1.344 FT

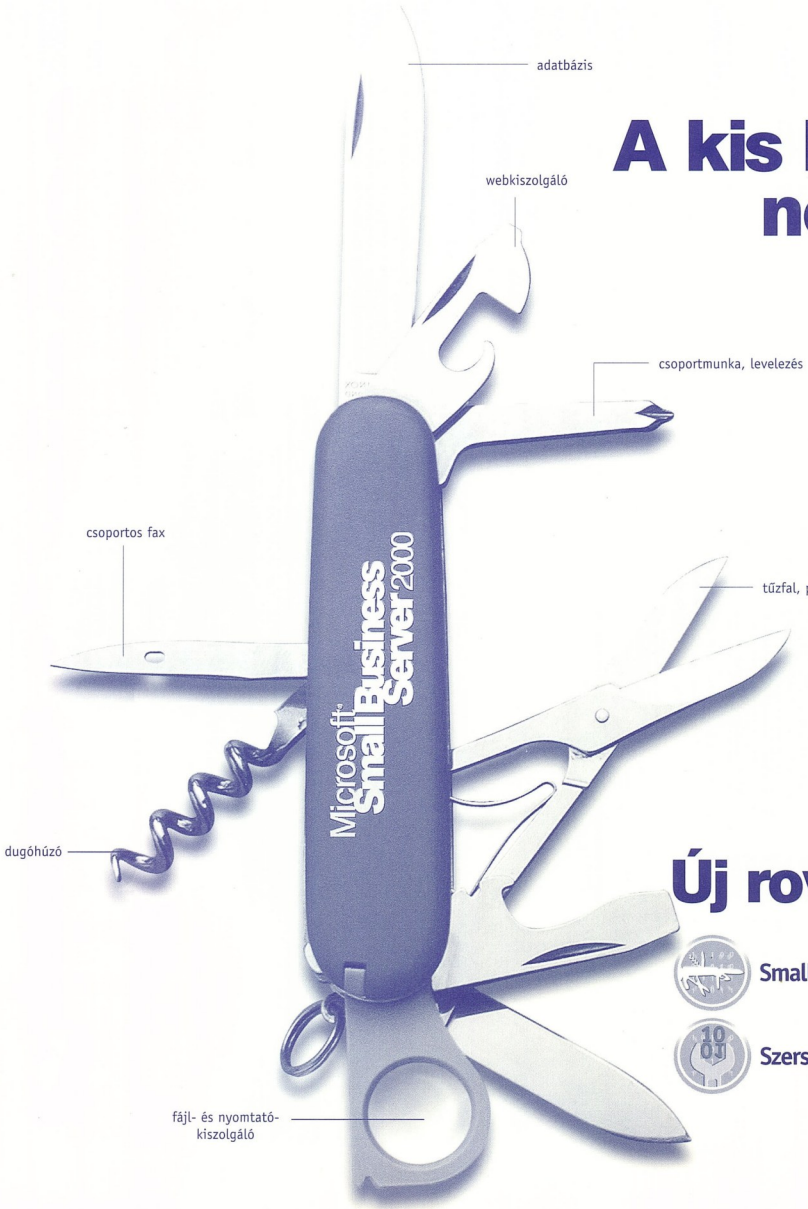
II. ÉVFOLYAM 6. SZÁM

ISSN 15845185 06

9 771586 518005

tech.net

A MICROSOFT MAGYARORSZÁG SZAKMAI MAGAZINJA



A kis késit nekik!

Új rovatok!

Small Business Server

10 ÚJ

Szerszámosláda

2001 Június

tech.net

A Microsoft Magyarország Szakmai Magazinja

Szerkesztőség
Főszerkesztő: **Fóti Marcell**
marcellf@netacademia.net
Főszerkesztő-helyettes: **Fülöp Miklós**
mick@netacademia.net
Szerkesztőség címe:
1105 Budapest, Ihász utca 13.
Tel.: 263-2732
technet@netacademia.net
Nyilvános levelezési lista:
tech.net@lyris.netacademia.net

Kiadja és terjeszti
a **NetAcademia Kft.**
Terjesztési, előfizetési információ:
Tel.: 263-2732
terjesztes@netacademia.net
Megjelenik havonta, ára 1.344 Ft
Példányszám: 3.000

