookie_params(time()+90000,      "/test",      "test.com");
```

Felülírhatod a session élettartamát, ahogy a kód első sorában látod. Felülírhatod az időt és az elérési utat, ahogy azt a második sorban látod, vagy az időt, elérési utat és domaint, ahogy a harmadik sor mutatja. Figyeld meg, hogy az idő másodpercben van megadva, és a 86 400 másodperc egyenlő egy nappal. Azt is jegyezd meg, hogy ha a böngészőben - például a nyári időszámítás miatt - rossz idő és dátum van beállítva, akkor előfordulhat, hogy a böngésző a manuálisan beállított élettartam miatt leállítja a cookie-dat. Jobb, ha nem állítod be az időt, hogy a cookie ne szűnjön meg a böngésző session tartama alatt.

session_set_save_handler()

A session_set_save_handler() az összes session-rekord feldolgozó rutin nevét fogadja el, és regisztrálja őket a szkriptben való használathoz. Ezt a függvényt a session_start() és az első session_register() előtt használd, hogy a session nyitó és beolvasó függvények készen legyenek a session-feldolgozás kezdetén, A következő példában a session-kezelő-függvényeket a funkciójuk alapján neveztem el:

```
function session_open ($path, $name)

   return(true);
   }           function
session_close()
   {
   return(true);
   }             function
session_read($id)
   {
   return($data);
   }  function  session_write($id,
$data)
   {
   return (true);
   } function session__remove
($id)
   {
   return(true);
   }              function
session_gc($life)
```

```
    return(true);

session_set_save_handler("session_open", "session_close",
    "sessionread", "session_write", "session_remove", "session_gc");
```

A **session_open()** a session-fájlokra mutató elérési utat és egy session-nevet fogad, és kevés haszonnal jár, akárcsak a **session_close(). A session_read()** a session-azonosítót fogadja el, és beolvassa a session-adatokat, amelyeket kikódolva ad vissza. A **session_write()** session-azonosítót és a már kódolt session-adatokat fogadja el, és igazat ad vissza. A **session_remove()** session-azonosítót fogad el, és törli a session-t. A session_gc(), a hulladék-eltávolító alkalmanként lefut, és az időt túllépő vagy nem törölt session-ök rekordjait törli.

session_start()

A session_start() elindít vagy folytat egy session-t. A session automatikusan elindul a **session_register()** használatával, így vannak szkriptek, amelyekben a **session_start()-ra** semmi szükség sincs, én mégis mindig használom, hátha valaki később törli vagy elmozdítja a **session_register()**-kódot. A session_start()-nak bármilyen, a böngészőnek küldött outputot, így a fehér köz karaktereket is meg kell előznie:

```
session_start();
```

session_unregister()

Ez a függvény eltávolítja a regisztrációs lista egy változóját. A **PHP** 4.0.6-ban a függvény nem rendelkezett a **session_register()** többszörös paraméter és paramétertömb tulajdonságaival. A függvény hamisat ad vissza, amikor az eltávolítás nem sikerül, mint itt látható:

```
$reg = "title";
if(!session_unregister($reg))

    print("<br>Unregister failed for      $reg);
```

session_unset()

Ez a függvény egyenértékű az unset()-nek a session összes regisztrált változóján való lefuttatásával. Ha egy kijelentkezés alkalmával törölsz egy biztonságos session-t, futtasd le az **(miset)-et** az összes olyan személyes értékek, mint a jelszó egy mozdulattal való eltávolítására. Ezt követően a kód továbbmegy azokra a változókra, amelyeket egy nem bejelentkezett személy használ:

```
session_unset () ;
```

setcookie()

A **setcookie()** egy cookie-nevet és opcionális paraméterek halmazát fogadja el, és beállítja, hogy a cookie az aktuális HTML-fejrészekkel legyen továbbítva. A setcookie()-t azelőtt kell futtatni, mielőtt bármilyen HTML-outputot létrehoznál, mert a **HTML** megakadályozza, hogy fejrészeket, így cookie-kat küldj. A setcookie()-t használd a cookie-k beállítására, ha további cookie-kat, illetve ha hagyományos session cookie-kat akarsz, de nem akarod a

PHP automatikus cookie-jait használni. A következő kód kiad egy cookie-t és törli az ugyanilyen nevűt:

```
setcookie ("mycookie") ;
```

'" oiohiíhvx! *t* blcnfcj 32OÍ!

.?, **E jnox?rv** A

következő kód cookie-t állít be az összes paramétert megadva:

```
setcookie("mycookie", "pizza", time()+90000, "/php/", "phptect.com", 1);
```

Az első paraméter a név ("mycookie"), a második az érték ("pizza"), a harmadik a lejárati idő, amely az egy napnál kicsit hosszabbra van állítva. A negyedik paraméter a /php/-könyvtárra alkalmazza a cookie-t, az ötödik pedig beállítja, hogy csak a phptect.com [n] domain alatt működjön. Az utolsó paraméter azt szabályozza, hogy a cookie csak HTTPS-sel működik. A lejárat és a biztonsági paraméterek helyett lehet O-át írni, a sztringparamé-tereket helyett pedig lehet üres sztringet ("").

A cookie megjelenik ugyanezen böngésző következő oldallekérésében, ha az ugyanezen a domainen és könyvtáron belül van. Az érték egy a cookie-val megegyező nevű változóban jelenik meg. Amikor a PHP megkapja az előző cookie-t, az értéket a $mycookie változóba teszi, és a következő kóddal megjelenítheted azt:

```
print("<br>My Cookie: " . $mycookie) ;
```

1

Az eredmény ez lesz:

i. **es**

```
pizza
```

Cookie-t úgy törölsz, ha ugyanolyan névvel, egy üres értékkel, egy múltra állított lejárati idővel, ugyanazzal a könyvtárral és ugyanazzal a domainnel cookie-t küldesz. Az előző cookie törlésére ezt használd:

```
setcookie("mycookie", time () -90000, "/php/", "phptect. corn", 1);
```

Vásárlói szolgáltatások

A session-ökkel és cookie-kal szolgáltatást nyújtasz, nem pedig biztonságot, bevásárlókosarat vagy belépési korlátot. A cookie-kat kikapcsoló emberek is meglátogathatják honlapodat, és elérik a kontaktoldalt, amely arról ad információt, hogy mire használod a cookie-kat. Tedd lehetővé, hogy a session-ök a cookie-król URL-ekre váltsanak, amikor a cookie-k ki vannak kapcsolva, és mérlegeld azt, hogy egyáltalán nem használsz cookie-kat, illetve kizárólag olyan oldalon használsz, ahova a látogató egy létező cookie-val lépett be.

Bizonyosodj meg arról, hogy a cookie-jaid úgy vannak beállítva, hogy a domainedre, illetve ha lehet, a szerveredre térjenek vissza. Minden szerveren kell, hogy legyen olyan opció, amely csak azokat a cookie-kat engedélyezi, amelyek a küldő domainre térnek vissza, de a jelenlegi böngészők vagy nem adják ezt a lehetőséget, vagy úgy elrejtik, hogy lehetetlen kibogozni, vagy rossz választást adnak. Sajnos úgy tűnik, hogy az eredeti cookiemeghatáro-zások néhány munkahelyi gyakorlaton levő iskolássrác hirtelen döntésének az eredményei,

minden professzionális programozási vagy logikai elv híján, így csupa olyan böngészővel kell küzdened, amely a legalapvetőbb cookie-használati protokollt is nélkülözik.

Sose tárold a bevásárlókosár tartalmát cookie-ban; cookie-ban legyen a session-azonosító, minden mást viszont a szerveren tarts. Ha olyan oniine vásárlórendszerre bukkansz, amely az árakat cookie-ba rakja, akkor olyan rendszert találtál, amely ki van téve az árak csalási szándékkal történő megváltoztatásának. Rengeteg társaság kénytelen jelentős pénzügyi veszteséget elkönyvelni, mert képtelenek biztonságossá tenni a cookie-kat.

Ha egy alkalmazásban JavaScriptet veszel észre, problémás pontra tapintasz. Tízezer dollárért tanítom meg a weblapfejlesztőknek a következő titkot: Ne *használj JavaScript-et,*

Nézd végig a nagy, több millió dolláros fejlesztésekkel létrehozott weboldalakat, és 90 százalékuknál kódolási hibát találsz a legalapvetőbb képi rollover-szkriptekben is. Gondold végig, hogy milyen valószínűtlen, hogy a JavaScript "varázsló" működő bevásárlókosarat hoz létre JavaScript-ben, ha a varázsló rollover-kódja tele van lukakkal. Mielőtt bármilyen JavaScript-et hozzáadnál oldaladhoz, állj neki egy teljes és szisztematikus böngésző tesztelő programmal, amely jó néhány napba, emberbe és több ezer dollárba fog kerülni.

:s«9i xs> vn

U"?

>l086t6tl6QÍ0£e ki*.

vgori ,lorr*

.. lv '..-..-.-■ ■*

$■■_'-■,íí fi

Gyors megoldások

Session indítása cookie-kkal és fájlokkal *>a

Ez a megoldás a session-ök használatához minimálisan szükséges dolgokat mutatja. Csak ha ez már működik, akkor kísérletezz a változók regisztrálásával és a többi megoldás példáival:

```
[Session]; Handler used to store/retrieve data.
session.save handler = files
session.save_path = t:/session
session.use_cookies = 1
session .auto_start = 0
session.cookie_lifetime = 0
session .use_trans_sid = 1
url_rewriter.tags =
"a=href, area=href,frame=src,input=src,form=fakeentry"
```

A teszt Unix-rendszerhez a php.ini-fájl a megfelelő alapbeállításokkal kell, hogy rendelkezzen. A session.save_handler legyen files-ra állítva. Unix alatt a session.save_path alapértelmezetten /tmp-re van állítva. Windows-ban állítsd c:\temp-re. Az NT munkaállomásomon az eldobható tesztfájlok a t: meghajtón vannak, így lefoglaltam a t:/session-könyvtárat, amint azt a kiemelésben láthatod. A session.use_cookies alapbeállításban 1 a cookie-k használatához, de a session.auto_start 0-ra van alapértelmezésben állítva, így a session addig nem kezdődik, amíg nem használod a session_start()-ot. Ez az ideális tesztkonfiguráció, így a session kezdete előtt megváltoztathatod a beállításokat. Ideális a frame-ekhez is, amelyeknél minimalizálni akarod az időt, amíg nyitva van a session-fájl. Továbbá ideális az olyan oldalakra is, amelyek alkalmazzák a session-ök változtatását - például bejelentkezéskor. A session.cookie_lifetime alapbeállítása a 0, ami azt jelenti, hogy a session cookie addig marad meg, amíg a böngészőt be nem zárják.

A session.use_trans_sid úgy van beállítva, hogy a session-azonosítót az URL-jeidhez vagy linkjeidhez adja, ha a cookie-k nincsenek használva, így próbáld ki ezt a beállítást legalább egyszer úgy, hogy a böngészőben kikapcsolod a cookie-kat. A url_rewriter.tags azokat a HTML tag-eket listázza ki, amelyek automatikusan megkapják a session-azonosítót. Próbáld ki ezt a kikapcsolt cookie-kkal olyan oldalakon, amelyeken vannak olyan dolgok, mint a HTTP átirányító fejrészek, és figyeld meg, hogy mi az, ami nem kapja meg automatikusan a session-azonosítót. A függvényekkel létrehozhatod a hiányzó session-azonosítókat.

A HTTP átirányító fejrészeket a következő kóddal hozhatod létre:

```
header("Location: /test/index.html");
```

Ha a HTTP átirányító fejrészeket a fejrészek elején bocsátód ki, a böngészőt más oldal lekérésére kényszerítik (a HTTP-fejrészek nem HTML tag-ek, így az URL-újraíró nem írja őket újra).

A következő függvény egy elhelyezkedést leíró sztringet fogad el, és a session-azonosítóval adja vissza, amennyiben az definiálva van:

```
function session header ($location)
    *
    $location = "Location: " . $location; if
    (definedC'SID") )

        if (strpos(Slocation, "?") === falsé)

            $location .=

        "?"; else

        $location .= "&"; $location .=

    "session=" . SID;

    return($location);
```

A kód ellenőriz egy ?-lel jelölt *létező* lekérdezési sztringet, majd annak megfelelően ?- vagy &-jellel hozzáadja a session-paramétert.

A session-ök használatához a következő kódot add a szkripthez:

```
<?php
session_start() ;
?>
```

A kód fejrészeket fog küldeni, így ezt azelőtt kell a szkriptbe berakni, mielőtt bármilyen nemfejrész jellegű outputot létrehoznál. Ez azt jelenti, hogy nem lehet **print-** vagy echo-utasítás a session kezdete előtt, és nem lehet üres sor, kocsivissza vagy újsor a <?php-tag előtt.

A session mentési könyvtár úgy fog kinézni, mint a 19.1 ábrán, session-önként egy fájllal. Az összes fájlnév első része egy közös előtag, így a fájlt akkor lehet megkülönböztetni, ha egy normális megosztott könyvtárba helyezed. A fájlnév második része a session-azonosító. Figyeld meg, hogy a fájl hossza nulla, mivel a változók még nincsenek regisztrálva.

Name	Size	Type	Modified	Attributes
sess_512adcdefd83a3246066lee39d92077f	OKB	File	7/27/01	12 44 PM

19.1 ábra Session-fájl session mentési útvonallal

Teszteld most le a session_register()-t. A következő kód különböző típusú változókat állít be, amelyek között többdimenziós tömbök is vannak. Mindegyik regisztrálva van, így láthatod azt, hogy hogyan néznek kis egy session-rekordban:

```
$integer = 235;
$string = "test text";
$special = "special character:" . chr(245);
$array[] = "test";
$array[] = "test2";
$again["DE"] = "Germany";
$again["JP"] = "Japán";
$multiple[l] [1] = "a";
$multiple[l] [2] = "b";
$multiple[2][1] = "c";
if(!session_register("integer", "string", "special", "array", "again",
    "multiple"))

    print("<br>session register failed");
    1
```

A regisztrálás a session_register() használatakor történik, de a változók ténylegesen akkor lesznek a session-rekordba írva, amikor a szkript véget ér. A szkript végén a változóknak létezniük kell, és azt az értéket kell tartalmazniuk, amit el akarsz menteni. Ez alól csak az olyan szkript kivétel, amelyben session_end() van, mert az azonnal megírja a session-rekordba a változókat.

Hogy néz ki egy session-rekordfájl? A következő sor (a könyv tördelése miatt van csak megtörve) mutatja egy olyan fájlt tartalmát, amelyhez az előző regisztrált változókat rendeltem hozzá, és az alapértelmezett PHP-kódolást, nem pedig az opcionális WDDX-kódolást használtam:

```
integer|i:235;string|s:9:"test text";special|s:31:"quote " and special
character: >";array|a:2:{i:0;s:4:"test";i:1;s:5:"test2";}again|a:2:(s:2:"
DE";s:7:"Germany";s:2:"JP";s:5:"Japán";}multipleI a:2:{i:l;a:2:{i:1;s:1:"
a";i:2;s:l:"b";}í:2;a:1:{i:1;s:1:"c";}}
```

Az integer az első változó neve. A | jelzi, hogy a típus meghatározása következik. Az i az egész, az s a sztring, az a a tömb, az o pedig az objektum. A : jelzi, hogy attribútum következik, az első attribútum, a 235 pedig az egész értéke. A sztringek első attribútuma a hosszúságuk, a második pedig az értékük. A Sspecial-ben levő kétszeres idézőjel (") és speciális karakterek nincsenek például idézőjellel kimentve vagy speciálisan kezelve, mert a sztringet a hosszúság előzi meg, és a sztring definiálásához csak a hosszúság kell. A sztring körüli idézőjelek valójában csupán dekorációk a mezőhossz miatt.

A kapcsos zárójelek ({}) a tömbön belüli elemek definiálására szolgálnak, egy elem pedig lehet maga is tömb, ahogy a multiple-tömbben látható. A tömb összes eleme egy értékpár, amelyből az első jelenti az elem nevét, a második pedig az értékét.

Amikor rengeteg kód van a szknptben, és menet közben hozol létre változókat, tudnod kell, hogy a session regisztrálva van-e. Használd a következőknek megfelelően a session_is_registered()-öt, hogy egy létező regisztrálást ellenőrizz:

```
if(session_is_registered("integer"))

    print ("<br>integer is registered" );
```

Nem kapsz hibajelzést, ha egy változó egy szkriptben kétszer van regisztrálva, vagy ugyan abban a session-ben egy másik szkript is regisztrálja azt. A regisztrációs folyamat tesztelé sének egyetlen indoka egy esetlegesen óriási változólista hosszú regisztrációs folyamatának átugrása, amennyiben nincs rá szükség. . ■ ;":ój93» ■*

A következő' kóddal megpróbálom elmagyarázni, hogyan működik a regisztrálás:

```
if(isset($zzzz))                                                    i j i
                                ■r      •. f.Y.Í <!■)
    unset($zzzz);
    print ("<br>zzzz                                    -n ; ") -;s.~<; ip*-? .10
    unset");
    }
else                                                                    ín;
    $zzzz = "Sounds üke a mosquito.";
    print("<br>zzzz set");
    }                                   *^>'?r.■■{                  £ HÍ
if (session_is_registered("zzzz") )
    {
    print("<br>xxxx is already registered")


    session_register("zzzz");                   A ■ j!M h-fn>iyi-nfv ;>;>■>$> '.ro">
    print("<br>registered zzzz");                í-;
                                                -:í- -4-
```

A **kód** a **$zzzz** változót használja, ellenőrzi, hogy létezik-e, ha nem, akkor beállítja, ha pedig létezik, akkor kiiktatja. Egyúttal a kód regisztrálja a változót, amennyiben még nincsen regisztrálva. Az állapotot minden alkalommal megjeleníti, így láthatod, mi történik.

Amikor először futtatod a szkriptet, a következő eredményt kapod, mivel a változó nincsen sem beállítva, sem regisztrálva:

```
zzzz    set                                         ; t f :.£
registered    zzzz
```

Mikor másodszor futtatod a kódot, a következő eredményt kapod, mivel a változó be van állítva **session_start()-tal,** és regisztráltként van megjelenítve:

```
zzzz    unset                                       ^ „„„i,jti,.v ;! -.uv
xxxx    is    already    registered
```

Amikor a session_start()-ot futtatod, a session-rekord minden változója a memóriába kerül, és regisztrálva lesz. A változót nem kell minden szkriptben regisztrálni. A bejelentkezés **alatt** létrehozott változót lehet akkor regisztrálni, és később elfelejtheted, de amíg a session tart, a változó regisztrált marad. Amikor eléred a kijelentkezési oldalt, szüntesd meg a változó regisztrálását, hogy megakadályozd ugyanazon session későbbi szkriptjeibe való elmentését.

A következő eredmények a szkript harmadik futtatása után jelennek meg. A változó - mivel hiányzott — újra beállításra kerül, de a program még mindig regisztráltként ismeri fel. Hogyan sikerült a PHP-nak ez a trükk?

```
zzzzset                                    ■                                    ▮,ı
xxxx is already registered
```

A következő sor egy session-rekord tartalmát mutatja, amikor a változó regisztrált, de nincs beállítva:

```
!zzzz|
```

Figyeld meg, hogy egy felkiáltó jel (!) van a változó neve előtt, de nincsenek attribútumok a | után. A PHP így továbbítja oldalról oldalra a regisztrált változót akkor is, ha a változó nincsen beállítva.

A teszt teljessé tételéhez itt a kód a $zzzz regisztrálásának megszüntetésére:

```
if (! session unregister ("zzzz" ) )       .     .,
                                                    ujir

    print("<br>session_unregister                                    "{}
    failed.");
```

Kísérletezz a regisztrálás eltávolításával a kijelentkezési oldaladon. Jegyezd meg, hogy a regisztrálás eltávolítása nem iktatja ki az aktuális szkriptben a változót, csak megakadályozza, hogy a következő szkriptnek átadd.

Most már mindent megcsinálhatsz session-ökkel, igaz, egyelőre egy szerveren és frame-ek nélkül. Még sok minden hátra van, ha egy böngészőben több frame van, melyek ugyanazon a session-rekordon osztoznak, vagy ha több szerver teszi ugyanezt.

Session indítása MySQL használatával

Ha csak néhány változóra használsz session-fájlokat, akkor a session-fájlok területet pazarolnak az olyan fájlrendszereken, mint a FAT vagy az ext2. Mérlegeld az olyan fájlrendszerek, mint a NTFS vagy a Reiser használatát, vagy ugorj egyből az adatbázisban tárolt session-rekordokra. A MySQL kiválóan alkalmas a session-rekordokra, mivel minimális többletráfordítással tárolja a session-rekordokat, és minden olyan platformon elérhető, amely PHP-t futtat.

Problémáid lesznek a fájlok szerverek közötti megosztásával, ami a teljesítményt és a hozzáférési kérdéseket illeti, és a problémák exponenciálisan növekednek a szerverek számának növelésekor. Az adatbázisok törekednek az adatokon való osztozás könnyebbé tételére, és a teljesítménnyel kapcsolatos problémák is lineáris ütemben növekednek az exponenciális helyett.

Ha egyetlen szerverről van szó, az optimális teljesítmény érdekében tartsd a fájlokat a memóriában. Ha heap típusú táblázatokat használsz, akkor a MySQL engedi, hogy a memóriában tárold őket. Amikor heap-táblázatot definiálsz SQL használatával, írd type=heap-et a táblázat neve után, ahogy a következő SQL-töredék mutatja. A heap-táblázatoknak korlátozott indexelésük és SQL keresési lehetőségeik vannak, de a session-rekordokhoz ezek tökéletesen megfelelnek: *Cbi-*

```
create   table   test   type=heap                    «"'-v-   ^u.-«^   ■ - - ■
.•■■ »i   *,..,.v..;•!.-.
```

Ehhez a megoldáshoz az alapértelmezett MySQL-táblázat létrehozást használd, amely MyISAM típusú táblázatot fog eredményezni. A táblázatban nincs tranzakció-feldolgozó többletráfordítás, így a lemezalapú táblázat optimális teljesítményét kapod. A következő SQL egy session nevű táblázatot hoz létre a session-rekordoknak, melyben az id-mezőbe kerül a session-azonosító, a time-ba - a véget érő session-ök használatához - az aktuális idő kerül, a data pedig a kódolt session-adatokat fogadja:

```
create table
    session (
    id varchar(32) not
    null, time
    timestamp(14), user
    tinytext not null, data
    text not null, primary
    key (id)
```

A user egy további mező, amely a látogató nevét kapja, amennyiben az bejelentkezett, ezál tal gyorsan meg tudod jeleníteni, kik vannak bejelentkezve. Az SQL-t egy alkalmazáson, például phpMyAdminon vagy mysql_query()-n keresztül vidd be. A MySQL-függvényeket az 5. fejezetben mutatom be.

Hivatkozás:	oldal:
Táblák létrehozása	160

Az id-mező egy 32 bájtos meghatározott hosszúságú karakter, mivel ez a PHP által létrehozott session-azonosító mérete. A rekordot az elsődleges index bmáns keresésével találja meg, elég gyorsan. Ha az adatbázisod lehetővé teszi a fix hosszúságú rekordallokációt, készíthetsz egy kicsit gyorsabb elérési módszert, de korlátozva leszel abban, hogy mit tárolhatsz a session-rekordban, hiszen az egyes rekordok rögzített hosszúságúak.

Az időjelzés mezőre több adatbázisban is figyelni kell. A MySQL korábbi változatai az adatbázismezőket üresen hagyták sor beszúrásakor; az időjelzés csak sorok frissítésekor volt frissítve. A MySQL legutolsó változatában az időjelzés be van állítva a beszúrásra. Ellenőrizd az adatbázisodat - az időjelzésmező minden alkalommal be kell legyen állítva, mert egy üres időjelzésű sor lejártnak fog tűnni, és a hulladéktisztító rutin ki fogja törölni.

Jelezd a következő sort a szkript elejére beszúrva, hogy felhasználó által definiált session-kódot használsz (ez a sor meg kell, hogy előzze a session_set_save_handler()-t, aminek pedig meg kell előznie a session_start ()-ot):

```
session_module_name("user");
```

Add hozzá a következő függvényeket a szkripthez. A függvények a session vagy az oldal nevét (database) és az session-adatbázis nevét (table) adják meg a hátralevő kódnak:

```
function session db()

    return("petermoulding")
```

```
function session table()

    return("session");
```

A session_db() kiterjeszthető, hogy az adatbázist a **mysql_select_db()-n** keresztül válaszd, és egy megtelelő függvény megírásával újraválaszthatod azt az adatbázis, amelyet a session-adatbázis választása **előtt** választották ki. A kód tesztelésére használt rendszerben az Összes táblázat egy ofdalspecifikus adatbázisban van, így az újraelosztás nem szükségszerű, és a tesztoldal éles változatát kezelő Internet-szolgáltatónál jelenleg virtuális honlaponként egy adatbázis van, így nincsen újraválasztó kód. A MySQL-függvények közé tartozik a régebbi **mysql_db_query(),** amelybe paraméterként beadhatod az adatbázisnevet, de ez a függvény hibás, ezért el fog tűnni.

A következő függvénnyel megnézheted a session-függvények, így még a **session_write()** üzeneteit is, amely utóbbi akkor fut, amikor a böngésző output be van zárva. Megváltoztathatod ezt a függvényt, hogy oda küldje az üzeneteket, ahova csak akarod, így a rendszernaplóba is, de győződj meg róla a tesztelés alatt, hogy mindenhol van hozzáférésed. Azt javaslom, ezt a megközelítést addig használd, amíg a session-ök nem működnek, utána pedig távolíts el minden információs üzenetet, csak a hibaüzeneteket tartsd meg, és irányítsd őket a normál hibanaplódba:

```
function session_log{$message) {if($file = fopen("t:/session.txt", "a"))
    {
    fwrite($file, date("Y-m-d H:i:s ") . $message . "\n");
    fclose ($file);
```

Itt van annak a naplófüggvénynek a módosítása, amely a dátum/idő megjelenítést egy microtime()-mal helyettesíti, így pontosan láthatod, meddig van megnyitva egy adatbázis-kapcsolat. A területmegtakarítás érdekében az adatokat mellőztem, és ennek a változatnak az eredményét a megoldás későbbi részében mutatom meg:

```
function session log ($message)
    {
    if($file = fopen("t:/session.txt", "a"))
        {
        $m = explode(" ", microtime()) ;
        fwrite($file, date("H:i:s", $m[l]) . substr($m[0], 1) . " "
            . $message . "\n");
        fclose ($file);
```

Kell egy függvény a session-feldolgozás megnyitására, egy másik pedig ennek bezárására. Ezek mindössze egy naplóüzenetet produkálnak. A te függvényeid bármit meg tudnak csinálni, így akár egy távoli adatbázisszerverhez is csatlakozhatsz. Amikor 45 szervered kapja

az intelligens router által elosztott csatlakozásokat, az összes szervereddel csatlakozhatsz egy közös adatbázisszerverhez, amelyen csak egy adatbázis és egy táblázat van, kizárólag a session-rekordok számára. (Olyan adatbázist válassz, amelyet kiterjeszthetsz legalább két, villámgyorsan cserélhető adatbázisszerverre, felkészülve arra az esetre, ha ez egyik szerver leáll.) Amikor megosztott adatbázisszerverekhez csatlakozol, a session_start()-ot minél később tedd a szkriptedbe, hogy ezzel minimalizáld a csatlakozási időt. Használd az új session_end()-függvényt, amennyiben az elérhető, hogy a szkriptedben a lehető leghamarabb megszüntesd a kapcsolatot. Nézz utána az 5. fejezetben elmagyarázott állandó kapcsolatnak:

```
function session_open($path, $name)

    session_log("session_open");
    return (true); }
function session_close() {
    session_log("session_close");
    return(true);
```

Rögtön a **session_start{)** futtatása és a **session_open()** futása után az aktuális session-rekord beolvasására a PHP lefuttatja a **session_read()-et,** feltéve, hogy a rekord létezik.

A következő, felhasználó által írt függvény, a **session_read()** a session-azonosítót fogadja el, és egy session-adatokat tartalmazó sztringet ad vissza. Ha az adat nem elérhető, a függvény nulla hosszúságú sztringet ad vissza. A függvény adatbázis-kapcsolatot feltételez, amelyet létrehozhatsz a függvényben vagy korábban a session_open()-ben:

```
function session_read ($id)

    session_log("session_read");
    if(!mysql_select_db(session_db()))
        í
        session^log("session_read select database error: "
            . mysql_error()) ;
        return(falsé);
        }
    Ssql = "select * from " . session_table ()
        . " where id='" . $id . "'";
    if(!$result = mysql_query($sql))

    session_log("MySQL error: " . "
    with SQL: " . $sql) ; return (falsé)   mysql_error()
    ; ) if (mysql_num_rows ($result) )

        session_log("MySQL query returned '
            . " rows.");                    mysql_num_rows($result)
        $row = mysql_fetch_assoc(Sresult);
        session__log( "session_read returned
                                              $row["data"]);
```

```
    return ($row [ "data" ]

©1SS                                    JEpsmn
    session_log("session_read found zero rows with
    SQL:  " return ("");                                    $sql);
```

A függvény a megfelelő adatbázis kiválasztásával, a mysql_select_db()-vel kezdődik, az adatbázis nevét pedig a session_db() adja. Minden adatbázisfüggvény ellenőrzésre kerül, és hiba esetén a kód naplózza az üzenetet, ezt követően pedig kilép **(return())** a hamisat visszaadó függvényből.

Az SQL-lekérdezést, vagyis a $sql-t a session_table() táblázatnevének és a session_read()-be paraméterkém beadott session-azonosító használatával hozhatjuk létre. A **mysql_query()** elvégzi az SQL-lekérdezést, és siker esetén az eredményazonosítót, hiba esetén pedig hamisat ad vissza. A mysql_num_rows() a select-lekérdezés által megtalált sorok számát adja vissza, és azt akarod, hogy egyet adjon vissza. A session elején a nulla a kívánatos eredmény, az ezt következő beolvasásokkor pedig a hiba. Nincsen ellenőrzés a hiányzó session-rekordra, mert nincsen arra jelzés, hogy egy session-rekordnak mikor kellene léteznie. Még ha a session el is kezdődött, a rekord le is járhatott, és lehet, hogy a hulladékeltávolító rutin eltávolította. Nincs ellenőrzés a sorduplikációra sem, mert az elsődleges kulcs az azonosító, és az adatbázis megakadályozza a duplikációt.

A kód ellenőrizhetné, hogy a rekord lejárt-e, de ez a hulladéktisztító rutin feladata. Az olyan biztonsági rendszerekben, ahol a időkorlát nagyon rövid, végezz egy lejárati ellenőrzést is, mert lehet, hogy a hulladékeltávolító rutin a session lejárta utáni percekben még nem futott le.

A **mysql_fetch_assoc()** egy sort asszociatív tömbként ad vissza, ahol a kulcsok a mező nevek. A kódolt session-adat a $row["data"]-sztringben van, és a sztringet még kódoltan küldöd vissza. Ha a session-öket a felhasználói azonosító vagy más változók alapján ele mezni akarod, a változókat extra mezőkként hozzáadhatod a sorhoz, ahogy én a user-rel tettem, és választhatsz, hogy ugyanezeket a változókat benne hagyod a kódolt adatban vagy kihagyod őket onnan. Ha a változókat kétszer tárolod el, diszkrét mezőként és kódolt adatként, pazarlód a területet, de megspórolod azt, hogy a beolvasáshoz vissza kelljen állí tanod a változókat. Ha a változókat csak diszkrét mezőként tárolod, a saját felelősséged, hogy a beolvasáshoz visszaállítsd őket. . rt ,, ... ---------- ,-.

A session_write() - ahogy a lenti kód mutatja - a session-azonosító és a kódolt session-ada-tot fogadja el. A session-adatot megírja az adatbázisnak, ha ez sikerül, igazat ad vissza, hiba esetén pedig hamisat. A mysql_select_db() a session_read()-nek megfelelően kiválasztja az adatbázist. Az SQL, vagyis az $sql a session_table() táblázatnevének és a session-adat használatával jön létre. A session-adat addslashes()-be van téve, mert tartalmazhat egyszeres idézőjeleket, amelyek megzavarhatják az SQL-t. Amikor az adatok az adatbázisba kerülnek, a MySQL eltávolítja a hozzáadott perjeleket, így nem kell ezzel foglalkoznod, amikor visszakeresed az adatokat. Ha a php.ini-ben az automatikus perjelek be vannak állítva, és nem tudod őket kikapcsolni, hagyd ki a kódból az addslashes()-t.

A $PHP_AUTH_USER biztosítja a felhasználói azonosítót, így láthatod, hogy kiknek vannak aktív session-jeik. A mysql_query() felfrissíti az adatbázist, és igazat ad vissza, ha az SQL szintaktikailag helyes. A mysql_affected_rows() a frissítés által aktuálisan megváltoztatott sorok számát adja vissza: a 0 hibát jelez, az 1 a helyes működést, I-nél nagyobb szám esetén pedig rossz az SQL-ed vagy az adatbázis-táblázatod definíciója.

Amikor nulla rekord frissül fel, a kód azt feltételezi, hogy ez a session első frissítése, és helyette a SQL insert és egy másik mysql_query() használatával beszúrja a rekordot. Bizonyos adatbázisok engedik, hogy a frissítés automatikusan elvégezze a beszúrást, vagy engedik, hogy beszúrj valamit, ami egy létező rekord helyére kerül. A MySQL-ben van egy replace-parancs, amely felfrissíti a rekordokat, és újakat szúr be. Ellenőrizd az adatbázisod, és mérlegeld az update helyett a replace használatát, különösen, ha az adatbázis forgalmazója is ezt javasolja, lévén a replace gyorsabb, mint az update.

A külön frissítés és beszúrás egyik előnye az, hogy a frame-eken belüli több ablakból való frissítések között képes vagy a mezőket keresztellenőrzésnek alávetni, de ez messze meghaladja a jelen megoldás kereteit. A frame-ekre várnod kell egy kicsit, amíg az új session_end()- és session_readonly()-függvények elérhetők lesznek, vagy írj nekem egy üzenetet, hogy a létező függvényekkel elmagyarázzam a frame-ek kezelését:

```
function session_write(Sid, 3data)

    session_log("session_write") ;
    if(!mysql_select_db(session db()))

        session_log("session_write select database error: "
            . mysql_error());
        return (falsé);

    $sql = "update " . session_table ()                .ai Jica .
        . " set data = '" . addslashes ($data)
    . "'"; if(isset($PHP_AUTH_USER))

        $sql .= ", user='" . addslashes($PHP_AUTH_USER) . "'";

    $sql .= " where id='" . $id .
    "'"; if(!$result = mysql_query($sql))

        session_log("session_write error " .
            mysql_error() . " with SQL: " . $sql);
        return (falsé) ;
    if(mysql_affected_rows ())

        session_logí"session_write update affected "      i&oas EO .
            mysql affected rows() . " rows with SQL: " . $sql);
        return (true);
        }
    session_log("session_write updated zero rows with SQL: " . $sql); $sql
    = "insert " . session_table()
        . " set data = '" . addslashes(Sdata) . "', id='" . $id . "'";
    if(!$result = mysql_query($sql))
```

```
    session_log("session_write error " . mysql_error()
        . " with SQL: " . $sql);                             *
    return(falsé);

else                               itt'.: .:           :JÍí a í'i
                                               l-:r.?-víü:JÍí
                                              a

    session log("session write inserted with SQL:
    return (true) ;                                   $sql);
    )                                                         k t z
                                                              ,.4.
```

A kód egészen addig tökéletesnek tűnik, amíg nem teszteled. A MySQL jelenlegi változatában (és néhány más adatbázisban is) egy furcsa dolog történhet. Ha felfrissítesz egy rekordot, és az új értékek történetesen megegyeznek az előzőkkel, a MySQL nem tekinti a frissítést frissítésnek. Ez akkor is így van, ha az időjelzésnek automatikusan fel kell, vagy fel kellene frissülnie. Amikor az előző kód olyan frissítést végez, amely nulla frissítéssel jár, a kód megpróbál elvégezni egy beszúrást, és hibát okoz. Egyszerű a megoldás. A set data=-sort helyettesítsd az azt következővel. A módosított sor az időjelzés mezőt üresre állítja, ami arra kényszeríti a MySQL-t, hogy felfrissítse az időjelzés mezőt, és a frissítést annak is tekintse: /

```
. " set data = '" . addslashes($data) . "', time =
null";
```

A session_remove() egy létező session-rekordot távolít el, és a session_destroy() indítja el. A session_destroy()-t a kijelentkezés alatt is lehet session eltávolítására használni. A biztonságos session-rekordokból ilyenkor minden titkos dolog törlődik, és a kijelentkezett felhasználó számára új session-t hoz létre. A session_remove() eltávolítja a session-azonosítót, az SQL delete-en és mysql_query()-n keresztül törli a session-rekordot, és igazat vagy hamisat ad vissza sikeres művelet vagy hiba esetén. A kódod csak abban az egyetlen esetben fogja megpróbálni a törölt rekordot törölni, ha a felhasználó kijelentkezik, majd a böngészőjének a Vissza-gombjára kattintva megpróbál újra kijelentkezni.

A hibanaplózás és a hibaüzenetetek ugyanazok, mint a session_read()-ben és a session_write()-ban. Az egyetlen dolog, amit ebben a kódban választhatsz, annak ellenőrzése, hogy a felhasználó kijelenkezett-e, a SPHP_AUTH_USER el van-e távolítva, és kilépett-e az LDAP-ből vagy bármilyen más, kapcsolódó rendszerből. A függvény kikényszerítheti a minden más rendszerből való kilépést, vagy csak jelenti a hibát, ha a session-t az előtt szakították meg, hogy a felhasználó kijelentkezne a többi rendszerből: ."■' T" ,

```
function  session_remove ($id)                        éía-jCívs 'J;Y
    {
    session  log ("session  remove");    "'         '- (SÍTIJ) JVJUJP^.
    if(!mysql_select_db(session_db()))        :              ■

        session_log("session_remove select database error: "  ■
    :'    . mysql_error () ) ;                                SS.
 . _   return (falsé) ;               í-S' . " :..Q~: rí;-iví " .