Előfizethető megrendelőlevélben a szerkesztőségnél:
1105 Budapest, Ihász utca 13.
Fax: 261-7145
http://technet.netacademia.net/subs

Hirdetésfelvétel:
Bársonykalapács Marketing
Felelős: **Udvarev Rita**
Tel./Fax: 214-0923
info@velvethammer.hu
1027 Budapest, Fő utca 67. V. 1.
Grafikai tervezés, kivitelezés, nyomdai előkészítés:
Bársonykalapács Marketing
Művészeti vezető: **Balogh Zoltán**

Nyomda:
Cerberus Kft.
1066 Budapest, Lovag u. 14.
Felelős vezető: **Schmidt Gábor**

ISSN 1586-5185

A fiúk a bányában... és a horgászok

Az alkalmazott tudományoknak mindig is voltak központi fellegvárai – olyan kutatóközpontok, melyben egyesültek az emberi és anyagi erőforrások egy adott technikai probléma megoldására és kifejlesztésére. Alkalmazott tudományokról lévén szó, a legnagyobb központok általában nem állami támogatással működnek, hanem egy-egy vállalat üzleti célú fejlesztéseinek kutatási hátterét adják, így az informatikai újítások zöme is piaci szereplők, cégek által működtetett kutatóközpontokban lát napvilágot.

Kályhának tekinthetjük az AT&T Bell Laboratóriumot, ahol többek közt a UNIX-ot is kifejlesztették, de innen jött a C nyelv, a celluláris távközlés, a T-1-es vonali adatátvitel, a tranzisztor és még néhány más Nobel díjas találmány is, ahogy ezt az utód Lucent Technologies reklámjában is olvashatjuk. A másik kályha természetesen a Xerox által működtetett PARC *(Palo Alto Research Center)*, ahol nem kisebb dolgokat találtak ki, mint a grafikus felhasználói felület, a legördülő menük vagy maga az egér. Történelmi már az a mindannyiunk számára közismert pillanat, amikor Steve Jobs és Bill Gates együtt látogatott el a PARC-ba, ahol egy magyar származású mérnök egy új megközelítésű kezelőfelülettel ellátott operációs rendszer tesztváltozatát mutatta be nekik. Ez a magyar mérnök, Charles Simonyi azóta a Microsoft-nál dolgozik már több, mint 25 éve - ahol szintén létrejött egy, az előzőekhez hasonló kutatóközpont, a Microsoft Research. Az MS Research kutatási eredményeinek egy részét közvetlenül is láthatjuk a Microsoft termékekben, *(mint pl. az SQL Server English Query, a hangfelismerés az Office XP-ben)*, azonban a kutatások nagy része nem termékspecifikus. Egy, az MS Research-öt bemutató előadáson engem nagyon meglepett, hogy Redmondban olyan, kifejezetten matematikai kutatások is folynak, mint például az „utazó ügynök" néven ismert probléma megoldásának keresése – azaz a feladatok lineáris időben való algoritmizálhatóságának kutatása. De ott van a misztikus „szándék alapú" programozás *(intentional programming)* kérdésköre is, amivel Charles Simonyi és kollégái már több, mint egy évtizede foglalkoznak. A kutatáshoz sok idő, és persze még több pénz kell – a Microsoft úgy tűnik, mindkettőt hosszú távon is biztosítani tudja. A fiúk lemennek a bányába, néha feljönnek levegőt venni, elmesélik hogy állnak a munkával – ha az eredmény nem megfelelő, visszaküldik őket. Egész addig, míg fel nem hozzák a mélyből azt a gyémántot, amit keresnek.

Amikor a fiúk nemrég feljöttek, hoztak magukkal néhány dolgot, amiből idén április elejére egy kész termék lett: megjelent végre a SharePoint Portal Server 2001 *(rövidítve SPS, korábbi kódnéven Tahoe – ebből is látszik, hogy a fiúk mivel töltik egyébként szabadidejüket: mind Whistler, mind Tahoe egy-egy síparadicsom neve Washington államban, Redmondtól kicsit északra)*. Mit is tud az SPS? Három fő funkciója az átfogó ismeretkezelés, a portálszolgáltatások és az elektronikus dokumentumkezelés. Az első, az ismeretkezelés kicsit fellengzősen hangzik, de épp ez az a terület, ahol a legtöbb MS Research kutatási eredmény megtestesül – az SPS már-már mesterséges intelligenciához hasonló eszközök arzenálját vonultatja fel ahhoz, hogy uralni tudjuk a fájlszervereinken és egyéb információtároló alkalmazásainkban kialakult dokumentumkáoszt.