    $sql = "delete " . session_table () . " where id=11' . $id . "'";
    if($result = mysql_query ($sql) )  - -           ■
```

```
í.  .5
        session_log("MySQL query delete worked.");
        return(true);                          .

    else
        {
        session_log("MySQL update error: " . mysql_error()
            . " with SQL: " . $sql);
        return (falsé) ;                    ·■'<■■■'■■ aï:c..c.í
```

A hulladékgyűjtés nem nagy szám az adatbázisban, csak intelligens adatbázis-tervezés és egy sor SQL kérdése. Mindössze azt akarod, hogy minden olyan session-re, amelyik lejárt, vagy a felhasználó bezárta a böngészőjét, és amelyen nem volt session_destroy(), elvégezz egy session_remove()-ot. Ha a session_remove()-nak van kijelentkezési kódja, és vannak olyan session-ök, amelyekben a felhasználó kijelentkezés nélkül zárta be a böngészőjét, akkor minden egyes lejárt session-ön végig kell futtatnod a session_remove()-ot. Először megmutatom ezt, és utána folytatom az egyszerűbb kódokkal.

A session_gc()-ben válassz ki minden lejárt session-t, az eredményeket olvasd be egy tömbbe, fuss végig ezen a tömbön, a session-azonosítókat pedig egyenként tápláld be a session_remove()-ba. A session_remove() törli a session-rekordokat, és elvégzi a kapcsolódó kijelentkezéseket.

A következő kód feltételezi, hogy nincsen ok a session_remove() futtatására, és az összes lejárt session-rekordot egyszerűen egy SQL delete-tel törli:

```
_ttfunction  session_gc (Slife)

    session_log("session_gc");
    if(!mysql_select_db(session_db()))

        session^log("session^gc select database error: " . mysql_error() ) ;
        return(falsé);

    Ssql = "delete " . session_table () . " where time < '"ií,1^..
        . date ("YmdHis", time() - $life) . .. ;          ;,n,
    print(p(red("session_gc sql: " . $sql)) );
    if($result = mysql_query($sql) )
        r JÍÍ.J3C
        session_log("session_gc deleted " . mysqlaffected_rows()
            . " rows.");
        return (true) ;                    ;í"ovome ■  •::: Í2P.í>a" ) po£  .ıí
        }                                  ( " i )<ib_nc IE3.V?) cíb  joo. aa_Ip3v-m:
    else                                ·'■ ■  ■ ■ M i i . - .  ii .;;-■ t- A  .  j ■

        session_log("session_gc error: " . mysql_error()
            . " with SQL: " . $sql);                    LV191
        return (falsé) ;                    . ; •■: Ü  ... . " ■{ -
    '   }                                   i-  ne i; r--~r.  . " .:--»í .,!," = íns?
```

A kód nagy része - mysql_select_db(), mysql_query() és a hibajelentés - ugyanaz, mint a session_read()-ben. Az SQL, vagyis az $sql az SQL delete-et és a session_table()-ból való nevet használja, és minden olyan sort kiválaszt, amelynek az időmezője korábbi, mint a lejárati idő. A lejárati időt úgy kapjuk meg, hogy az aktuális időből kivonjuk a PHP által a session_gc()-nek adott sessionélettartam-értékét.

Figyeld meg, hogy ez a megközelítés nagyon kevés gondolkodást vagy számolást tartalmaz. Némely emberek kísérletezgetnek azzal, hogy a lejárati időt a cookie-ban állítsák be, de ez hibákhoz vezethet, ha a felhasználó böngészője más időzónában van, vagy a böngészőnek nem teljesen olyan időalapú lejárati rendszere van, mint amilyenre számítasz. A lejárt session-ök eltávolítására a hulladékgyűjtést használni azzal jár, hogy a session-ök lejáratá hoz ugyanazt a dátumot/időt használod, mint a session-rekordok létrehozásához és frissí téséhez. Ezért a session-rekord hiányát tekintheted a cookie lejáratának is. Valójában soha nincsen szükség a cookie leállítására, mert a legrosszabb, ami történhet, az az, hogy valaki egy hosszú szünet után visszatér a böngészőhöz, olyan linkre kattint, amely cookie-t küld a szerveredre, majd egy új, üres session-t kap. Nem fog egy bejelentkezett session-t újra éleszteni, mert a szkripted nem fogja megkapni a biztosított session folyatatásához szüksé ges session-értékeket.

b iseea pO-UOi-de-dtljrő: *tt*

A következő kód összeköti az előző függvényeket: " ^u

```
session_set_save handler("session open", "session_close",
   "session_read", "session^write", "session_remove", "session_gc");
```

A session_set_save_handler() tájékoztatja a PHP-t, hogy melyik, a felhasználó által írt függvények végzik el az egyes session-függvényeket. A függvénynek a session_start() előtt kell lefutnia ahhoz, hogy legyen hatása. A PHP valamelyik fejlesztési változatában a session_set_save_handler() akkor is működött, ha a session_module_name()-et elfelejtetted.

A következő, session_start()-tal kezdődő kód az összes előző kódot használja:

```
session_start();
```

Amint a session_start()-ot használod, a PHP lefuttatja a session_open()-t és a session_read()-et. Ha egyedi csatlakozásod van egy session-adatbázishoz, allokálja az erőforrásokat és megnyitja a kapcsolatot. Az erőforrások a szkript végéig használatban maradnak, és függetlenül attól, hogy a szkript hogyan ér véget, a PHP csak akkor nem futtatja le a session_write()-ot és a session_close()-t, ha a szerver lángokban áll.

A következő kód teszteli a session-t a változók létrehozásával, regisztrálásával, illetve egy üzenet megjelenítésével: ;

```
if(!isset($zzzz))
   $zzzz = "Sounds üke a mosquito.
   print("<br>zzzz set" ) ; }
if(!session_is_registered("zzzz")) {
   session_register ("zzzz") ;
   print("<br>registered zzzz" );
```

~
-A

:*l* ,JS£

,37

A session-ön belül az első oldallekéréskor a tesztkód a következő eredményeket adja:

```
zzzz set
registered zzzz                                    ;ifM» ne-/!,-; r»«íhnim *«i
```

A változó be van állítva és regisztrálva van. A session-ben következő szkriptek nem eredményeznek üzenetet, mert a változó helyre van állítva, és a PHP session-kódja regisztráltként jelöli.

Hogy néz ki egy session-rekord a MySQL-táblázatban? A 19.2 ábrán a phpMyAdmin-on keresztül megjelenített MySQL-beli session-rekordot látsz. ntvi[r>] a

Hogy néz ki a session-napló? A következő lista egy a napló microtime()-változatát használó session-napló:

```
13:57:16.64906100 session_open '13:57:16.64980800 sessionread
13:57:16.65543500 MySQL query returned 1 rows.
13:57:16.65623200 session__read returned zzzz | s : 23 : "Sounds üke
a
   mosquito.";
13:57:16.95838700 session_write
13:57:16.96103100 session_write update affected 1 rows with SQL:
update
   session set data = ' zzzz I s :23 : V'Sounds üke a mosquito. \"; ' where
   id='9b7078 927131c3aledblfcbecf4 7e4af'
13:57:16.96346100 session_close
```

Figyeld meg a rendet: nyit, beolvas, ír és zár. A beolvasás közvetlenül a nyitás után, a zárás pedig közvetlenül az írás után történik. Ha egy frame-eket tartalmazó oldalt nyitnál meg, az első megnyit és beolvas a főoldalra vonatkozna, és az összes fennmaradó nyit, beolvas, és ír összekavarodna. Csak egy fájllal vagy sorzárolással tudod a hozzáférést besorolni, de a lezárás lassítja a hozzáférést és a következő oldalak megjelenítését. A hozzáférést gyorsíthatod a **session_start()** utolsó pillanatra hagyásával, és microtime-naplót használva ellenőrizheted a fáradozásaid eredményét. d-r■? -a i r< : ' , r :

Amikor már működnek a session-jeid, lásd el megjegyzésekkel a napló információbejegyzéseit, és mellőzd az adatok megjelenítését, mert az adatmezők hatalmasra nőhetnek.

id	tinié	user	data
9b7078927131c3aledblfcbecf47e4af	20010728121954		zzzz|s:23:"Sounds üke amosquito.";

19.2 ábra Session-rekord MySQL-táblában

in--

Az aktuális felhasználók megjelenítése

Ha az előző megoldásban felsorolt session-függvényeket használod, könnyedén kiírathatod az aktuálisan aktív online felhasználókat. A következő lista felhasználó szerint van rendezve, jelzi a felhasználói azonosítót, azt, hogy mikor kértek le utoljára oldalt, és a session-azonosítót:

user	time	id
Agnetha	20010728121954	9b7078927131c3aledblfcbecf47e4af
Anni-Frid	20010728183450	cd95db833aa0a98b72a673108892349c
Benny	20010728180308	df5bc09dd416e42f3044d86e90f0db51
Bjsrn	20010728183216	628720ele48457e94d28a4664e0b0987
Others:	2	

íibl

Rendezheted a listát idő szerint is, hogy felmérhesd a session-jeidnek legjobban megfelelő lezárási pontot. Ha a bejelentkezés alatt visszakeresed az egyes felhasználók nevét, e-mail-címét vagy telefonszámát, és ezeket hozzáadod a session-rekordhoz, illetve ehhez a listához, már kész is van a címlistád. Amikor fel akarod hívni a felhasználókat, hogy elmondd, hogy a szerver zsírozás és olajcsere miatt nem működik, hívd elő ezt a telefonszámos listát, amelyet a következő kóddal állíthatsz elő:

```
$sql = "select user, time, id from session order by user";
if($result = mysql_query ($sql) )

    $others = 0;
    print("<table>");
    while ($row = mysql_fetch assoc ($result) )

        if(!isset($heading>)

            Sheading = "<tr>";
            while (üst ($k, $v) = each($row))

                Sheading .= "<td>" . $k . "</td>";

            print($heading . "</tr>");
            reset(Srow);

        if(strlen($row["user"]))

            print("<tr>");

            while (üst ($k, $v) = each($row))     ft;

            print("<td>" . $v . "</td>");

        print("</tr>") ; else

            $others++;
            }          . .     " ' ■

    print ("<trXtd>Others :</tdxtd>" . $others . "</tdx/tr>") ;
    print("</table>");

else sjízobom
    {session_log("MySQL error: " . mysql_error() . " with SQL: "
    . $sql); return (falsé);
```

Az SQL egyszerűen egy felhasználó szerint rendezett lekérést végez el. A mysql_query() egy redményekkel teli kurzort készít, a mysql_fetch_assoc() pedig egyenként visszakeresi a sorokat. Az if(!isset(Sheading))-kód egy fejrészt állít be a mezőnevek használatával. A külső while()-ciklus a sorokon fut végig, a belső pedig a mezőkön. Az 5. fejezetben olyan változatait is megtalálod ennek a kódnak, amelyek szebb eredményeket és egyszerűbb változtatást tesznek lehetővé.

Hivatkozás:	oldal:
Adatbázis létrehozása	**160**

Figyelmet érdemel az Sothers kiszámítása. Ha egy sorban nulla hosszúságú felhasználómező van, a sort nem írja ki, csak hozzáadja az $others-ban levő számhoz. A táblázatsor kódjának utolsó sora tartalmazza ezt a másik számot, így megtudhatod, hogy hány látogatód van, aki nem jelentkezik be. Esetleg érdemes pufferolni a táblázat outputját, és előbb az Sothers-ben levő számot kiíratni.

Ha a hulladékgyűjtés úgy van beállítva, hogy csak ritkán fut, és az aktivitási szint alacsony, a listát javíthatod azzal, ha az adatbázis beolvasása előtt lefuttatod a hulladékgyűjtést. Ha az előző megoldás session-kódját használod, a session_gc()-függvényt bármikor meghívhatod a lejárt session-ök törlésére.

A session_end() használata

A session_end() nem létezik a PHP 4.0.6-ban, így itt pótlom azt, és azt feltételezem, hogy a tényleges PHP session_end() ugyanazt csinálja, mint az ebben a megoldásban alkalmazott függvény. A lényeg abban van, hogy a session_end() csökkenti az adatbázishoz való csatlakozás időtartamát, és használatával csökkenthető az az idő is, amíg a fájlok zárolva vannak a frame-ek megjelenítésekor.

juj-14
rtai!)

Ha fájlokat akarsz a session-rekordokhoz használni, vagy várd meg a hivatalos PHP session_end()-függvényt, vagy cseréld le a PHP beépített fájlfeldolgozását a "Session indítása MySQL használatával" című megoldásban használt kóddal, de fájlbeolvasással és -írással helyettesítve a MySQL-függvényeket. Hogy biztos legyél abban, hogy a fájl zárolva van a használat alatt, a session_open()-nel nyisd és a session_close()-zal zárd a fájlt.

Adatbázisok

"<t'J><tt\>:a: vrí *C<b; *áns*

Ha működik a "Session indítása MySQL használatával" című megoldásban levő kód, a következő módosításokkal helyettesítsd a normál szkriptlezáró session-t a felhasználó által írt session_end()-del:

■ >ÍÍÍUL'Í

.JÍ

```
function session_filler_close() {
    session_log("session_filler_close");
    return(true);
```

A változtatás első lépése két dummy-függvényhozzáadás, amelyek úgy tesznek, mintha megírnák a session-rekordot és zárnák a session-t. A következő függvény úgy viselkedik, mintha a session_close() lenne, de valójában nem tesz semmit. A session_close() semmit nem csinál a példakóddal, de módosítható, hogy zárja a kapcsolatot, és eltakarít bármit, amit a session_open()-ben beállítasz. A session_filler_close() csupán a session_close()-t helyettesíti a session_set_save_handler()-ben, így a session_end()-ben lehet a session_close()-t használni.

A következő session_filler_write() pontosan ugyanazt a semmit csinálja, mint a session_filler_close(), azzal a különbséggel, hogy ez a függvény más paramétereket fogad el:

```
function session_filler_write($id, $data)
{
    session_log("session_filler_write");
    return (true) ;
```

Változtasd meg a session_set_save_handler()-t, hogy a session_close-t session_filler_close-ra, a session_write-t pedig session_filler_write-ra cserélje, ahogy itt láthatod:

```
session set savé handler("session open", "session fillér close",
    "session_read", "session_filler_write", "session_remove",
    "session_gc");
```

Most a session_end()-del el kell végezned a session igazi írását és zárását. A következő függvény pontosan ezt teszi. Az egyik sor lefuttatja a valódi session_write()-ot a session-azonosítóval és a kódolt session-adatokkal. Egy másik sor a session_close()-t futtatja, majd egy sor visszaadja a session_write() által visszaadott értéket:

```
function session_end()

    session_log ("session_end for id " . session_id () ) ;
    $result = session_write(session id(), session_encode());
    session_close();
    return ($result) ;
    }
```

Ha a te session_close()-változatod használhatóan működik, a következőképpen terjesztheted ki ezt a session_end{)-et, hogy probléma esetén visszaadja az egyedi függvényértékeket. Eldöntheted azt is, hogy akarod-e a session_close()-t akkor is futtatni, ha van hiba a session_write()-ban:

```
function session_end()

    session_log ("session_end for id " . session__id() ) ;
    if(!session write(session id(), session encode()))
```

```
    return (falsé);

    return(session_close());
                                                    i  ! l ). F; se
```

A következő kóddal teszteld a megváltoztatott kódot: if(isset ($zzzz))

```
    print("<br>zzzz:    "   .   Szzzz   );

else {
    $zzzz = "Sounds liké a mosquito.";
    print("<br>zzzz set" );

if(!session_is_registered("zzzz") ) {
    session_register("zzzz");
    print("<br>registered zzzz" );

session_end();
$zzzz = "My brain after herbai tea.";
```

A kód értéket állít be a $zzzz változóban, regisztrálja a változót, lefuttatja a **session_end()**-et, majd megváltoztatja a $zzzz-ben levő értéket. Futtasd az oldalt többször, és minden egyes alkalommal nézz bele a sessionnaplóba, hogy phpMyAdmin vagy valami használó segítségével ellenőrizd a session-rekordot.

Az oldal következő és minden azt követő megtekintésekor a következő sort kapod:

zzzz: Sounds üke a mosquito.

A változó sohasem tartja meg a második értéket, mert a **session_end()** a session értékét a változtatás előtt menti.

Mennyi időt takaríthatsz meg a session-kapcsolódásokon és a hasonló dolgokon? A következő egy élő oldal megszerkesztett session-naplója:

```
14:51:22.62918300 sessionopen
14:51:22.62982100 session_read
14:51:22.76881400 session^end
14:51:22.76935400 session_write
14:51:22.77387100 session_close 14:
51:23.32397000 session_filler_write
14:51:23.32454800 session_filler_close
```

A session megnyitása és bezárása közti idő csupán 20,8 százaléka a session-nyitás és a PHP általi bezárása közti időnek. Ha az oldaladon hosszú oldalformázó rutinok vannak, sokkal többet is megtakaríthatsz, mint ebben a példában.

Ha távoli kapcsolatot használsz és állandó kapcsolatot nem, ahogy azt az 5. fejezetben elmagyarázom, akkor a **session_end()** használatával minimalizálhatod az erőforrások lekö-

töttségének idejét. Ha a session-rekordokat fájlokban tárolod és frame-eket használsz, minimalizáld az ugyanazon frame-en belüli kizárási időt a session_end() használatával.

RÁK

Ha a PHP session_readonly()-ja működik, bizonyos típusú szkriptekben tovább minimalizálhatod az erőforrások felhasználását és a kizárási időt, de sokat kell dolgoznod a session_readonly() teljes kihasználásáig. Egyebek között erőforrásokat kötsz le akkor is, amikor a session_readonly()-ban való beolvasáshoz csatlakozol az adatbázishoz, így ennyi erővel végezhetsz normál írást és beolvasást is. Ehelyett inkább koncentrálj a session_start() és session_end() közötti idő minimalizálására, és ha mindent minimalizáltál, küldj nekem a honlapomon keresztül a sikeredről egy üzenetet. Felállítok neked egy oldalt, amely a session_readonly() használatával segít a páratlanul hosszú szkriptekben további előnyöket elérned.

Ha a session_readonly()-t frame-ekben használod, egy másik frame frissíti az adatokat, te pedig egy normál session_write()-tal kísérletezel, akkor felülírod a másik frame frissítéseit. A megoldás a következőket tartalmazza: a session_end() használatával minimalizálni lehet az egyes frame-ek sessionrekord-használatát, a session-rekordban diszkrét mezőkben tárolni a frame-ek közötti üzeneteket, diszkrét frame-specifikus változók vagy frame-sepecifikus kódolt változók, és a rettenetes JavaScript-tel a többi frame frissítésre kényszerítése, hogy felvegyék az üzeneteiket. íme a legjobb terv. A főoldal szkriptje hajtódik végre először. Zárold a session-rekordot a főoldal frissítésére. A főszkript használatával állítsd be az összes frame-hez szükséges adatot. A session_end() használatával mentsd az adatokat a session-rekordba. Engedd, hogy az összes többi frame sorra kerüljön a session-rekord használatával. Ha egy frame üzenetet akar küldeni egy másik frame-nek, a küldő frame felfrissíti a session-rekordot, majd a JavaScript használatával frissíti a fogadó frame-et. A fogadó frame csökkentheti az erőforrások használatát, ha tudja, hogy a frame-frissítéshez szükséges-e a session-rekord frissítése vagy elegendő a beolvasása. A fogadó frame zárolhatja a rekordot a frissítéshez, beolvassa a rekordot, eldönti, hogy beolvas vagy frissít, majd feloldja a zárolást. A lezár, olvas, felold gyorsabb, mint a zárol, olvas, frissít, felold.

Jót

JMK ,

20. fejezet

XML

Gyors megoldások	oldal:
Az XML-fájlok megjelenítése	703
Az XML-adatok megjelenítése	**704**
XML-adatok értelmezése	**707**
XML nyitó- és zárótag-ek összeillesztése	**711**

Áttekintés

Mi az XML?

Az XML rövidítés a Kiterjeszthető Leíró Nyelv elnevezést (Extensible Markup Language) takarja. Az XML a Standard Általánosított Leíró Nyelv (Standard Generalized Markup Language, SGML) utóda és a HTML testvére, amely szintén egy SGML alapú leíró nyelv. Az SGML az Általánosított Leíró Nyelv (Generalized Markup Language, SGML) leszármazottja, ami az IBM-nél 1973-ban indított projektből bontakozott ki.

Dolgoztam a GML egyik korai kereskedelmi változatával, s munkánk az egyik első próbálkozás volt a GML egyszerűsítésére. Az én fő feladatom egy illesztőfelület létrehozása volt két számítógép - egy nagy teljesítményű online információs rendszer és egy hatalmas adattároló - között, ami valós idejű adatkonverziót tett lehetővé egy egyedi formátumból GML alapú formátumba. Ma a PHP és e könyv segítségével, no meg egy kis gyakorlással te két hét alatt meg tudod írni és le tudod tesztelni mindazt, ami egy nagyon tapasztalt Assembler programozónak egykor két hónapjába került. Az egész valós idejű adatcsere-problémát megkerülhetjük, ha az adatbázisunkat egyszer XML formátumúra alakítjuk. A kétheti munka legnagyobb részét az adatelemzés tenné ki, és körülbelül két nap lenne maga a programozás.

A www.w3.org/XML/ weboldalon számos olyan linket találsz, amelyek túlságosan is sok információt próbálnak kínálni az XML-ről. Először olvasd el ezt a fejezetet, gyakorold a Gyors megoldásokat, és csak aztán olvasd el a "hivatalos" anyagokat, amelyek még azt is leírják, hogyan kell feltenni a pontot az i-re vagy áthúzni a t betűt.

Mi a varázslat az XML-ben? Mit csinál? Mit nem csinál?

Miért csodálatos az XML?

Az XML előtt sokat vitatkoztak azon, hogyan kellene az adatokat megosztani. Az olyan egyszerű formátumok, mint a vesszővel elválasztott változók (Comma Separated Varible, CSV) egyszerű és hatékony módját jelentették az adatátvitelnek, de fájlonként csupán körülbelül egy adatbázis-táblázatnak megfelelő mennyiségű anyagot tudtak átvinni. Strukturált adatok továbbításához, mint amilyen mondjuk egy egész adatbázis vagy egy teljes dokumentum, rengeteg CSV-fájl bonyolult rendszerére volt szükség. S hogy a dolog még tovább bonyolódjon, néhányan nem értették meg a vessző használatát a CSV-ben, és helyette inkább tabulátorokat használtak. A sztringek idézőjelbe tétele könnyű, az idézőjeles sztringekből szintén könnyű kiszedni az idézőjelet, de akkor a végén a CSV-fájlok különböző idézőjeleket vagy eltérő idézőjel-kiszedési módokat tartalmaznak, esetleg néhány fájlban nincs is idézőjel a sztringek körül. A tucatnyi létező CSV-fájl formátum mellé képzeljétek el azt a több száz szoftverkészítőt, akik nem is törődtek a CSV-fájlokkal, s végeredményként máris végtelen számú egyedi fájlformátumot kapunk.

Az XML enyhít a problémákon azzal, hogy egyetlen közös fájlstruktúrát teremt meg. Ez a szerkezet lefedi azt, amit a CSV-fájlnak kellett volna, nem enged meg olyan önkényes változtatásokat, mint például a vesszők tabulátorokra cserélése, és elfogadja a hagyományos táblázat típusú és a kevésbé szokványos dokumentum típusú struktúrát is. Az XML lehetőséget nyújt arra, hogy a fájlba fájldefiníciót építsünk be, így a számítógép felismeri a fájlszerkezetet, és ellenőrizheti, hogy az új adatok megfelelnek-e ennek.

Mit csinál az XML?

Az XML úgy strukturálja az adatokat, hogy az egyes adatelemeket tag-ekkel választja el egymástól. A tag-ek segítségével lehetőségünk van arra, hogy hosszú, értelmes nevekkel lássunk el minden adatelemet, de rövid kódolt tag-eket is használhatunk a jobb helykihasználás érdekében. A tag-eknek lehetnek jellemző paraméterei, és a fájlnak lehet egy átfogó szerkezetdefiníciója, az úgynevezett Dokument Típusú Definíció (DTD) vagy egy külső DTD-re való hivatkozása, a Dokument Típusú Deklaráció. A deklaráció egy definíciót tartalmazó külső fájlt nevez meg, s az a definíció határozza meg a fájlstruktúrát.

Mit nem csinál az XML?

AZ XML nem készít jó cappuccinót, nem hozza el a világbékét, és nem segít az adatelemzésben sem. Az adatok meghatározásához még mindig neked kell elvégezned az adatelemzést és az XML-szerkezet kifejlesztését. Ha adataidat szabványos XML-struktúrában akarod használni, akkor továbbra is a szabványhoz kell igazítanod. Tegyük fel, hogy cipőket gyártasz, és létezik egy nemzetközi XML-szabvány a cipőgyártók bőrszállítók felé továbbított megrendeléseire. Ekkor neked még mindig ki kell találnod, hogyan illeszthető rá a definíció a te adataidra. Meg kell határoznod, melyik mezőben legyen a bőr minősége, milyen fokozatokat lehessen a fájlban megadni, és azok hogyan kapcsolódjanak a te rendszereden tárolt fokozatokhoz. Nem sok értelme van egy olyan XML-fájlt elküldeni, amelyben a/d és *nagyon jó* fokozatok szerepelnek, miközben a fogadó gép A, B és C fokozatokat vár.

Ha az XML-fájl tartalmaz egy szerkezetdefiníciót vagy egy külső definícióra való hivatkozást, a számítógép automatikusan elvégezhet néhány ellenőrzést az új adatokon, még mielőtt a fájlba illesztené azokat. Ennek ellenére mindig meg kell bizonyosodnod arról, hogy a definíció teljes, illetve arról, hogy mi az, ami hiányzik. Hiányzó definíciórészek miatt érvénytelen adatok is beillesztésre kerülhetnek.

A www.xml.org/xml/registry.jsp weboldalon megtalálhatók különböző iparágak XML definíciói. Keresd meg az adataidhoz legközelebb álló szabványmeghatározást, majd gondold végig, hogy tudod-e használni vagy változatnod kell saját adataidon, hogy illeszkedjenek a szabványhoz, esetleg egy teljesen új szabványra van szükséged. Néhány ilyen szabványt iparági bizottságok alkottak meg, ezért azokból kitűnően látszik, mire van szüksége az * adott ágazat szereplőinek: egy szerény átlagos megoldásra vagy a gyakorlatban használhatatlan túlbonyolított valamire. Gondold végig, hogy mire képes az XML, hogy mire alkalmasak az iparágadban fellelhető XML-definíciók, valamint hogy mire van szüksége az adataidnak, majd ezek fényében válassz egy megoldást.

AZ XML nem helyettesíti a HTML-t

Benőit Marchal XML-ről szóló kiváló könyvében, az *XML példákon keresztül* című fejezetben a szerző azt állítja, hogy az XML a HTML helyettesítése. Ez enyhén szólva a dolgok leegyszerűsítése. Az XML a HTML egy részét helyettesíti csupán (a formátum definíciót) de a tartalmat nem (az önálló tag-eket). Az XML megmondja, hogyan készíts egy <hl> tag-et, de azt már nem, hogy mit csinál a <hl> tag. Az XML lehetőséget nyújt új tag-ek dinamikus hozzáadására a régi böngészők kiakasztása nélkül, azt viszont már nem mondja meg a böngészőnek, hogy a tag-nek új sort kell-e kezdenie, milyen legyen a szöveg színe, hogy egyáltalán jelenítsen-e meg a tag szöveget. A tag-ek tartalmát az XHTML, a HTML XML-formátumba átírt változata határozza meg.

Adatok

Most egy egyszerű XML-adatsor következik a Gyors megoldások között található egyik példából. A kisebb jel (<) indítja az XML tag-et, a nagyobb jel (>) pedig zárja. Az indító tag egy tag nevet tartalmaz, körbevéve a < és a > jelekkel, az ennek megfelelő záró-tag pedig a nyitás után (<) egy perjelet (/) tartalmaz. Úgy néz ki, mint a HTML, és az XML-fájlok könnyen is szerkeszthetők a legtöbb HTML-szerkesztővel:

```
<p>mustard  seed</p>
```

Hogyan kezeli az XML azokat a speciális karaktereket, amelyek a CSV-fájloknál problémát okoznak? A következő szöveget AbiWord-ben gépeltem, ez egy szövegszerkesztő, ami az XML-fájlban dokumentumokat tárol:

```
Start  tag:  <
End  tag:  >
Tab:
Single quote: '
Double quote: "
```


íme a szöveg, ahogy az XML-fájl tárolja. A kisebb/nagyobb jelek HTML stílusú speciális karakteralkalmazások lesznek, &-jellel az elején és kettősponttal a végén. A tabulátor, a szimpla és dupla idézőjel változatlan maradt:

```
<p>Start tag: &lt;</p>
<p>End tag: &gt;</p>
<p>Tab:       </p>
<p>Single quote: '</p>
<p>Double quote: "</p>
```

A tageknek szinte bármilyen nevet adhatunk, néhány nevet megtartottak különleges esetekre. Az XML-megjegyzéseket a HTML-megjegyzésekhez hasonlóan felkiáltójel, kötőjel, kötőjel (!--) kombinációval jelöljük a nyitás (<) után, és két kötőjellel (—) a zárás (>) előtt, így:

```
<!-- This file is an AbiWord document. -->
```

Mivel az XML-nek csak egyféle változata létezik, ezért az XML-adatok kezelésének is; de ha ez egyszer megváltozna, az XML-fejlesztők egy speciális tag-et is megadtak, ami azonosítja a fájlt létrehozó XML-variációt:

```
<?xml    version="l.0"?>
```

Vegyük észre, hogy a tag <?-lel és a nyelvnéwel kezdődik, úgy, mint a PHP-szöveg. Fontos, hogy a PHP-szövegeket < ? php-vel kezdjük a < ? helyett, mert így a PHP-szövegek XML-kompatíbilisek, és a PHP-kódtöredékeket más oldalakkal együtt XML-adatbázisokba tárolhatod. A végén minden szerkesztő és Internet-szolgáltató az XML-t fogja használni.

A tag-eket többféleképpen írhatjuk meg. A HTML-ben definiált break tag az itt következő három forma bármelyikében írható. Mindhárom elfogadott az XML-ben, de az elsővel vagy az utolsóval az XML-kompatíbilis böngészők esetleg nem működnek megfelelően:

```
<br/> <br                                                    : hí
/>
<brx/br>
```

Az apró részletek végzetesek lehetnek. Van-e szóköz a sztnngben? /? Lehet-e üres egy tagpár?
 </br> lehet üres, de a bekezdés tag-et nem üresként definiáljuk, így a   nem elfogadott (csak a XHTML-ben nem az, az XML-ben igen).

Az adatok struktúrájára és tartalmára vonatkozó szabályokat a DTD tartalmazza, így a DTD-t mindennek be kell olvasnia, ami az XML-adatokat kreálja, hogy az adatok biztosan helyesek legyenek.

A szövegsorok nem befolyásolják az XML tag-eket. A következő példa egy elég hosszú megjegyzéstag első- és utolsó két sorát mutatja az egyik XHTML DTD-ben. Mikor az XML beolvassa a sortöréses adatokat, a sortöréseket eltünteti, mielőtt lefordítja a tag-eket.

```
    Extensible HTML version 1.0 Strict DTD
```

Ez a példa szintén egy többsoros tag az XHTML-ből:

```
<!ENTITY   %   focus
   "accesskey       %Character;     #IMPLIED
   tabindex         %Number;        #IMPLIED
   onfocus          %Script;        #IMPLIED
   onblur           %Script;        #IMPLIED"
```

Vegyük észre a tag-név előtti felkiáltójelet, a százalékjeleket (%), és a # jel használatát. A DTD-k furcsa tag-eket és attribútumokat tartalmaznak, amelyek együtt már majdnem úgy viselkednek, mint egy programnyelv. Ha neked kell DTD-t írnod, és azon veszed észre ma gad, hogy a DTD-k által megengedett valamennyi lehetőséget használod, akkor valószínű leg kihagytál valamit az adatelemzési szakaszban, és vissza kell venned az adatellenőrzési szabályokból. . . . . .

Korábban említettem a HTML <hl> tag-et. íme a <hl>-tag XHTML-definíciója:

```
<!ELEMENT hl %Inline;>
<!ATTLIST hl %attrs;
```

A <hl>-definíció a %inline-ra vonatkozik, ahol a % helyettesítést jelent. A %inline azt jelenti, hogy a %inline-sztringet egy inline nevű, ENTlTY-definícióban meghatározott sztringgel kell kicserélni. A %attrs-t az attrs-definíciójában található sztringgel cseréljük ki. Következik az inline definíciója:

```
<!ENTITY % inline "a | %special; I %fontstyle; |
    %phrase; |%inline.forms;">

<!-- %Inline; covers inline or "text-level" elements
—> <!ENTITY % Inline "(#PCDATA | %inline; |
%misc;)*">
```

Hoppá, kettő is van. A neveknél számít a kis- és nagybetű, az XHTML alkotói egy névtípust két megjelenésben használtak. (Remélem, saját DTD-dben sosem csinálsz ilyet.) A <hl>-hez %inline és nem %Inline tartozik. Majd a %Inline tartalmazza a %inline-t. Brrr. Az ilyen elnevezési zavarok olyan hibákat hoznak, amelyek akkor vezetnek problémákhoz, miután a weboldal élő lesz, mikor emberek milliói látogatják az oldalt, és minden egyes hiba 100 USD veszteséget jelent a vállalatodnak.

Külső elemek

Az XML-fájIban leírt XML-elemeket belső elemeknek nevezzük. A külső elemek olyan elemek, amelyek egy helyi szerveren található fájl külső definíciójára mutatnak, vagy egy másik szerverre. Az előbbi kód egy belső elemet mutat, a következő elem pedig helyi fájlra mutató külső elem:

```
<!ENTITY headingl SYSTEM "headingl.ent">'~".:3i'»V .'KTíl filc 12 i
```

A következő elem külső, és egy fájlra hivatkozik URL-en keresztül:

```
<!ENTITY headingl SYSTEM "http://petermoulding/headingl.ent">
```

Nem értelmezett elemek f ;lsy^-"~ ^

Az elemek nem XML-adatokra is utalhatnak a nem értelmezett elemeken keresztül. A nem értelmezett olyan adatot jelent, amit nem akarsz XML-ként elemezni. Képfájlok, multimédiafájlok beillesztésére szolgál, vagy bármi máséra, ami nem XML-kompatíbilís, de a programod tudja használni, vagy továbbítható egy böngészőnek. Ha XML-adatokat elemzünk, hogy felvehet]ük-e őket egy oldalra, és nem értelmezett elemmel találkozunk, célszerű a hivatkozást egy HTML kép- (<img>) vagy anchor tag-be (<a>) beszúrni, aztán hadd dolgozzon a böngésző:

```
<!ENTITY image SYSTEM "earth.jpg" NDATA JPEG> iöi X£ limtbv Í£:
```

Komplex DTD-k írása ^f-i^k 31

Mielőtt a DTD-khez nyúlsz, fejezd be ezt a fejezetet, vedd végig a példákat, aztán fogj egy XML-ről szóló jó könyvet. Ne feledd, ha összezavar egy konstrukció a DTD-ben, puszta remény sincs rá, hogy egy hétköznapi halandó megértse a konstrukciót, és néhány a DTD-t használó szoftver is ugyanezzel a problémával szembesül majd. Adataidat és a DTD-ket tartsd egyszerű formában.

Szerkezet '

Az AbiWord-fájlokban **<section>** tag-ek vannak, amelyek <p> tag-eket tartalmaznak. A DTD adja a szabályokat, és az AbiWord DTD <section>-definíciója szerint a **<section>** a következőképpen tartalmazhatja a <p>-t:

```
<section><p>A  paragraph.</px/section>
```

Ha a DTD megmondja, hogy a következő <section> a <p>-n belül elfogadott-e akkor egy bekezdés elfogadott egy bekezdésen belül, ami logikailag nehezen értelmezhető. Amikor DTD-t hozol létre, szabályokat kell kialakítanod, hogy elkerüld az olyan helyzeteket, ahol egy adat nem értelmezhető:

```
<p><section>A paragraph.</section></p>
```

A következő szerkezet egész biztosan nem elfogadott. Az első rész a <section>-t mint a <p> szülőjét adja meg, a tag szülőjén kívül viszont nem lehet a tag-et befejezni:

```
<section><p>A paragraph,</section></p>
```

Nevek

Az XML tag-nevek betűvel vagy alsóvonással (_) kezdődnek. A név többi része lehet betű, szám, alsóvonás pont (.) vagy kötőjel (-). Az xml-nevekre lefoglalt előtag, a kettőspont (:), bár a nevekben elfogadott, az XML-nevek közötti szóközökre fenntartott. A nevekben meg kell különböztetni a kis- és nagybetűket, de kérlek, ne zavarj össze senkit ezek váltogatásával.

Attribútumok

Az XML tag-nek lehetnek attribútumai. Ha kiválasztunk egy betűtag-et az XML-ben, színattribútumot rendelhetünk hozzá, a tag-et így gépeljük be: **<font color="green">.** Az XML-ben az attribútumok köré is idézőjelek kellenek. A HTML-ben nem szükségesek az idézőjelek, **bár** megengedettek; vagyis mindig használd őket a HTML-ben, és akkor a HTML-ed XML- és XHTML-kompatíbilis lesz.

CDATA " ■ ' * ■ ^--^

Néha az XML-adatok között lehet olyan, ami az XML-feldolgozó számára XML-nek tűnik, vagyis szeretnénk, hogy ezeket a speciális adatokat a sztenderd XML-behatárolókon kívül más is behatárolja. Ha az adataidban egy ilyen formula található: A < B > C, azt úgy olvassuk, hogy A *kisebb,* mint B, *és* B *nagyobb,* mint C. Mégis, az XML-processzor a <B>-t mint B nevű XML tag-et próbálja definiálni. A CDATA-t a <[CDATA[és]]> írásjelek határolják be, és minden más CDATA-n belüli speciális betű és körülhatároló karakter, a <B>-t is beleértve, nincs figyelembe véve.

DTD

A DTD (Document Típusú Deklaráció) az XML-fájl sémája. Az XML-fájl megalkotása előtt fel kell állítani egy sémát, hogy tudjuk, mi lesz és mi nem lesz megengedett a fájlban. Ha egy XML-fájlt olvasunk, mi döntjük el, hogy az adatokat érvényesnek fogadjuk el, vagy a séma alapján érvényesítjük,

A DTD az összes XML-fájlban található elemet és attribútumot meghatározza. Te adod meg, hogy melyik elem melyikbe legyen beágyazva. Az elemeket és az attribútumokat kötelezővé vagy szabadon választhatóvá teheted. Képzelj el egy termékkatalógust tartalmazó fájlt. A *súlymezőt* kötelezővé teheted, hogy mindig ki tudd számolni a fuvarköltségeket. A *színmezőt* esetleg szabadon választhatónak hagyhatod, mert néhány áru csak egyféle színben érkezik.

Ne feledd, tulajdonképpen kétféle DTD létezik. A Dokument Típusú Definíció az XML-ben meghatározott séma. A Dokument Típusú Deklaráció egy <!DOCTYPE>-tag, amely egy külső Dokument Típusú Definíció-fájlra mutat. A külső deklarációban több fájlnak azonos definíciója van.

A DTD-k sokkal komplikáltabbak tudnak lenni, mint kellene. Ha nem tudod elmagyarázni a sémát annak, aki adatokat készít elő a fájlod számára, valószínűleg túl bonyolult sémát készítettél. A rövid tag-nevek helyet takarítanak meg a fájlban, viszont az egymáshoz kapcsolódó fájlok között hasonló nevek jelenhetnek meg, aminek következményeként az emberek könnyen összekeverhetik a neveket. Ha kiküldünk egy híreket tartalmazó XML-fájlt és egy időjárás-jelentést tartalmazót, győződjünk meg róla, hogy az azonos elnevezésű tag-eknek pontosan ugyanaz a definíciójuk és jelentésük is, valamint, hogy a különböző jelentésű elemeknek megkülönböztethető a nevük is. így az emberek az időjárás-jelentés elemeinek jelentését nem fogják összekeverni a hírek hasonló elemeivel.

Az elemazonosítót az <!ELEMENT> tag-gel definiáljuk, ahogy az itt következőt. A sor tartalmazza az elem nevét, azt a nevet, ami az elem tag-jében lesz, majd azon elemek listáját, amelyek ebben az elemben szerepelnek:

```
<!ELEMENT product (name, price, weight, color)>
```

A példa elemdefinícióval a következő termékadatot vihetjük be:

```
<productxname>truck</name><price>$120, 000</price>
   <weight>5 tons</weightXcolor>Red</colorX/product>
```

Ha már ismerjük az elemdefiníciókat, általában a DTD-ből megismerhetjük egy XML-fájl mezőtartományát. Mikor XML-fájlt készítünk, minden a DTD-kben foglalt szabályt ismernünk kell, valamint azt sem árt tudnunk, hogy hogyan találjunk egy jó könyvet, vagy weboldalt. Ha az XML-ről szóló legjobb és legfrissebb könyveket keresed, küldj egy üzenetet a weboldalamra. Én mindegyiket elolvasom, egyikben esetleg érthetőbbek a magyarázatok vagy jobbak a diagramok.

Van még néhány szabály, amelyek megkönnyítik a DTD-olvasást, hogy megértsük az XML-fájl struktúráját. A product-elem definíciójában a name módosítás nélkül jelenik meg, ami azt jelenti, hogy a <name> tag-nek pontosan egyszer szabad megjelenni a <product>-ban. Ugyanez vonatkozik a felsorolásban szereplő többi tag-re is. Ha egy elem neve pluszjellel (+) végződik, az elem egyszer vagy többször is megjelenhet. Ha a product definíciójában szerepelt a color+, a product több <color> tag-et is tartalmazhat, a következőképpen:

```
<productxname>truck</name><price>$120,000</price>
    <weight>5    tons</weightXcolor>Red</color><color>Green</color>
<color>Blue</colorx/product>
```

Ha az elem neve csillaggal (*) végződik, az elem nullaszor vagy többször jelenhet meg. Ha kérdőjel (?) áll a végén, nulla vagy egy alkalommal jelenhet meg az elem. Ha az elemek neveit függőleges vonal (|) választja el, az azt jelenti, hogy csak egyetlen elemet választunk. Ha a product- (termék) elem a következő definícióval szerepel, akkor a product-nak lehet color- (szín) eleme vagy fabric- (anyag) eleme, de nem lehet mindkettő:

```
<!ELEMENT product (name, price, weight, {color | fabric))>
```

Figyeljük meg a zárójelet, amely az elemek alcsoportjait csoportosítja. A zárójelekhez módosításokat is adhatunk. Ha a color és a fabric is megengedett a product-ban, és mindkettő nulla vagy több alkalommal jelenhet meg, akkor így is bevihetjük őket: (color,fabric)*.

Legutoljára az attribútumdefiníciókat kell elolvasni, hogy ki tudjuk dolgozni, milyen attribútumok jelenjenek meg az adatokban. Nincs értelme 30 000 000 adat feldolgozásának csak azért, hogy a 29 897 465. adaton új attribútumot fedezzünk fel. A DTD-k tartalmazhatnak ATTLIST-definíciókat, amelyek megmutatják, milyen attribútumok jelennek meg egy elemben, és meghatározzák azok jellemzőit, így tudhatjuk, mire számíthatunk egy XML-fájl olvasásakor. Az attribútumoknak alapértelmezett értéke is lehet, így az alapértelmezést lehet használni az első 29 897 465 adatnál az attribútumok helyett.

A következő attribútumdefiníció szerint a fabric-elem attribútuma a fire-proof, amely lehet igaz vagy hamis, és alapértelmezésben hamis. Az alapértelmezés akkor érvényes, ha a fabric tag-et a fire-proof-attribútum nélkül találjuk. Mikor XML-fájlokat olvasunk, csak az elem nevére és az attribútum nevére van szükség. Amint két neved van, meg tudod találni és meg tudod jeleníteni az attribútum értéket. A legtöbb adatnál ez minden, amire szükség van:

```
<!ATTLIST fabric fire-proof (true | falsé) "false">
```

Névmezők

Ha definiálsz egy XML tag-et a fájlodban, és azt a fájlt elküldöd nekem, én használhatom a saját tag-jeimet, ezekkel adatokat adhatok a fájlhoz, s azt elküldhetem egy harmadik em-

bérnek, aki nem tudja, melyik tag kihez tartozik, melyikért kit hibáztasson. Az XML-tag tulajdonosazonosításának problémáját megoldhatod az XML-névmezőkkel. A dokumentumod tetején elhelyezel egy névmezőutalást weboldaladnak arra a lapjára, ahol tag-jeidet dokumentáltad. Az URL-nek egy rövid előtagot adsz, s ezt az előtagot ragasztod minden tag-ed elé. A <heading>-tag (fejléc) lesz a <your:heading> (a te fejléced), és ha én is egy heading nevű tag-et adok hozzá, akkor az lesz a **<my:heading>** (az én fejlécem).

A következő xmlns-paraméterek az XML-fájl tetején egy tag-ben találhatók, és minden előtagot a megfelelő weboldalhoz kapcsolnak:

```
<tag_name    xmlns:your="
             http://your_web_síte.com/tags/"
             xmlns:my="http://petermoulding.com/tags/">
```

Az xmlns-paraméterek olyan tag-be találhatók, amely az előtagos tag-eket veszi körül. Ez azt jelenti, hogy különböző előtag-„szettet" használhatsz a dokumentum minden egyes részénél.

Xlink és XPointer

Az XML Kapcsoló Nyelv (XML Linking Language - Xlink) segítségével XML-dokumentu-mokba inzertálhatunk elemeket, hogy a dokumentumon belül mozogjunk, vagy más dokumentumokhoz eljussunk, a HTML-fájlokhoz hasonlóan. A specifikációt a **www.w3.org/TR/2001/REC-xlink-20010627/** alatt találod. Az XML Mutató Nyelvvel (XML Pointer Language - Xpointer) a URL-ek megfelelőit írhatjuk meg XML-ben, beleértve az XML-dokumentum struktúráján belüli elérési utakat. Együttes használatukkal egyik dokumentumból rögtön egy másik közepébe linkelhetünk, egészen az egyéni tag-ekig.

XML-függvények

Ez a rész az **xml_** előtagú PHP-függvényeket öleli fel.

Telepítés

Használd az Apache 1.13. 9-et vagy későbbi változatát, mert az Apache tartalmazza az expat-könyvtárat **(www.jclark.com/xml/)** a PHP-hez.

A Unix-rendszerekhez a **--with-xml** használatával állítsd össze a PHP-t. Ha nincs telepítve az expat vagy egy expat-et tartalmazó Apache verzió, menj a **www.jclark.com/xml/** oldalra, installáld az expat-et majd, állítsd össze a PHP-t.

A Windows 98-hoz, a Windows NT-hez és a Windows 2000-hez a Win32 bináris beépített XML-támogatással rendelkezik.

Függvények

Az XML-adatokhoz **xml_** előtagú függvények sorozatos alkalmazásával jutunk. A folyamat hasonló ahhoz, mint mikor egy fájlt adatról adatra olvasunk, ami megfelelő kontrollt ad a feldolgozás felett, biztosítja, hogy a kódod minden adatot lát, és segítségével kis ráfordítás-

sal hatalmas fájlokat tudsz beolvasni. Az elkülönített domxml_-függvények faszerkezetben jutnak az XML-adatokhoz, de ehhez az összes XML-adatnak a memóriában kell lennie.

Az xml_-függvények használatakor az adatok feldolgozására egy értelmezőt hozunk létre, majd átfuttatjuk rajta az adatokat a xml_parser()-en keresztül.

xml_parser()_create

Az **xml_parser()_create** paraméterként egy szabadon választott karakterkészletet fogad el és egy értelmezőazonosítót ad vissza, amely más XML-függvényekhez szükséges. Ha a szabadon választható karakterkikódoló paramétert használod, az elfogadott értékek az ISO-8859-1, az US-ASCII, és a UTF-8. Az ISO-8859-1 az alapértelmezés. A példa a következő:

```
if(!$parser = xml_parser_create())
    print("<br>parser create failed!");
```

xml_parser_free()

Az **xml_parser_free()** az értelmező eltávolításával forrásokat szabadít fel. Az értelmező eltűnik a szöveg végén, ezért ennek a függvénynek akkor vehetjük legnagyobb hasznát, ha az XML-értelmező utolsó használata után a szövegben sok erőforrás-igényes feldolgozás szerepel:

```
if(!xml_parser_free($parser))
    print ("<br>Parser not free!");
```

xml_set_object()

Az **xml_set_object()** XML-függvények objektumokban való használatakor **alkalmazzuk.** Az **xml_set_object ()** az **xml_parser_create** ()-től fogad el egy értelmezőazonosítót és egy utalást az objektum nevére, ahogy azt a következő példa is mutatja. Ezt a függvényt használd az xml_parser_create() után és minden kezelőbeállító függvény előtt:

```
xml_set_object(Sparser, &$this);
```

xml_set_element_handler()

Az **xml_set_element_handler()** paramétere egy értelmezőazonosító, az induló elemkezelő neve és a záró elemkezelő neve. A függvény regisztrálja az elemkezelőket az értelmezőben, és siker esetén igazzal, hiba esetén hamissal jelez vissza. A kezdő elemkezelőnek el kell fogadnia az értelmezőazonosítót, a tag-nevet és a tag attribútumainak listáját tartalmazó tömböt (ha van ilyen). Az XML-ben próbáld minimalizálni a tag-ek használatát. Számos DTD az adattároláshoz olyan attribútumokat használ, amelynek child tag-ben kellene lennie. A záróelemkezelőnek el kell fogadnia az értelmező-azonosítót és a tag-nevet:

```
function start_element_handler($parser, $tag, $attributes)
{
>
function end_element_handler($parser, $tag)

if(!xml_set__element_handler($parser, "start element_handler",
    "end element handler"))
```

```
print ("<br>xml_set_element_handler
failed!"); }
```

xmí_set_character_data_handler()

Az xml_set_character_data_handler() paramétere egy értelmezőazonosító és a karakteradat-kezelő neve. A függvény regisztrálja a kezelőt, és siker esetén igazat, hiba esetén hamisat ad vissza. A karakteradat-kezelőnek el kell fogadnia az értelmezőazonosítót és a karakteradatokat:

```
function character_data_handler($parser, $data)


if(!xml_set_character_data_handler($parser, "character_data_handler") {
   print("<br>xml_set_character_data_handler failed!");
```

xml_set_default_handler()

Az xml_set_default_handler() paramétere egy értelmezőazonosító és az alapértelmezett kezelő neve. Regisztrálja a kezelőt, és siker esetén igazat, hiba esetén hamisat ad vissza. Az alapértelmezett kezelőnek el kell fogadnia az értelmezőazonosítót és az adatokat, és valamilyen műveletet **kell** végeznie. Kezdésnek azt ajánlom, ömlesszük be az adatokat kiíratásra:

```
function default_handler($parser, $data)


if(!xml_set_default_handler($parser,     "default_handler"))
    {
   print("<br>xml_set_default_handler failed!");
    }
```

xml_set_external_entity_ref_handler ()

Az **xml_set_external_entity_ref_handler()** paramétere egy értelmezőazonosító és a külső elemhivatkozás kezelőjének neve. Regisztrálja a kezelőt, és siker esetén igazat, hiba esetén hamisat ad vissza. A külső elemkezelő megkapja az értelmezőazonosítót, egy sztringet, amely szóközökkel elválasztott elemnevek felsorolását tartalmazza, egy base nevű sztringet, amely jelenleg üres, a rendszerazonosítót és a nyilvános azonosítót. Azt javaslom, hagyd ki ezt a függvényt addig, míg a többi XML-dolog nem működik, és bízd ezt a tag-et az alapértelmezett kezelőre:

```
function external_entity_ref_handler($parser, $entity_names, $base,
   $system_id, $public_id)


if(!xml set_external_entity_ref_handler($parser,
   "external_entity_ref_handler") ) {
   print ("<br>xml_set__external_entity ref handler failed!"); }
```

xml_set_notation_decl_handler ()

Az xml__set_notation_decl_handler () paramétere egy értelmezőazonosító és a jelöléskezelő neve. Regisztrálja a kezelőt, és siker esetén igazat, hiba esetén hamisat ad

vissza. A jelöléskezelőnek jelölésdeklarációval kapcsolatos információkat kell elfogadnia < **[NOTATION name {systemld | publicld}>** formátumban. A jelöléskezelő megkapja a jelölés nevét, egy base nevű sztringet, amely most üres, a rendszerazonosítót és a nyilvános azonosítót. Azt javaslom, hagyd ki ezt a függvényt addig, míg a többi XML-dolog nem működik, és bízd ezt a tag-et az alapértelmezett kezelőre:

```
function notation_handler($parser, $notation_name, $base,
   $system_id,
   $public_id)
   {
   }
if(!xml_set_notation_decl_handler($parser, "notation_handler"))
   print ("<br>xml set notation deci handler failed!");
```

xml_set-processing_instruction_handler()

Az **xml_set-processing_instruction_handler()** paramétere egy értelmezőazonosító és a feldolgozó utasításkezelő neve. Regisztrálja a kezelőt, és siker esetén igazat, hiba esetén hamisat ad vissza. A feldolgozó utasításkezelőnek el kell fogadnia az értelmezőazonosítót, egy célnevet és a cél adatait. A feldolgozó utasítások <? target data ?>-formában szerepelnek. Azt javaslom, hagyd ki ezt a függvényt addig, míg a többi XML dolog nem működik, és bízd ezt a tag-et az alapértelmezett kezelőre:

```
function processing_instruction handler($parser, $target, $data>

   }
if(!xml_set_processing_instruction_handler($parser,
   "processing_instruction_handler") ) {
   print("<br>xml_set_processing_instruction_handler failed!"); }
```

xml_set_unparsed_entity_decl_handler()

Az xml_set_unparsed_entity_decl_handler() paramétere egy értelmezőazonosító és a nem értelmezett elemdeklaráció kezelőjének neve. Regisztrálja a kezelőt, és siker esetén igazat, hiba esetén hamisat ad vissza. A nem értelmezett elemdeklaráció kezelőjének NDATA-pa-raméterrel kell elfogadnia a külső elemek deklarációjával kapcsolatos információkat. A külső elemdeklarációk formátuma **<!ENTITY name {publicld | systemld} NDATA notationName>.** A kezelő megkapja az elem nevét, egy base nevű sztringet, amely jelenleg üres, a rendszerazonosítót és a nyilvános azonosítót. Azt javaslom, hagyd ki ezt a függvényt addig, míg a többi XML-dolog nem működik, és bízd ezt a tag-et is az alapértelmezett kezelőre:

```
function unparsed_entity_decl handler($parser, $entity_name, $base,
   $system_id, $public_id, $notation_name)


if(!xml_set_unparsed_entity_decl handler($parser,
   "unparsed_entity_decl_handler"))
```

```
print("<br>xml set unparsed__entity_decl_handler failed!");
-}
```

xml_parse()

Az **xml_parse()** paramétere egy értelmezőazonosító, egy feldolgozandó adatokból álló sztring és egy szabadon választható igaz érték, amely azt jelöli, hogy ez a sztring az utolsó feldolgozandó sztring. A függvény igazzal jelez vissza, ha minden működik, és hamissal, ha az értelmezőazonosító nem tud érvényes értelmezőt azonosítani, vagy ha az értelmező nem működik:

```
if(!xml parse ($parser, Sdata) )
{
print("<br>xml_parse failed with " .
htmlentities($data)); }
```

xmljget_error_code()

Az **xml_get_error_code()** paramétere egy értelmezőazonosító, és vagy a legutolsó hiba hibakódját adja vissza, vagy pedig egy hamis értéket, ha az értelmezőazonosító érvénytelen. A következő kód a hibakódot és a switchQ-függvényt használja a működésbe lépéshez, ami ebben a rövid példában csak egy üzenet kiírása. Figyelj arra, hogy nem kell megjegyezned a hibák számát, mert a PHP XML támogatásában szerepel egy a hibakódok nevét tartalmazó lista:

```
switch(xml_get_error_code($parser))
{
case XML_ERROR_NONE:
""
break;
case XML_ERROR_NO_MEMORY:
print("<br>Out of raemory.") ;
break;
default:
print ("<br>Oops, unknown error!");
)
```

Itt következnek a hibakódokra definiált nevek az **xml_parse()-ból:**

```
XML_ERROR_NONE
XML_ERROR_NO_MEMORY
XML_E RROR_S YNTAX
XML_ERROR_NO_ELEMENTS
XML_E RROR_INVALID_TOKEN
XML_ERROR_UNCLOSED_TOKEN
XML_ERROR_PARTIAL_CHAR
XML_ERROR_TAG_MISMATCH
XML_ERROR_DUPLICATE_ATTRIBUTE
XML_ERROR_JUNK_AFTER_D0C_ELEMENT
XML_ERROR_PARAM_ENTITY_REF
XML_ERROR_UNDEFINED_ENTITY
XML_ERROR_RECURSIVE_ENTITY_REF
XML_ERROR_ASYNC_ENTITY
XML_ERROR_BAD_CHAR_REF
XML_ERROR_BINARY_ENTITY_REF
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
XML ERROR MISPLACED XML Pl
```

```
XML_ERROR_UNKNOWN_ENCODING                '-    "• •                ' ! ) 1 i
XML_ERROR_INCORRECT_ENCODING
XML_ERROR_UNCLOSED_CDATA_SECTION                  jJLi|, -juiTi^a--- /-ogiiq  '
XML_ERROR_EXTERNAL_ENTITY_HANDLING                •   :'          '   ' .'  ^
```

E hibák legtöbbje magától értetődő; dupla attribútum például akkor fordul elő, mikor egy adott attribútum kétszer szerepel egy tag-ben. Ha olyan hibával találkozol, amely nem egyértelmű, írasd ki az utolsó sztringet, amely az xml_parse()-hoz van, és ne felejtsd el beletenni egy **htmlentities()-be.**

xml_error_string()

Az **xml_error_string()** paramétere egy hibakód, eredménye a hibát leíró sztring. A következő kód egy XML-hibából származó olvasható üzenetet ír ki:

```
print ("<br>"    .    xml_error_string(xml    get_error_code($parser)));    ^
```

xml_get_currentjine_number()

Az **xml_get_current_line_number()** paramétere egy értelmezőazonosító, eredménye az **xml_parser()-en** éppen futó sor száma. A következő kód a sor számát írja ki. írasd ki ezt a számot, amikor egy XML-inputfájl hibáit jeleníted meg, így könnyen végigmehetsz a fájlon, és megtalálhatod a megfelelő sort:

```
print ("<br>Current    line    number:    "
    .    xml_get_current_line_number($parser) ) ;
```

xml_get_current_coloumn_number()

Az **xml_get_current_coloumn_number()** paramétere egy értelmezőazonosító, eredménye pedig az **xml_parser()-ben** éppen futó sor oszlopának száma. Az alábbi kód a sor és az oszlopszámot írja ki. Jelenítsd meg ezeket is az XML-inputfájl hibáival együtt:

```
print("<br>Current    line    number:    "
    .    xml_get_current_line_number(Sparser) .    "    and    column
    number:    " .    xml_get_current_column    nuraber(Sparser));
```

xml_get_current_byte_index()

Az **xml_get_current_byte_index()** paramétere egy értelmezőazonosító, visszatérési értéke pedig az értelmező input pufferében levő aktuális bájt száma. A következő kód a bájtindexet írja ki, és csak akkor van haszna, ha az XML-adatok kiírásakor az inputadatpuffert is kiíratod. (Ne felejtsd el használni a **htmlentities()-t** az XML tag-ek láthatóvá tételéhez):

```
print("<br>Current byte index: " .
xml_get_current_byte_index($parser)) ;
```

xml_parse_into_struct()

Az **xml_parse_into_struct()** paramétere egy értelmezőazonosító, egy XML-adatokból álló sztring és két tömb az XML-adatokból dekódolt adatstruktúrák fogadására. Mindkét tömb elé ki kell tenni az &-jelet, hogy hivatkozással jusson tovább; ez a technika ritkulóban van, így a függvényt talán megváltoztatják, hogy a referenciával érkező tömböket őt nélkül is elfogadja. Az egyik tömb az adatokat fogadja, a másik pedig az adatok indexe lesz:

```
if(!xml_parse_into_struct($parser, $data, &$values, &$index)
    {
   print("<br>Error   during   parse.");
```

Ehhez a függvényhez az összes adatnak egyszerre a memóriában kell lennie, ezért nem alkalmas nagy XML-fájlokhoz. Ha csak kevés adatot szeretnél kinyerni egy nagyobb XML-fájlból, majd azt egy szerkezet segítségével feldogozni, használhatod a kód módosított változatát, amit a Gyors megoldások részben "Az XML-adatok értelmezése" cím alatt találsz. Ezzel beolvashatsz egy nagy fájlt és részhalmazokat képezhetsz belőle, amit aztán továbbadhatsz az xml_parse_into_struct()-függvénynek.

xml_parser_set_option()

Az xml_parser_set_option() paramétere egy értelmezőazonosító, egy értelmező opció neve, és az opcióra vonatkozó ú) érték. Visszatérési értéke siker esetén igaz, ha pedig az értéket nem lehet beállítani, akkor hamis. A függvény segítségével beállíthatsz néhány itt is említett XML-feldolgozó opciót, de ahogy az XML-feldolgozó kód fejlődik, a lehetőségek száma is egyre bővül. Az **XML_OPTION_CASE_FOLDING** ellenőrzi, hogyan kezelik az XML-függvények a tag nevek kis- és nagybetűit. Az XML-függvények alapértelmezésben rendelkeznek azzal az intelligens opcióval, hogy egyféle betűre fordítanak minden tag nevet, így nem származhatnak hibák a tag-ek elgépeléséből, vagyis nem veszítünk el egy tagét, csak **mert** az **inputban** "headingOne"-nak írt tag szerepel "headingone" helyett. Sajnos az XML-ben a kis- és nagybetűk miatt előfordulhatnak ugyanolyan nevek, az XHTML-ben konkrétan két mező is szerepel **Inline** és **inline** névvel. Hogy az ilyen klasszikus hibával „ellátott" XML-fájlt is fel tudjuk dolgozni, kapcsoljuk ki a betűk váltogatását, így:

```
if(!xml_parser_set_option(Sparser,    XML_OPTION_CASE_FOLDING,     0))

    print ("<br>Error    setting
    option.");                                          tai'-fz .<'"•'. x
    }                                 ...          .;
```

Az XML_OPTION_TARGET_ENCODING segítségével a céladat bekódolását tartalmazó sztringet határozhatjuk meg; a jelenlegi opciók az ISO-8859-1, az US-ASCII és az UTF-8. Az alapértelmezés az, hogy a forrás bekódolást használjuk a cél bekódoláshoz. Egy új XML-forrás tesztelésekor azt akarjuk, hogy az output közvetlenül kapcsolódjon az inputhoz. Tesztelj enélkül, állítsd működésbe a tag-et és minden mást, majd ezután kísérletezz a bekódolás változtatásával.

Az **XML_OPTION_SKIP_WHITE-tal** kihagyhatod az üres karaktereket. Ezt nem teszteltem le, mert az alapértelmezés a legtöbb fájl esetében jól működik. Ezenkívül, ha valóban ki is kell hagynod az üres karaktereket, tesztelés alatt gyakran el kell fogadnod őket, vagyis akkor megállapíthatod, hogy a beérkező fájl megfelelően formázott-e. .

xml_parser_get_option()

Az **xml_parser_get_option()** visszaállítja egy XML-opció aktuális beállítását. A lehetséges opciók listáját keresd az **xml_parser_set-option()-nál.** Az alábbi kód egy opció aktuális beállítását írja ki:

Sf; *rhnty* jiiv'ri \

```
print("<br> XML_OPTION_TARGET_ENCODING: "
    . xml_parser_get_option($parser, XML_OPTION_TARGET_ENCODING));
```

utf8_encode()

Az utf8_encode() az ISO-8859-1 sztring adatait kódolja UTF-8 formátumúra, hogy a 16 bites Unicode-adatokat 8 bitesekkel keverhesd. Az UTF-8 leírását az RFC 2279-ben találod. (www.faqs.org/rfcs/rfc2279.html) Unicode-adatokat vehetünk XML tag-ekbe az XML dekódoló felborítása nélkül, és az adatokat az **utf8_decode()** használatával 16 bites Unocode-adattá alakíthatjuk vissza. Az alábbi példa bekódol egy sztringet, majd XML tageket vesz a sztring köré, ezzel jelezve, hogy az adatok UTF-8 kódoltak. Mikor az adatokat az XML függvények ellenőrzik, a záró elemkezelő felismeri a **<utf8>** tag-et, és dekódolja a sztringet:

```
$data .= "<utf8>" . utf8_encode($string) . "</utf8>";
```

utf8_decode()

Az uf8_decode() az **utf8_encode()** párja, a 8 bites sztringet kódolja 16 bites Unicode-dá:

```
$data = utf8_decode($utf8_data);
```

IMiA if 9fi 15

XSLT

Az eXtensible Stylesheet Language Transformation (Kiterjeszthető Stíluslap Nyelv Átalakítás) programok a kísérleti PHP XSLT-függvények használatával lehetségesek (kísérleti a PHP4-ben és még mindig kísérleti a 4.0.6-ban). Szükséged lesz egy jó könyvre az XML stíluslapokról (XSL, XML Style Sheet)) és az XLS transzformációs nyelvről (XSLT). A PHP XSLT-függvények a Sablotron-szoftver segítségével töltik az adatokat az XML és XSL fájlokba, az eredmény pedig egy újabb XML-fájl, egy XHTML- vagy sima HTML-fájl - az XSL fájlban lévő utasításoktól függően.

HTML, DHTML vagy XSLT? % \

Az XSL egy XML alapú stíluslap nyelv, amely néhány új böngészőben már elérhető. Az XSLT egy XML-dokumentum másik XML-dokumemummá alakításának a nyelve, segítségével az XML-adatok XHTML-adatokká alakíthatók. Az XHTML az XML DTD-kben definiált HTML 4 verziója, és néhány újabb böngészőben használható. Együttes segítségükkel az XML-adatokat weboldalként jelenítheted meg. II-

A HTML 4 a szabvány HTML a jelenleg használatos legtöbb böngészőben. Nagyszerűen alkalmas a statikus adatokhoz, de nem teszi lehetővé dinamikus adatok használatát. A HTML PHP-vel való kibővítésével a szerveren is lehet dinamikus adatokat beilleszteni HTML-be. A PHP-vel és a HTML-lel a HTML-t úgy formázhatod, hogy bármilyen böngészőnek megfeleljen, és akármilyen adatok megjelenítésére képes legyen. A különféle XML-technológiák segítségével a szerveren vagy a böngészőn lehet elvégezni a beillesztést és a formázást, de az adatok XML-formában kell hogy szerepeljenek, és ha böngésző alapú eszközöket használsz, akkor a legújabb böngészőre lesz szükséged. $$

A dinamikus HTML (DHTML) a HTML dinamikus új nyelvvé való alakítására irányuló próbálkozás a JavaScript hozzáadásával, ám a JavaScript egy szerényen definiált nyelv, korlátozott kapacitásokkal, gyenge böngésző'támogatással, és behatárolt a szerver alapú adatokhoz való hozzáférése. PHP kell hozzá, hogy minden böngészőben kialakítsuk a megfelelő JavaScript-et, ezért sokkal könnyebb elhagyni a JavaScript-et, és csak a PHP használatával írni meg az egész oldalt. A JavaScript nem képes a szerveren található adatbázis olvasására, a PHP szükséges az adatok JavaScript-be történő betöltésére, és a PHP tudja a teljesen kiépített HTML-be betölteni az adatokat. Szerintem a DHTML-ből a D a Doh!-t jelenti.

A HTML és a PHP együtt nagyszerű kombinációt alkotnak. A PHP és a szerveren találha tó XML szintén. Minden olyan technológia, ami a böngésző intelligenciáján alapszik, kiszá míthatatlan, és a potenciális látogatók körét is korlátozza. A transzformációkat és a formá zást a szerveren végezd. ,

Az XSLT telepítése

Az XSLT használja a Sablorton-t, a Sablotron pedig használja az expat könyvtárat, de nem veszi be az XML-függvények által használt verziót.

A Sablotron Unix alá történő telepítéséhez kövesd a következő lépéseket:

1. Tedd fel a legújabb Sablorton-t és expat-et a www.gingerall.com/-rol.
2. Fordítsd be PHP-t a -with-sablot-tal.

A Sablotron Windowsra vagy Windows NT-re való installálásához ezeket a lépéseket kövesd:

1. Állítsd le az Apache-ot vagy a web-szolgáltatót.
2. Töröld ki a pontosvesszőt (;) az itt következő sor elejéről a php.ini-ben.

```
extension=php_sablot.dll
```

3. Másold a phb_salbot.dll-t a c:/Program Files/php/extensions-ból a c:/windows/system-be (vagy a c:/winnt/system32-be a Windows NT-ben és a Win dows 2000-ben).
4. Másold a phb_salbot.dll-t és az expat.dll-t a c:/Program Files/php/dlls-ből a c:/windows/system-be (vagy a c:/winnt/system32-be a Windows NT-ben és a Win dows 2000-ben).
5. Indítsd újra az Apache-ot.

XSLT-függvények -..^HHA*!-.

Az XSLT-feldolgozás az xslt_create()-tel kezdődik, amely létrehoz egy XSLT-folyamatot, majd az xslt_run() következik az XSLT-fájl feldolgozásához, végül az xslt_fetch_result() az eredmények visszanyeréséhez. Alternatív függvénykombinációkkal a feldolgozás változtatható, beleértve fájlok csoportjainak egy ütemben történő feldolgozását is.

xslt_create()

Az xslt_create() egy új XSLT-forrásazonosítót hoz létre, amely minden más XSLT függvényben használható. A következő kód a $x-et állítja fel mint XSLT-forrást. A $x az első paraméter a következő' példákban:

```
$x = xslt createO;
```

xslt_openlog()

Az XSLTfeldolgozás a hibákat egy általad választott naplóba rakja, amit az **xslt_openlog()-gal** nyithatsz meg. Az **xslt_openlog()** paramétere egy XSLT-forrásazonosító, egy fájlnév és egy egész számmal megadott naplózási szint. A forrásazonosító megadása azt jelenti, hogy minden XSLT-folyamatnak lehet külön naplóállománya. Mikor ezeket a sorokat írom, a naplózási szint még nincs dokumentálva, szóval kísérletezz vele bátran. A feldolgozás ellenőrzéséhez a saját kódodon belül fellelhető hibákra is kell tesztet futtatnod, így már azelőtt tudni fogod a hibákat, hogy a naplót megnéznéd, ráadásul a napló tartalmát összepárosíthatod az **xslt_error()** segítségével visszajelzett hibákkal:

```
if (!xslt_openlog($x, "t:/xslt.txt", 1))
    print("<br>xslt_openlog failed.");
```

xslt_errno()

Az **xslt_errno()** visszaküldi az utolsó XSLT-hiba hibaszámát. Ha ellátod XSLT-forrásazo-nosítóval, az a hiba az XSLT-feldolgozás utolsó hibája. Ha kihagyod a forrásazonosítót, a hiba az összes folyamat hibái közül az utolsó. A következő kód a legfrissebb hiba számát írja ki a $x-ből:

```
print("<br>xslt error " . xslt_errno($x)) ;
```

Az alábbi jelenik meg, ha nincs hiba:

```
xslt error 0
```

xslt_error()

Az **xslt error()** az utolsó XSLT-hiba hibaszövegét küldi vissza. Ha XSLT-forrásazonosító-val látod el, a visszaadott hiba az adott XSLT-folyamat utolsó hibája lesz. A következő kód a $x XSLT folyamat legutóbbi hibájának szövegét írja ki:

```
print("<br>xslt error " . xslt_error($x)) ;
```

Ezt az üzenetet kapod, ha nincs hiba:

```
xslt error OK
```

xslt_run()

Az xslt run() paramétere egy XSLT-forrásazonosító, egy input XSL- és XML-fájl neve URL-ben, egy eredménypuffer opcionális neve, és egy XSLT-paraméterekből, valamint egy XSLT-argumentumokból álló szabadon választott tömb. Az output a pufferbe érkezik, és

az xslt_fetch_result()-tal nyerhető vissza. A puffer neve alapértelmezésben **"/jresult", az xslt_run()-ban** és **az xslt_fetch_result()-ban** is. A következő kód lefuttatja az **xslt_run()-t,** keresi a hibákat, majd visszahozza és megjeleníti az eredményt egy HTML-oldalként:

```
if(xslt_run($x, "./test.xsl", "./test.xml"))

   print("<br>" . xslt_fetch_result($x));

else

   print("<br>xslt_failed failed.");
```

xslt_fetch_result()

Az **xslt_fetch_result()** az a függvény, amellyel az **xslt_run()** által talált eredmények visszahozhatok. Az alábbi kód visszahozza a találatokat, és ahelyett, hogy **weboldalként jelenítené meg, lefordítatlan** HTML-ként hozza fel, így az összes tag **látható az eredményeken belül:**

```
$string = xslt_fetch_result($x) ;
print ("<br>XSLT data: " . htmlentities ($string) ) ; .,
```

xslt_output_begintransform()

Az xslt_output_begintransform() segítségével egy stíluslapon egy egész sor átalakítást **elvégezhetsz. Add** meg a **stíluslapot** az **xslt_output_begintransform()-ban,** és minden transzformáció azt a stíluslapot használja, míg le nem futtatod az **xslt_output_endtransform()-ot:**

```
xslt_output_begintransform("./test.xsl");
```

xslt_output_endtransform()

Az **xslt_output_endtransform()** befejezi az aktuális, az **xslt_output_begintransform()-ban** megadott **XSLT** fájl használatát:

```
xslt output endtransform()
```

xslt_set_sax_handler()

Ha az XSL-ró'l és az XSLT-ről olvasol a W3 weboldalon, vagy máshol, találkozhatsz a Simple API for XML (SAX)-ra történő utalásokkal - a Sablotron megjelenése előtt ez szolgált az XML-dokumentumok átalakításra. A SAX-kezelők úgy használhatók az XSLT-fel-dolgozásban, hogy felállítunk egy a SAX-kezelőket tartalmazó tömböt, és azt mmt a **xslt_set_sax_handler()** második paraméterét alkalmazzuk. A következő kód feltételezi, hogy a SAX-tömböt már felállítottuk, lefuttatja az **xslt_set_sax_handler()-t,** majd ellenőrzi a hibákat, és hiba esetén üzenetet küld. (Én még nem láttam a SAX-ot az újabb XSLT-függ-vényeket és a Sablotron-t alkalmazó site-oknál, szóval lehet, hogy sosem lesz szükséged erre a függvényre):

```
if (!xslt_set_sax_handler($x, $array) ) {
    print("<br>xslt_set_sax_handler    failed.");
```

xsltJransformQ

Az xslt_transform() egy egyszeri átalakítást hajt végre, és engedi, hogy minden szükséges fejlettebb Sablotron-paramétert megadj. Fogd a Sablotron-hoz járó dokumentációt, és olvasd el a paraméterekről szóló részt. Az alábbi kód az xslt_transform()-ot mutatja hibakeresésben:

```
if (!xslt_transform(" ./test .xsl", " ./test.xml", $result, $parameters, /*™|
    $arguments, $result_buffer) )                                    «.;»-^VJ
    {
    print("<br>xslt_transform failed.");                                  " '
```

xslt_process()

Az **xslt_process()** az **xslt_transform()** egyszerűsített változata, ami az input- és output-sztringeket is elfogadja. Nincsenek paraméter vagy argumentum opciók. Akkor válaszd, ha az adataid már egy sztringben vannak, és az XSL-fájl extra paraméterek nélkül is el tudja végezni az átalakítást:

```
if (!xslt_process ($xsl, $xml, $result) )                  rí ,U"iS£J     i
    {
    print ("<br>xslt_process error " . xslt_errno() . " " . xslt_error());
```

xslt_closelog()

Az xslt_closelog() egy meghatározott XSLT-folyamat idejére bezárja a log-ot, ahogy azt a következő kód is mutatja. Azután megnézheted a log-ot, hogy ellenőrizd, működött-e a transzformáció:

```
if(!xslt_closelog($x))
    {
    print("<br>xslt_closelog failed.");
```

xslt_free()

Szabadítsd fel az összes forrást egy adott XSLT-folyamatra, ahogy itt látható:

```
xslt    free ();
```

WDDX

A Web Distributed Data Exchange az OpenWDDX-ből **(www.openwddx.org)** lehetővé te szi, hogy a programozási információkat, például változókat sztringbe írva alkalmassá te gyük e-mailes vagy más karakter alapú továbbításra. További magyarázatokért és példákért olvasd el a 16. fejezetet. . r j :v• j

A WDDX elősegíti a program-program kommunikációt, ha a forrás- és a célprogramok szorosan kapcsolódnak. Nem ajánlom, hogy a WDDX-et PHP-szövegben található értékek JavaScript-re vagy valami hasonló böngésző alapú technológiára való átalakítására használjuk. A WDDX korlátozott adatdefiniáló lehetőséggel rendelkezik, és előfordulhat, hogy

olyan adatot kreál, amit a cél rosszul értelmez. Még ha a formátum értelmezése megfelelő lesz is, a jelentés értelmetlen lehet, ha a forrás és a cél nincsenek szinkronban. A WDDX alkalmas a saját szervereid közötti adatáramlásra, de nem alkalmas a te szervered és idegen szerverek között zajlóra.

DOM

A Document Object Model (DOM) egy teljes dokumentum felépítéséhez vagy egyes részeihez ad hozzáférést, olyan félig random üzemmódban, mint amikor JavaScript-et használva egy böngészőben végigmegyünk egy weboldalon, egy Visual Basic alkalmazással egy Word-dokumentumon vagy egy LDAP könyvtáron. Ismerned kell a szerkezet felső részeit, hogy az alsóbb részekbe is bejuthass, és a bejutáshoz a hozzáférés minden alkotóelemét ismerned kell. A DOM XML-függvényeket a domxml-elő'tag jelöli.

Mivel a DOM-nál az egész dokumentum előbb be kell, hogy kerüljön a memóriába, inkább kisebb dokumentumok vagy nagy szerverek esetén megfelelő. Ha egy nagy XML-fájlt szeretnél pásztázni, használd a hagyományos XML-függvényeket.

Mivel egy szerkezeten kell végigmenned, előre ismerned kell az adott szerkezetet, vagy legalább olyan függvényeket, amelyekkel az felfedhető. A DOM XML-függvények között találhatók olyanok, amelyek felismerik egy dokumentum szerkezetét, vagy új struktúrát építenek. Egy új szerkezet építése hasonlít egy többszintes tömb kialakításához, így egy XML-fájl építése a DOM XML-függvények segítségével olyan, mint egy komplex tömb létrehozása WDDX-szel. A végén ugyanúgy egy XML-sztring marad, amit el kell küldened.

Mikor használd a DOM XML-t? Ha a forrás- vagy céldokumentum valami AbiWord-szerű szövegszerkesztő dokumentum, vagy XSL típusú stíluslap, használd a DOM strukturált megközelítését. Ha a forrás vagy cél egy adatbázistábla, felejtsd el a DOM faszerkezetét, és alkalmazd a hagyományos XML-függvények szekvenciális megközelítését.

A PHP DOM-függvények egyelőre csak kísérletiek, ezért könnyen meghiúsíthatják egy megbízható weboldal kialakítására tett próbálkozásaidat. Ne számíts rá, hogy a DOM XML megbízhatóvá válik még jelenlegi és jövőbeli projektjeid élete alatt. A DOM XML-kísérlet pusztán hencegésre jó, valamint arra hogy az olyan munkaerő-közvetítő cégeknek örömet okozzon, amelyek azonnal ötéves DOM XML-tapasztalatot várnak el.

A DOM XML telepítése

A DOM XML Unix-ra történő telepítéséhez kövesd a következő lépéseket:

1. Telepítsd fel a legújabb GNOME XML könyvtárat a www.xmlsoft.org-ról.
2. Fordítsd be a PHP-t a --with-dom=[DIR]-rel.

A DOM XML Windows-ra vagy Windows NT-re való telepítéséhez kövesd az alábbi lépéseket:

1. Állítsd le az Apache-ot vagy a webszervert.

2. Töröld ki a pontosvesszőt (;) a következő sor elejéről php.ini-ben:
   ```
   extension=php_domxml.dli
   ```
3. Másold a php_domxml.dll-t a c:/Program Files/php/extensions-ből a c:/windows/system-be (vagy a Windows NT-nél és a Windows 2000-nél a c:/winnt/system32-be).
4. Másold a Libxml2.dll-t **a** c:/Program Files/php/dlls-ből a c:/windows/system-be (vagy a Windows NT-nél és a Windows 2000-nél a c:/winnt/system32-be).
5. Indítsd újra az Apache-ot.

DOM XML-függvények

Használd az xmldoc()-ot egy új dokumentumobjektum létrehozására, az **xmldocfile()-t** pedig egy meglevő fájl objektumkénti beolvasására. Az összes többi függvény működik az **xmldoc()-ból** vagy az **xmldocfile()-ból** vett objektummal.

xmldocQ

Az **xmldoc()** egy DOM XML objektumot hoz létre, és minden más DOM XML-függvény kiindulópontját jelenti egy új dokumentum létrehozásakor vagy egy már sztringbe illesztett fájl használatakor. Az alábbi példa egy egyszerű szövegsztringből hoz létre objektumot, amelyet ki is írat:

```
Sobject = xmldoc("<doc><title>Test</title><text>The cholesterol laden"
    . " cow tried to beat NASA to the moon.
</textx/doc>"); print_r($object);
```

íme az objektum kiíratása (figyelj oda, hogy az osztály **DomDocument):**

```
DomDocument Object ( [name] => [url] => [version] => 1.0 ..
    [standalone] => -1 [type] => 9 [compression] => -1
    [charset] => 1 [0] => 2 [1] => 74523584 )
```

xmldocfileQ

Az **xmldocfile()** egy DOM XML objektumot hoz létre, és az XML-adatok közvetlen fájl ból való beolvasásakor ez lesz minden további DOM XML-függvény kiindulópontja. A következő példa az AbiWord-dokumentumból hoz létre egy objektumot (AbiWord-dokumentumot más példákban használtunk). A fájl csak olvasható, vagyis az objektumban esz közölt változtatások nem jelennek meg a fájlban. A függvényhez kell a fájlhoz való teljes hozzáférés, ezért a **realpath()-t** a vonatkozó fájlnév veszi körül. Ez lehet, hogy nem működik, ha a dokumentumod mást használ, mint az alap XML típusú deklaráció, ezért kétszer is ellenőrizzük a fájl karakterbeállítási paramétereit a fájl **<?xml version="l.Q">** **tag-jeben:**