A könyvtárban a könyvtáros a saját természetes intelligenciáját használva dönti el, hogy egy könyv milyen kategóriába kerüljön. Az SPS-nek csak megmondjuk a kategóriákat, megmutatunk néhány dokumentumot, ami ebbe a kategóriába tartozik – az SPS pedig magától megtanulja, mi a kategória szabálya. Az összes többi dokumentumot *(legyen az több ezer)* indexelés után már önmaga fogja elhelyezni a megfelelő kategóriákba.

Keresésnél szintén tartalom szerint, intelligensen elemzi az eredményeket, és aszerint adja vissza azokat, hogy valóban arról szól-e az adott dokumentum, mint amit kerestünk *(Nem pedig aszerint, hogy a főcímben, vagy alcímben szerepelt-e a keresett kifejezés...)* Félelmetes.

No és természetesen megvan az az előny, hogy ha kategória szerint keresünk, a kategóriarendszer nem alkot egy fix hierarchiát – tetszőlegesen választjuk meg, hogy milyen nézetet akarunk látni. Egy dokumentum természetesen több kategóriába is tartozhat. Az is jó lenne, ha azonnal kapnánk értesítést, ha megjelenne egy új könyv, amiben mongúzok szerepelnek. Az SPS ezt is tudja.

Az SPS alapja az az univerzális adattároló motor, ami az Exchange 2000-et is hajtja: a Web Storage System *(de az SPS teljesen független az Exchange-től, egyáltalán nem igényli a jelenlétét)*. A portálszolgáltatásoknál a Digital Dashboard technológia végre nemcsak egy SDK, hanem kész megoldás formájában jelenik meg. A dokumentumkezelési funkció hivatalos iratok kezelésében segíthet a verziókövetéssel, a dokumentumok hivatalos útjának munkafolyamatba szervezésével, a megfelelő szerepkörök, jóváhagyási útvonalak elektronikus kezelésével. A portálszolgáltatással együtt a dokumentumkezelés teljesen önálló webes közzétételi rendszert alkothat *(pl. vállalati intranet, vagy egy szerkesztőségi rendszer)*.

A Microsoftnak persze nemcsak kitűnő bányászai vannak, hanem mellettük egy kisebb horgászegyesületet is fenntart. A horgászfiúk folyamatosan figyelik a vizet, és ha egy hal nagyon ficánkol, kihalásszák. Nemrég egy NetIQ Operations Manager nevű halat fogtak. Az SPS még alig született meg, máris egyedül érezte magát: a horgászok így április végén fogtak neki egy társat: a neve NCompass Labs Resolution 4.0. A kifogott halak a Microsoft-nál egy ideig hittanórára járnak, hogy megtanulják az új vallást, majd újra is keresztelik őket.

Ebben a keresztségben az NCompass Resolution is új nevet kap majd, így összel mi már a Microsoft Content Management Server 2001-gyel ismerkedhetünk meg. Bővül a .NET kiszolgálócsalád...

Horváth Tamás
Microsoft Magyarország
tamash@microsoft.com

Service Pack 2

Windows 2000 Service Pack 2

Megjelent a Windows 2000 Service Pack 2 **[1]**, amely a biztonsági hibajavítások mellett néhány újdonságot is tartalmaz:

- beállítható kompatibilitási szintek Windows 95-höz és Windows NT 4 SP5-höz
- 128 bites biztonsági alrendszer *(a telepítés után ez a rész már nem távolítható el)*
- új, külön letölthető telepítőeszközök *(deploy.cab)* és Support Tools
- UATA-100 eszközmeghajtók

Az angol nyelvű Windows 2000-re telepíthető javítócsomag a NetAcademia kiszolgálón is megtalálható **[2]**.

Nem titkos, de nem nyílt a Windows forráskódja

A Microsoft azt tervezi, hogy különböző feltételek mellett, de válogatott partnereinek *(szoftver-, és hardvergyártók, egyetemek számára)* elérhetővé teszi a Windows és más szoftverek kódjait, persze megfelelő jogi korlátozások és különböző licencfeltételek mellett **[3]**. A Microsoft hangoztatja, hogy ez a kezdeményezés *(„shared-source" **[4]**)* nem egyezik meg a manapság divatos open source megoldásokkal.

Licencelési tájékoztató magyarul

Számos új licenctípus *(köztük olyan frissítési opcióval, amely éves díjazású, a CAL árának 25-30%-ába kerül, és magába foglalja a mindenkori legfrissebb verzió, a korábbi verziók, valamint a terméktámogatás használatának jogát)* bejelentésével egyidőben megjelent a magyar nyelvű Licencelési tájékoztató, amely PDF és MSReader formátumban letölthető az **[5]** címről.

Microsoft Content Management Server 2001

Az Olvasónak talán ismeretlen a fenti cím; talán kicsit furcsa is, hogy miközben a hasonló célokkal készült SharePoint Portal Server szinte még meg sem jelent, a Microsoft máris valami újabb megoldással jelentkezik. Mégis: miután nyilvánvalóvá vált, hogy a webtartalom kezelésére komplett és hatékony eszközökre van szükség, a Microsoft felvásárolta az NCompass nevezetű céget **[6]**. A cég webtartalom-menedzsment eszköze, az NCompass Resolution 4.0 **[7]** már egy jó ideje nem kis cégeknél bizonyít. Az SQL 2000 alapon futó, a Commerce Server 2000-rel, a BizTalk Server 2000-rel, valamint a SharePoint Portal Server-rel már most is együttműködő alkalmazás épp a fentiek miatt nagyon könnyen beilleszthető a .NET infrastruktúrába: így lesz belőle majdan *(várhatóan 2001 őszén)* Microsoft Content Management Server. A **[8]** címen rövid regisztráció után megtekinthető Flash demo alapján igéretes kezdeményezésnek nézünk elébe.

A cikkben található URL-ek:

[1] http://www.microsoft.com/windows2000/
↳ downloads/servicepacks/sp2/default.asp
[2] http://download.netacademia.net/servpack/w2ksp2/
[3] http://www.microsoft.com/presspass/features/
↳ 2001/may01/05-03csm.asp
[4] http://www.microsoft.com/Business/
↳ Licensing/SharedSource/
[5] http://www.microsoft.com/hun/jogtisztasag/
[6] http://www.microsoft.com/presspass/press/2001/
↳ Apr01/04-30NCompassPR.asp
[7] http://www.ncompass.com/
[8] http://www.ncompass.com/products/
↳ ncompass+resolution/_product+tour.htm

Small Business Server 2000 *(I. rész)*

Nagy csomag, kis cégeknek

Most induló cikksorozatunkban bemutatjuk a Microsoft Small Business Server 2000 **[1]** programcsaládot, amely – nyugodtan állíthatjuk – a legjobb és legolcsóbb MS megoldás lehet a kisebb cégek informatikai hátterének biztosítására. Szinte mindent tud, amit a „nagyok" *(értsd a BackOffice család)*, persze itt-ott korlátokba ütközünk. Ha a korlátok már nagyon szorítanának, megvan a módja, hogy fájdalommentesen feloldjuk azokat: a Small Business Server 2000 *(a továbbiakban egyszerűen csak SBS 2000)* később frissíthető lesz a korlátok nélkül használható BackOffice kiszolgálócsaládra.

Mi van a csomagban?