```
$file object = xmldocfile(realpath("./fishcurry.abw") ) ;
printr($file_object);
```

íme az objektum kiíratása:

```
DomDocument Object ( [doc] => Resource id #3 [url] =>
i:\usr\home\petermoulding\web\root\phpblackbook\xml\fishcurry.abw
    [version] => 1.0 [standalone] => -1 [type] => 9 [compression] => -1
    [charset] => 1 )
```

Vegyük észre az input-fájl forrásazonosítóját és a URL-t, annak ellenére, hogy az **xmldocfile()** jelenleg csak olvasási hozzáférést biztosít. Az URL tartalmazza a fájl teljes, a szerverhez viszonyított relatív elérési útját. Arra is figyelj, hogy az objektum csak a fájl gyökércsomópontját tartalmazza, nem a teljes faszerkezetet. Ez azt jelentheti, hogy a függvény csak azokat a részeket olvassa be a memóriába, amelyeket te elérsz, ezek szerint te egy nagy fájl részeihez is hozzáférhetsz anélkül, hogy telítenéd a memóriát.

xmltreeQ

Az **xmltree()** egy elkülönített függvény, amely DOM XML-objektumokból létrehoz egy szerkezetet, és XML-alapú dokumentumok tartalmának megtekintésére szolgál. Az alábbi példa egy egyszerű szöveges sztringből hoz létre ob)ektumszerkezetet, majd a print_r-rel kiíratja:

```
$tree_object = xmltree("<doc><title>Test</title><text>The cholesterol
laden"
    . " cow tried to beat NASA to the moon.
</textx/doc>"); print_r($tree_object);
```

Itt az eredmény, amely az inputsztring minden csomópontját tartalmazza:

```
DomDocument Object ( [name] => [url] => [version] => 1.0
    [standalone] => -1 [type] => 9 [compression] => -1 [charset] =>
    1 [0] => 4 [1] => 75945888 [children] => Array ( [0] =>
    DomElement Object ( [type] => 1
    [tagname] => doc [0] => 5 [1] => 75932784 [children] => Array
    ( [0] => DomElement Object ( [type] => 1 [tagname] => title
    [0] => 6 [1] => 75932720 [children] => Array ( [0] =>
    DomText Object ( [content] => Test [0] => 7 [1] => 75932656 )
    ) ) [1] => DomElement Object ( [type] => 1 [tagname] => text
    [0] => 8 [1] => 75945184 [children] => Array (
    [0] => DomText Object ( [content] => The cholesterol laden
    cow tried to beat NASA to the moon. [0] => 9 [1] => 75945104
    ) ) ) ) ) ) )
```

domxml__root()

A domxml_root() paramétere egy DOM XML-objektum, visszatérési értéke pedig a dokumentum gyökérelemét tartalmazó objektum. A következő kódnak működnie kellene, de az Apache-ban a PHP 4.0.6. használatakor hibát okozott. A **<?xml** version="1.0">-tag hozzáadása sem oldotta meg a problémát. A domxml_root()-nak még mindig vannak olyan hibái, amelyeket ki kell javítani, mielőtt minden szerveren megbízhatóan lehet használni:

```
$root = domxml_root($object) ;
```

Az alternatíva a következő kód használata lehet:

```
print_r($object->root());
```

Ez a verzió gyorsabb egy egyszeri kódsor esetén. Az előző kód rövidebb, ha a gyökérelemhez sok hivatkozást szeretnénk rendelni. Ennek a verziónak az az előnye, hogy működik is.

Ez a **$object->root()** outputja. Figyelj oda a **DomElement** osztályra, a **doc tagname-jére,** az első tag-re a dokumentumban:

```
DomElement Object ( [type] => 1 [tagname] => doc [0] => 3 [1]
    => > 1
    73711360 )
```

domxml_add_root()

A domxml_add_root() paramétere egy DOM XML-objektum és az elem nevét tartalmazó sztring. Eredménye egy objektum, amely a dokumentum új gyökérelemét **tartalmazza.** A következő példa az **xmldoc()** objektumához rendel új gyökérelemet, de sajnos a PHP 4.0.6 kikészül tőle:

```
$root = domxml add root($object);
```

Következik az alternatíva, amely működik:

```
$root = $object->add_root("doc");
print_r($root);
```

íme a **$root-objektum DomElement** osztállyal, **doc tagname-mel,** ez az első tag a dokumentumban:

```
DomElement Object ( [type] => 1 [tagname] => doc [0] =>
3 [1] => 8945680
```

domxml_dumpmem()

A **domxml_dumpmem{)** az aktuális DOM XML-objektumot visszaömleszti a sztringbe, vagyis az **xmldoc()** fordítottja. A következő kód egy sztringet hoz létre az aktuális DOM XML-objektumból, feltéve, hogy a függvény működik. A PHP 4.0.6 Win32 verziójában azonban nem ez a helyzet:

```
$xml_string = domxml_dumpmem($object);
```

Alább ugyanaz a művelet látható, csak a hagyományos objektummódszer jelöléssel és egy print-utasítással, hogy az eredményt is lásd:

```
$xml_string = $object->dumpmem () ;
print("<br>" . htmlentities($xml_string));
```

íme a kimenet (több százezer print-utasítás kódolása után néha még mindig elfelejtem a **htmlentities()-t,** és csodálkozom, miért hiányoznak a kimenetből a kritikus elemek):

```
<?xml version="l.0"?> <doc><title>Test</title><text>The cholesterol
    laden cow tried to beat NASA to the moon. </textx/doc>
```

domxml_attributes()

A domxml_attributes{) el kellene, hogy fogadja egy dokumentum csomópontját tartalmazó objektumot, és tömbként kellene visszaadnia a csomópont attribútumait. A következő példa a PHP 4.0.6-ban hibát okoz:

```
t<i $attributes = domxml_attributes ($root) ; while
  (üst ($k, $v) = each ($attributes) )

    print("<br>" . $k . ": " . htmlentities($v) )
```

í

domxml_ge t_a ttrib ute ()

A domxml_get_attribute() paramétere egy dokumentum csomópontját tartalmazó objektum és egy attribútum neve. Eredménye a csomópont attribútuma objektumként visszaadva. A következő példa a dokumentum gyökércsomópontjának font-attribútumát kellene, hogy visszajuttassa, és ki kellene írnia az eredményt. Az xmldoc()-példában nincsenek attribútumok, ezért ez a függvény üres objektumokat küld vissza. A PHP 4.0.6-ban sajnos nem működik:

```
$attribute = domxml_get_attribute($root, "font");
```

domxml_set_attribute()

A domxml_set_attribute() paramétere egy dokumentum csomópontját tartalmazó objektum, egy attribútum neve és értéke, és végül az attribútumot objektumként küldi vissza. A következő példa létrehozza a font-attribútumot a dokumentum gyökércsomópontjához. Sajnos a PHP 4.0.6 három paraméterrel nem bír el, és a feladatot két paraméternél befejezi:

```
$font = domxml_set_attribute($root, "font", "times román");
```

íme egy működő verzió a hagyományos objektumjelöléssel plusz egy print_r()-rel, hogy az eredményt is lásd:

```
$font = root->set_attribute("font", "times
román"); print_r($font);
```

Az eredmény a DomAttribute-osztállyal:

```
DomAttribute Object ( [name] => font [value] => times román [0]
    => 4 [1] => 74570560 )
```

domxml_children ()

A domxml_children() paramétere egy dokumentum vagy egy csomópontobjektum, és a csomópont leágazásait felsoroló tömböt küld vissza, vagyis küldene, ha működne, de a PHP 4.0.6-ban ez még nem így van:

```
Schildren = domxml children($root);
```

íme egy működő verzió a hagyományos objektum-jelöléssel plusz egy print_r()-rel az eremények kiíratása végett:

```
$children = $root->children();
print_r ($children) ;                    jt^ ji,
```

A következő eredmény a tagname-bejegyzésben szereplő elemnevet mutatja, továbbá egy két objektumból álló tömböt, amelyek mindegyike DomElement osztályba tartozik.
A $root-nak két leágazása van, title- és text-elnevezéssel, ami megfelel az xmldoc()-ba be-

adott példasztringnek: ' ' 0

```
Array ( [0] => DomElement Object ( [type] => 1 [tagname] => title
    [0] => 4 [1] => 73443472 ) [1] => DomElement Object ( [type] => 1 [tagname]
    => text [0] => 5 [1] => 73443344 ) )
```

domxml_new_child()

A domxml_new_child() paraméterként egy dokumentum csomópontját tartalmazó objektumot, a leágazás nevét és értékét kellene hogy elfogadja, majd ezt követően a leágazást egy objektumként kellene visszaadnia. Az alábbi példa létrehozza az author-attribútumot a dokumentum gyökércsomópontjához, de az eredmény egy rossz paraméterszámról szóló üzenet, a paraméterek számának megváltoztatása pedig a PHP leállását okozza:

```
$child = domxml_new_child($root, "author", "Péter"); •
```

íme egy működő verzió a hagyományos objektum-jelöléssel és egy plusz sorral a kiíratáshoz:

```
$child = $root->new_child ("author", "Péter"); **"  "  ■f,:|"'.'"
print_r($child);                                                    .
```

Az eredmény a DomElement-osztállyal és az author-tagnéweí:

```
DomElement Object ( [type] => 1 [tagname] => author [0]
    => 4 [1] => 8703120 )
```

domxml_new_xmldoc()

A domxml_new_xmldoc() az XML-dokumentumok létrehozásának másik módja. A függvény paramétere egy XML-verzió, a visszaadott eredmény pedig egy dokumentum objektum. Gyanítom, hogy ezek az egymást átfedő függvények el fognak tűnni, és a maradékhoz rendelődnek majd extra paraméterek, mint az XML-verzió:

```
$object = domxml_new_xmldoc("1. 0") ;
print_r($object) ;
```

Az eredmény a domxml_new_xmldoc()-ból:

```
DomDocument Object ( [name] => [url] => [version] => 1.0
    [standalone] => -1 [type] => 9 [compression] => -1 [charset] => 1
    [0] => 4 [1] => 74013248 )                                          .
```

xpath_new_context()

Az xpath_new_context() paramétere egy dokumentumobjektum, és működése során egy új kontextust hoz létre az xpath_eval()-hoz, ám ez a függvény még nagyon új. A PHP 4.0.4-ben jelent meg először, a 4.0.6-ban megváltozott, és úgy néz ki, ismét meg fogják változtatni. A következő példa egy kontextus objektumot hoz létre, és kiíratja az objektumot egy print_r()-rel. A függvényt nem építették be a PHP 4.0.6 Win32 bináris állományaiba, így az outputra nem tudok példát mutatni:

```
$context = xpath_new_context($object) ;
print_r ($context) ;                    „.>     !
```

xpath_eval()

Az xpath_eval()-kontextus objektumot kiértékelendő sztringet és egy eredményobjektumot ad vissza. A következő példa az xpath_new_context()-ben létrehozott **$context-et használja, és** a **"/doc/title"** -t keresi. (A függvény nincs benne a PHP 4.0.6 Win32 bináris állományaiban, így az outputra most nem tudok példát mutatni):

```
$result = xpath_eval($context,
"/doc/title"); print_r($result);
```

-lll'}_ W9Í1 i

Gyors megoldások

nocqa

Az XML-fájlok megjelenítése

Az XML-fájlok tag-eket tartalmazó szövegfájlok. Hogy egy XML-fájl tartalmát megjelenítsd tesztelésre, elemezésre stb., egyszerűen kombinálj egy szimpla szövegfájl-megjelenítést a htmlentities()-zel, ahogy azt a következő kódban látod:

```
$line_length = 70;                                    _    ^q\^3J-'
$xml = file("./fishcurry.abw" ) ;
while(list($k, $v) = each($xml))
    while (strlen($v) > 0)                          *'    '       '       ^^

        print("<br>" . htmlentities(substr($v, 0, $line_length))); $v
        = substr($v, $line_length);
```

A file() az XML-fájlt beolvassa egy tömbbe, egy-egy tömbbejegyzést rendelve az input minden sorához (a sortöréskarakterek alapján). Az XML tartalmaz olyan SGML tag-eket, amiket a böngészők ismeretlen HTML tag-nek ismernek fel, ezért azokat a böngésző lenyeli. A htmlentities() tag kisebbmint nyitójelét (<) és a nagyobbmint zárójelét (>) a megjelenítésbiztos < és >-jelekre változtatja. A kódban van egy while()-ciklus a tömb végigolvasására, és egy extra while()-ciklus a soronkénti 70 karakterben meghatározott hossz megkurtítására.

A következőkben egy receptből a hozzávalók részt láthatod, AbiWord-ben készítve, ami egy a Windows-ra és Unix-ra készült szövegszerkesztő. A program az XML-t a dokumentumok formázásának tárolására, meghatározására és jelzésére használja:

```
<?xml  version="1.0"?>
<abiword version="0.7.14" fileformat="l.0">        rwv
  <!— This file is an AbiWord document.                                  —>
  <!— AbiWord is a free, Open Source word processor.                     —>
  <!— You may obtain more information about AbiWord at www.aJDisource.com —>
  <!— You should not edit this file by hand.                             —>
<i                                                                    ^
 <!— Build_ID = (nőne) —>                                  ,^„,.„
  <!— Build_Version = 0.7.14 —>
  <!— Build_Options = LicensedTrademarks:Off Debug:0ff BiDi
   :Off Pspell:Off —>
  <!-- Build^Target = /home/tom/release/abi/src/WIN32_l.1. 8
  _i386_OBJ/obj -->                              - .      -- _ .
  <!— Build_ComₚileTime = 14:34:43 ->       t   .    ™\ ~ *
  <!— Build CompileDate - Mar 30 2001 -->    H^
```

```
<pagesize pagetype="Letter" orientation="portrait" width="8.500000" he
ight="ll.000000" units="inch" page-scale="l.000000"/>
<section>
<p>l teaspoon of chili flakes</p>
<p>l soup spoon of capsicum</p>     jv ;-;>,
<p>l can coconut cream</p>
<p>500 gm light, boneless fish.</p>
<p>l packet of green vegetables</p>
<p>l packet of corn</p>
<p>mustard oil</p>
<p>mustard seed</p>
<p>ground mild chili</p>
<p>turmeric</p>              ...      -.t .
<p>cumin</p>
<p>medium onion chopped fine</p>
<p>fennel seed</p>
</section>
</abiword>
```

Az AbiWord-dokumentum rendelkezik az XML minden alapvető funkciójával, kivéve a DTD-t. Az <abiword>-tag határolja körül a dokumentumot. A <p> egy bekezdést fog közre. Vannak a szoftver leírásához megjegyzés tag-ek (<?— —>) a papírméret és formátum megadásához pedig a <pagesize>-tag. Hogy ne raboljam az idődet, és papírt kíméljek meg, a dokumentum nem tartalmazza a teljes receptet vagy a formázás AbiWord-ben található teljes eszköztárát. További kísérletezgetéshez töltsd le az AbiWord-öt. Ha a recept érdekel, küldj egy üzenetet a honlapomról.

Egy folyamatos adatáram vagy egy nagyon nagy fájl megjelenítéséhez olvasd el a Gyors megoldások rész „Az XML-adatok megjelenítése" c. részt. Hogy megtudd, a PHP XML függvények hogyan értelmeznek egy fájlt, olvasd el a „Az XML-adatok értelmezése" című részt.

Az XML-adatok megjelenítése

Előfordul, hogy egy XML-adatfolyam tartalmát szeretnéd tesztelésre vagy elemzésre megjeleníteni, de nem lehetséges a fájl egyszerű betöltése a memóriába, majd az onnan történő kiíratása, ahogy azt az előző példa mutatta, mert vagy túl nagy a fájl a memóriához, vagy mert az adatok folyamatosan áramlanak egy hálózati portról. Ez a megoldás az adatok feldolgozásának és megjelenítésének egy hosszabb módja, amihez nem kell, hogy az összes adat rendelkezésre álljon a megjelenítés kezdetekor. Az adatfolyam emulálásához a kód ugyanazt a fájlt olvassa, mint az előző példában, csak sorról sorra. Ezt a fájlolvasást a többi fejezetben ismertetett bármelyik függvénnyel helyettesítheted.

A következő kód az input-fájlt olvassa be sorról sorra a fgets() használatával, minden sorban megkeresi a < és >-jelekkel határolt tag-eket, és a $tags-tömbbe gyűjti őket:

```
$line_length = 70;
$file_name = "./fishcurry.abw";
$data = "";
```

```
if($file = fopen("./fishcurry.abw", "r")) {
   while(!feof($file))

       Sdata .= fgets($file, $line_length); if
       (substr(Sdata, -1) == "\\n")

       Sdata = substr(Sdata, 0, -1);

       if(substr(Sdata, -1) == "\\r") Sdata

       = substr(Sdata, 0, -1);

       $offset = 0;
       while ($offset < strlen (Sdata) )

           Spos = strpos(Sdata, "<", $offset) ;
           if(Spos === falsé)

               Soffset = strlen(Sdata);

           elseif(substr(Sdata, Spos, 2) == "</")

               Send = strpos(Sdata, ">", Spos);
               if($end === falsé)

                   $offset = $pos + 1;

               else

                   $tags[] = substr(Sdata, 0, $end + 1) $data
                   = substr($data, $end + 1); Soffset = 0;


           elseif($pos == 0)
               {
               $offset = $pos + 1;
               }
           else
               {
               $tags[] = substr(Sdata, 0, $pos);
               $data = substr($data, $pos);
               $offset = 0;


    felose ($file); }
if(strlen($data)) {
   $tags[] = $data;
```

un, az

te

```
while (üst ($k,    $v)    =
    each($tags)) { while(strlen($v)
    >   0)                                                    ;íiiríw

        print("<br>"    .    htmlentities(substr($v,      0,
        $line_length))};
        $v    =    substr ($v,      $line_length) ;      , ..      ... _ .,...,____ _ .__
```

A **fopenQ, az fgetsQ, a feof()** és **az fclose()** a 8. fejezetben bemutatott normál fájlkezelő függvények, és helyettesíthetők lennének hálózati beolvasófüggvényekkel. A fgets() min den új adatot hozzáad a Sdata-hoz, amely a teljes tag-ek kivonására van beállítva. A $data-ban megmaradt adatok aztán hozzácsapódnak az fgets()-ből származó következő adatszeg menshez. ;c > 39EÍ"Í3<) Sii.'iW

A tag-ek kivétele a Sdata-ból alapjában véve ugyanaz, mint az előző megoldásban használt kivonás, egy kis plusz logikai elemmel. Mikor a kód egy </-rel jelölt záró tag-et talál, az adatokat a záró tag-ig kivonatolja, így a különálló záró tag-ek külön tömbbejegyzésekbe kerülnek. Nincs külön logika azon tag-ekre, amelyek kezdő és záró tag-ek is egyben. Ha a fő fájlbeolvasó while()-ciklus után még marad adat a $data-ban, az egy $tags-bejegyzésbe **kerül.**

A kód egy whileQ-ciklussal zárul, amely végigolvassa a tömböt, és soronként 70 karakterig kiírja az adatokat és a tag-eket.

Hivatkozás:

Szövegfájl megjelenítése **279**

Itt következik az output utolsó része, ahol látszik a tag-ek elválasztása, különösen az egyszeres záró tag-eké:

```
<p>medium    onion    chopped
fine</p> <p>fennel    seed</p>                                  rdstr! Í5ET?

</section>
</abiword>
```

A tag-ek **$data-ba** való gyűjtése és rákövetkező megjelenítése nem valószínű, hogy alkalmas a sokáig futó, adattöredékeket hálózatról beolvasó szövegekre, mert túl nagy a veszélye a hálózati kimaradásnak. Ilyen esetekben az adatok egy fájlba vagy adatbázisba kerülnének, s mindegyik egy-egy sorát képezné a fájlnak vagy az adatbázis táblázatnak. Akkor rögtön el is olvashatnád a 9. fejezetet, és írhatnál egy szép formulát, hogy szakaszról szakaszra végigmehess a fájlon vagy táblázaton, ahol egy-egy szakasz tetszőleges számú sorból áll vagy bizonyos tag-ek megtalálásán alapul.

XML-adatok értelmezése

Ez a megoldás a PHP XML-függvény segítségével ellenőrzi az XML-dokumentumokat, az előző megoldásban látott AbiWord-dokumentumot, valamint az alapvető fájlfeldolgozó függvényeket használva.

Első lépésben hozzuk létre az értelmezőt. A következő kód az xml_parser_create()-tel végrehajtja ezt. A $parser-ben lévő értelmező'azonosító minden egyéb XML-függvényben használatos:

```
if(!$parser = xml_parser_create
    print("<br>xml_parser_create failed. " ) ;
```

Az értelmezőhöz különböző feldolgozókezelők szükségesek a beérkező XML-folyam részeinek feldolgozására. A legalapvetőbbek a kezdő- és záróelemkezelők. A következő kód a kezdőelemkezelő. A példa kiíratja a kezdő elemet. A következő gyors megoldás ("XML nyitó- és záróelemek összeillesztése") annyiban megy tovább, hogy csoportosítja a tageket:

```
function start_element_handler($parser, $tag,
    $attributes) print("<br>start_element " .
    htmlentities($tag));
```

A bemenő adatok között van a kezdő tag nevét tartalmazó sztring és az attribútumok listáját tartalmazó tömb. Az AbiWord-dokumentumban csak azok a tag-ek szerepelnek attribútummal, amelyek nem relevánsak egy weboldalon.

A következő kód a záróelem-kezelő. Ez a kezelő megkapja a tag nevét, így ez a művelet össze tudja párosítani a zárótag-et a nyitóval. A példa csak kiíratja a tag-et, így láthatod, mi történik:

```
function end_element_handler($parser, $tag)
    print("<br>end_element  htmlentities
                           ($tag)
```

Ha XML-feldolgozó kezelőket hozol létre, a kezelőket be kell jegyezned az őket használó értelmezőbe. A következő kód a kezdő- és a záróelem-kezelőt is regisztrálja. Minden XML set_handler-függvényben megválaszthatod, hogy ne fusson le a regisztráció, ha nincs bejegyezhető kezelő, vagy ki is hagyhatod az egyes kezelőket egy üres sztring használatával:

```
if(!xml_set_element_handler($parser, "start_element__handler",
    "end element handler"))
    print("<br>xml_set_element_handler failed.");
```

A karakteradat-kezelők a tag-ek között elhelyezkedő adatokat kezelik. A következő példa csak kiíratja az adatokat, de egy élesben működő feldolgozó rendszer az adatokhoz kapcsolódó zárótagig gyűjti össze az adatokat, majd végrehatja, amit a tag alapján kell:

```
function character_data_handler($parser, $data)

    print("<br>character_data htmlentities($data));
```

Regisztráld a karakteradat-kezelőt a következő kóddal. A karakteradat-kezelő minden XML-feldolgozó függvényhez szükséges, különben haszontalan lenne az XML értelmezése:

```
if(!xml_set_character_data_handler($parser, "character_data_handler"))

i       print("<br>xml set character_data_handler failed."); - (
    i
```

Az alapértelmezett kezelő az a kezelő, amely mindent megkap, ami máshol nincs feldolgozva. Először állítsd be úgy a kezelőt, hogy mindent kiírjon. Ha már más kezelőkön minden fontos végigfutott, az alapértelmezett kezelőben lévő maradék nagy valószínűséggel már nem érdekes a honlapod feldolgozási folyamatainak szempontjából:

```
function default handler ($parser, $data)      -<   < -lamB' -•  i

    print("<br>default htmlentities($data)
                                );
```

Regisztráld az alábbi kóddal az alapértelmezett kezelőt:

```
if(!xml_set default handler($parser, "default_handler"))

    print("<br>xml_set_default_handler failed.");
```

Talán hasznos lehet egy külső elemre való hivatkozás, de még látnom kellene egy példát, amit egy weboldalhoz kell feldolgozni. Ha valaki XML-ben küld neked cikkeket, és az XML vagy DTD tele van a küldő oldalára vonatkozó hivatkozásokkal, akkor problémás lehet az adatok megbízható feldolgozása. A következő kód kiíratja az adatokban talált külső elemre való hivatkozásokat, arra az esetre, ha lennének ilyenek a tesztfájljaidban (az AbiWord-példában nincsenek):

```
function external_entity_ref_handlert$parser, $entity_names,
    $base, $system_id, $public_id)

    print ("<br>external_entity_ref " . htmlentities($entity
    names));
```

A következő kód segítségével regisztráld a külső elemre való hivatkozást:

```
if(!xml_set_notation_decl_handler($parser, "notation_decl_handler"))

    print ("<br>xml set notation_decl_handler failed.");
```

A jelölésdeklaráció-kezelő a |elölésmeghatározásokat kezeli. A következő kód kiíratja a jelölést, arra az esetre, ha lennének jelölésdeklarációk a fájljaidban:

```
function notation_decl_handler($parser, $notatíon, $base, $system id,
   $public_id)                                                          fii

   print ( "<br>notation_decl " . htmlentities($notation));
```

A következő kód segítségével regisztráld a jelölésmeghatározás-kezelőt:

```
if(!xml_set_notation_decl_handler($parser, "notation_decl handler"))
   {
   print("<br>xml_set_notation_decl_handler failed.");              í
```

A feldolgozóutasítás tartalmazza a célfolyamat nevét és a feldolgozandó adatokat. A következő kezelő fogadja és kiírja a célt és az adatokat:

```
function processing_instruction_handler($parser, $target, $data)
   {
   print("<br>processing_instruction " . htmlentities($target, $data));
   }
```

Az alábbi kód segítségével regisztráld a feldolgozó utasításkezelőt:

```
if(!xml_set_processing_instruction_handler(Sparser,
   "processing instruction handler"))
   {
   print("<br>xml_set_processing_instruction_handler failed.");
   }
```

Itt látható egy másik kezelő arra az esetre, ha az adataid között lennének nem értelmezett elemmeghatározások:

```
function unparsed_entity_decl_handler($parser, $entity, $base,
   $system_id, $public_id, $notation)
   {
   print ("<br>unparsed_entity_decl " . htmlentities($entity,
      $notation));
   }
```

A következő kóddal regisztrálhatod a nem értelmezett elemdeklarációk kezelőjét:

```
if(!xml_set unparsed_entity_decl_handler($parser,
   "unparsed entity_decl_handler"))
   {
   print<"<br>xml_set_unparsed_entity_decl_handler failed.");
   }
```

Nos, túl vagyunk az unalmas részeken, kezdődhet az igazi munka. A következő kód a fichcurry.abw-fájlt fogja ciklusba ugyanúgy, ahogy az előző megoldásban látott kód, de az adatokat ezúttal az xml_parse()-ba továbbítja:

```
$line_length  =  7 0;
if($file = fopenC'./fishcurry.abw", "r"))
   {
   while(!feof ($file))

      $data = fgets (\$file, $line_length) ; if
      (substr ($data, -1) == "\n")
```

```
...  {
        $data = substr($data, 0, -1);
    if (substr($data,     -1)    ==    "\r")
        $data = substr($data, 0, -1);                    JCIJ"; -niiq
    if(!xml_parse(5parser,     $data))
        print("<br>xml_parse    failed    with          htmlentities($data))
        "

if (! xml_parse ($parser,     "",    true) )     ,'"
    print("<br>xml_parse    failed   on
eof."); fclose($file);
```

Minden beérkező sor újsor, vagy kocsivissza/újsor-karakterben végződik, ezért a kódban kétszer is benne kell legyen az if(substr($data, -1), hogy az új sorokat és a sor elejére ugrást is kivegye. Az xml_parse()-függvény feldolgozza az értelmező azonosítót, valamint az adatokat a fájl végével bezárólag. A fájl végét követően a feldolgozás befejezésére kódunk ismét az xml_parse()-t hívja meg, s most egy harmadik, igazra állított paraméterrel is rendelkezik a fájl végének jelzésére.

Ha az ellenőrzés befejeződött, az erőforrások felszabadítása érdekében a következő kóddal zárhatod be az értelmezőt. Általában nincs gond a PHP-val, mikor a szkript végén bezárja az értelmezőt, kivéve a rendszer terheltségét, ami a szkript elkészültéig fennáll:

```
if{!xml_parser_free($parser))
    print ("<br>xml_parser_f ree    f a i l e d . ");      trína   uz*& if-o. fCjd>" ) sr.': :r.
```

Az alábbi output az eredeti megkurtított változata, néhány sokszor ismétlődő sor kihagyásával:

```
default <?xml version="l.0"?>
start_element ABIWORD
default <!-- This file is an AbiWord document. -->
default <!-- Build_CompileTime = 14:34:43 —>
default <!-- Build_CompileDate = Mar 30 2001 -->
start_element PAGESIZE
end_element PAGESIZE                                    H - ' "
start_element SECTION
start_element P
character_data 1 teaspoon of chili flakes
end_element P
start_element P                          **
end_element P
end^element SECTION      ; -.
end element ABIWORD
```

Láthatjuk, hogy az XML-verzió számához és néhány megjegyzéshez az alapértelmezett kezelő volt szükséges. A kezdő- és záróelem-kezelők megjelentek, csakúgy, mint a karakter adatkezelő. Figyeld meg, hogy a tag-neveket nagybetűre állítottuk, hogy elkerüljük a betűkből adódó összekeveredést. Gondolkozz el, hogyan lehetne az ABIWORD kezdőelemet a záróelemmel összekapcsolni az egész fájlon keresztül.

XML nyitó- és zárótag-ek összeillesztése

Az előző Gyors megoldással egy XML-fájlt értelmezhetsz, és az XML által kiválasztott sorrendben megnézheted a visszaküldött alkotóelemeket. A következő lépés néhány tag összekapcsolása és feldolgozása. Ez a megoldás az előző megoldáshoz ad tag-kapcsoló kódot.

Hogy a zárótag-eket az adatokhoz és a kezdőtag-ekhez kapcsold, szükséged lesz egy tömbre, amelyben a kezdőtag-ek és az adatok tárolódnak. A következő $elements-tömb a kezdőelem-kezelő, a záróelem-kezelő és a karakteradat-kezelő között létesít kapcsolatot. Miután elolvastad a 17. fejezetet, a $elements-t és néhány itt következő programot objektumként is alkalmazhatsz:

```
Selements = arrayO;
```

A karakteradat-kezelőt meg kell változtatni, hogy egy tag számára összegyűjtse az adatokat. A karakteradat-kezelőt behívhatod az adatok egyes részeivel vagy az összes adattal, de ezen a kezelőn belül nincs rá mód, hogy megtudd, mikor vannak kész az adatok. A $elements-ben az aktuális elem mindig a nulla bejegyzés, az aktuális elemen belül pedig az adatok a "data" elemben gyűlnek. A "data"-t mindig a kezdőelem-kezelő hozza létre, így ez a karakteradat-kezelő egyszerűen hozzáfűzi az adatokat:

```
function character_data_handler($parser, $data) {
    global $elements;
    $elements[O]["data"] .= $data;
```

A kezdőelem-kezelő elkezdi a tag feldolgozását és minden attribútumot tartalmaz, ezért a tag-név és az attribútumok is új bejegyzésbe kerülnek a $elements-tömbben. Hogy a kód könnyebben érthető legyen, az új elem zéró elemként lesz beszúrva az array_unshift()-függvény segítségével, amely az új elemet a tömb elejére, az összes többi elemet pedig egy szinttel feljebb tolja. A "data" nulla hosszúságú sztringre van beállítva, így a karakteradat kezelő hozzá tudja fűzni az adatokat. Az egyszerű, <x/> formátumú tag-eket az XML kód dupla tag-nek fordítja <x></x> formában, ahol az első a kezdőelem-kezelőn, a második a záróelem-kezelőn fut át. Ez a példakód minden tag-nél a záróelem-kezelőben dolgozza fel a tag-eket, mikor azok már készen vannak. Figyeld meg, hogy a könnyebb feldolgozás érdekében a tag-ek kisbetűsre vannak állítva. A PHP XML-függvények az XML tageket nagybetűsre változtatják, ami számomra nagyon bosszantó, mintha kiabálnának, szóval én mindig visszaállítom kisbetűre:

```
function start_element__handler($parser, $tag, $attributes)

    global $elements;
    $x["attributes"] =
    $attributes; $x["data"] = "";
    $x["tag"] = strtolower($tag);          « ifiíooqíJI-V.Í-Ó íwnnoboiu a
    array_unshift($elements, $x) ;         :un
```

Miután minden tag le van zárva, a következő záróelem-kezelő elvégez minden tag-specifikus feldolgozást. A Selements-ben összekapcsolja a beérkező zárótag-et az aktuális kezdő tag-gel, és hibaüzenetet küld, ha a párosítás nem sikerül. Ha fölös zárótag-ed van, ez a kód kidobja azt, és úgy folytatja tovább. Ha plusz kezdőtag-ed van, a kód egy rakás hibaüzenetet produkál, mert a zárótag-ek nem kerülnek szinkronba a kezdőtag-ekkel. Mindkét esetben hiba lép fel az input XML-fájlban. Ha el akarod kerülni a túl sok hibás fájl vagy a rengeteg hibaüzenet miatt bekövetkező böngészőhibák kezelését, használj hibaüzenet-számolót, és lépj ki egy bizonyos számú hiba, mondjuk 10 után. Ne felejts el minden olyasmit megjeleníteni, ami segíthet megtalálni a hibákat egy nagy fájlban, beleértve a bájt és rekordszámolást is.

Néhány tag-nek van attribútuma, néhánynak nincs, ezért a hibák elkerülése végett az attribútumelem feldolgozását az isset()-be ágyazták. Egy tag attribútumait és az adatokat az array_shift()-tel távolíthatjuk el a $elements-tömbből, és a könnyebb kódolás érdekében a helyi $data- és $attributes-változókba kerülnek. A switch() minden tag-hez kiválasztja a megfelelő feldolgozást, és van egy alapértelmezett beállítása, amely az ismeretlen tag-ekkel érkező üzenetek esetében egyszerűen kiirja a tag nevét.

A switch()-utasítás utáni feltételes elágazások, az "abiword"-tó'l a "pagesize"-ig, tartalmaznak minden tag-et, amit ez a kód szándékosan nem dolgoz fel. (Vedd észre, hogy break-utasítás hiányában a switch() átugrik egyik feltételes elágazásról a másikra, ezért előfordulhat, hogy több case-utasítás csupán egy eseményt eredményez.) Ezek a tag-ek attribútumokkal kiegészített HTML-megjegyzésekként kerülnek az oldalra. A kód ezen részét használhatod arra, hogy felszívd az adatok megjelenítéséhez nem szükséges tag-eket vagy azokat, amelyek feldolgozására még nem állsz készen. Mikor XML-feldolgozást fejlesztesz egy adott XML-fájlhoz, sokszor könnyebb ellenőrizni az első állomásokat, ha végig tudsz haladni az XML-fájl végéig, és egyszerűen csak kihagyni azokat a tag-eket, amelyekhez nem tartozik megfelelő feldolgozási folyamat.

A section-elágazás a section tag-et HTML tag-gé alakítja. Nem tudom, hosszú távon is ezt szeretnék-e az emberek az AbiWord-nél, de ez illeszkedik a HTML-struktúrába, és egy példa a tag-ek átalakítására. Ha sok egyező formájú tag-ed van, és néhányukat át kellene alakítani, felállíthasz egy fordítótömböt, és csak az abban szereplő tag-eket alakítod át.

A p-elágazás a különálló tag-ek feldolgozásának összetettségét mutatja. Ebben az esetben az AbiWord-fejlesztők különálló attribútumtag-ek helyett egyetlen props nevű tag-be tettek be számtalan formázó paramétert. A kód ezen része ki kell, hogy kódolja a kulcs/érték-párokat a props-ból, és a párokat az egyszerűség kedvéért a $attributes-tömbbe rakja. Egy kulcs, a text-align úgy tűnt, mintha közvetlenül a bekezdés align-paraméterének for-

dítódna, így ez feldolgozásra kerül, míg a kulcs/érték-párok a props-ból a $attributes-ba kerülnek. A $attributes-ban van egy ellenőrzés a dupla bejegyzések kiszűrésére, amely hibaüzenetet küld, így megtalálhatod azokat az elágazásokat, amelyek fedik egymást:

```
function end_element_handler($parser, $tag)

    global $elements;
    $tag = strtolower($tag);
    if($tag == $elements[O]["tag"])     '
        {
        if(isset($elements[O] ["attributes"]))

            $attributes = Selements[0]["attributes"]

        $data = $elements[0]["data"];
        array_shift(Selements);
        switch($tag)

            case "abiword":
            case "c":
            case "field":
            case "1":
            case "lists": case
            "pagesize":
            print("<!-- "
                                $tag . " ");
               while (üst ($k, $v) = each ($attributes) ) {
                   print(" " . strtolower($k) . "=\"" .   $v

               print($data . " -->\n"); break;
            case "section": $tag = "div"; print("<div" .
               $tag); while (üst ($k, $v) = each ($attributes)
               )

                   print(" " . $k . "=\»» . $v . "\"");

            print (">■• . $data . "</" . $tag . ">\n"); break;
            case "p":
               if (isset ($attributes["PROPS"] ) )   ''"'

                   $props = explode("; ",
                   $attributes["PROPS"]); while (üst ($k,  rsoDr»5S' $v) =
                   each ($props) )
                       {
                       $att = explode(":", $v);
                       if($att[0] == "text-align")
                           {
                           $att[0] = "alígn";
                           } if (isset
                       ($attributes[$att[0]]))
```

```
                    print("<br>Duplicate attribute; " . $att[O]


                else
                                                        bre
                    $attributes[$att[O] ] = $att[l];

                }                                  1 ., OK í- Jfe    «
            unset ($attributes ["PROPS"] ) ;   « ;f»j. :,C■ '-= -;6
            }                                    ^' e^'riíe,. tíi;r<: .*n
        print("<" . $tag);
        while (Üst ($k, $v)           = each ($attributes) )
            {
            print(" " . $k          . "=\"» . $v . "\"");

    print(">" . $data . "</" . $tag . "; break;
    default:
        print ("<br>end_element " . $tag . " not processed.\n") ;


else (
    print("<br>end_element    "    .
        $tag .    $elements[O]["tag"]);     "   does   not
                                          match   "
```

Az alábbi a kiíratott eredmény egy része, a jobbra zárt bekezdéssel: 1

teaspoon of chili flakes

1 soup spoon of capsicum

1 can coconut cream

íme a kiíratott eredmény mögötti HTML egy szakasza (a pagesize-tag HTML-megjegy-zésként jelenik meg, a paragraphtag-ben pedig ott az align="right")

```
<!— pagesize pagetype="Letter" orientation="portrait"
width="8.500000"
    height="ll.000000" units="inch" page-scale="l.000000" -->
<p>l teaspoon of chili flakes</p>
<p align="right">l soup spoon of capsicum</p>
<p>l can coconut cream</p>                          _ ^_.
```

Az AbiWord-dokumentum dekódolásának példája számos lépést mutat meg, amelyek segítenek neked kedvenc XML-dokumentumod dekódolásában. Néhány tag és attribútum nem alakítható webolda- paraméterré, ezért nincs értelme dekódolni őket. Néhány XML-tag és -paraméter úgy van kialakítva, hogy az érvénytelen adatok bevitelét megakadályozza, így ezekre nincs szükség, ha csak megjelenítésre kódolod ki az adatokat. Próbáld meg a kódot saját XML-fájljaidra alkalmazni. Válassz kis fájlokat, hogy a fájl mérete ne akassza ki a böngészőt. Ahol a kód a tag-eket HTML-megjegyzéssé alakítja, ott világosszürke szöveggé
alakíthatod őket, hogy összefüggésükben lásd a tag-eket.

i: . .iDe-Sie^cs aisun -p.iraine *étér.tk* tor-

Úgy látszik, hogy az AbiWord-dokumentumok feleslegesen összetettek, és túl sok attribútumot használnak ott is, ahol az önálló tagek hasznosabbak lennének. Az attribútumok úgy tűnik, kompatíbilisek lehetnének a stíluslapokkal, de ez már meghaladja e könyv terjedelmét. Én továbbra is folytatom az AbiWord-dokumentumok weboldallá való dekódolását. Ha érdekel a frissített kód, küldj egy üzenetet a honlapomról.

Segédlet

Algoritmusok

Függvények

Opciók

Operátorok

Típusok