Az SBS 2000 a következő komponenseket tartalmazza:

- Windows 2000 Server – fájl- és nyomtatáskiszolgáló, webkiszolgáló *(IIS)*, terminál-szolgáltatások
- Exchange 2000 Server – csoportmunka, internetes levelezés
- Internet Security and Acceleration (ISA) Server 2000 – tűzfal és gyorsítótár, virtuális magánhálózatok és RAS szolgáltatás
- SQL Server 2000
- megosztott faxszolgáltatás
- megosztott modemszolgáltatás
- Microsoft Health Monitor 2.1 – a kiszolgáló és az alkalmazások teljesítményének, paramétereinek ellenőrzése, naplózása
- FrontPage 2000 és Outlook 2000 kliensalkalmazások

– mindez *(tényleg)* hihetetlen jó áron. Az éremnek persze két oldala van, a később említendő korlátozások mellett a Small Business Server jellegzetessége, hogy az összes kiszolgálóalkalmazás ugyanazon a számítógépen fut – a kis cégnek általában nincs is szüksége ennél többre. Mire a kiszolgálók erőforrásigénye átlépi az egy gépen biztosítható határt, a vállalat valószínűleg éppen kinövi a Small Business Server-t. Az SBS 2000 erőforrásigénye a kiszolgálón a következő:

	minimális	ajánlott
processzor	PII 300	PIII 500
memória	128 MB	256 MB
merevlemez	4GB	2x4GB tükrözve
	CD-ROM, 3.5" floppy *(A meghajtóként)*	
képernyő	800x600, 256 szín	1024x768 *(az adminisztrációs eszközöknek)*
egyéb	Hálózati csatoló *(NIC)* Faxmegosztáshoz: CLASS-1 faxmodem Proxy, RAS, modem sharing: + 1 modem	

Amikor a Performance Monitor *(vagy a Health Monitor)* jelzi a processzor, a memória vagy a lemezegység túlterhelését *(bizonyos számlálók ilyenkor az égbe szöknek)*, itt az ideje, hogy a zsebünkbe nyúljunk, és bővítsük a gépet. Általánosan igaz az a nézet, hogy minél több szolgáltatást használunk, minél több adatot tárolunk, minél több forgalmat generálunk, annál nagyobb processzorteljesítményre, memóriára és annál jobb, nagyobb és gyorsabb háttértárra lesz szükségünk. A fenti táblázat rovatait tehát fenntartásokkal kezeljük; az esetünkben érvényes erőforrásigényt az általunk telepített alkalmazások és a felhasználók száma határozza majd meg.

Mitől Small Business a Small Business?

Az SBS 2000 jól definiálható korlátozásokat tartalmaz a BackOffice-hoz képest, ez különbözteti meg a nagytestvértől. Az első korlátozással már találkoztunk: bár a tartományba további tartományvezérlőket telepíthetünk *(persze külön Windows 2000 Server licenccel)*, az összes SBS kiszolgálóalkalmazásnak ugyanazon a gépen kell futnia. Apropó, tartomány: az SBS 2000 nem teszi lehetővé, hogy tartományok közötti megbízotti kapcsolatokat hozzunk létre. Windows 2000 környezetben ez azt jelenti, hogy az SBS tartományunk nem lehet más tartományi fa, netán erdő része. Nem, a mi kis SBS tartományunk a saját erdejében, az erdőt alkotó egyetlen fa egyetlen ágának csúcsán csücsül. A fától csak akkor láthatjuk meg az erdőt, ha továbblépünk a BackOffice felé.

Az SBS részeként szállított Exchange 2000 Server-ben nem hozhatunk létre telephelyek közötti csatolókat, úgynevezett site connector-t. Ezzel és az előző, tartományra vonatkozó korlátozással a Microsoft azt szeretné elérni, hogy az SBS-t ne lehessen hatékonyan használni nagyobb vállalatok telephelyein. Ha az anyacég a telephelyre költségtakarékossági okokból SBS-t telepít, hát tegye, de akkor le kell mondani a közös tartományi adatbázisról és a központilag felügyelt és együttműködő levelezésről és csoportmunka-kiszolgálóról. Az Exchange „cserében" tartalmaz viszont egy POP3 Connector-t, aminek segítségével az Exchange képes a külső Internetszolgáltatónál fenntartott privát postaládáinkból a céges mailboxunkba időnként letöltögetni a leveleinket. Ez a szolgáltatás tipikusan azoknak a cégeknek készült, akik egy ISDN, netán hagyományos telefonvonal segítségével érik el az Internetet, lassú, és elsősorban nem állandó kapcsolaton keresztül.

50 felhasználó

Az SBS 2000 legfeljebb ötven felhasználó/munkaállomás használatát engedélyezi. Miközben az Active Directory-ba végtelen számú felhasználót *(és postaládát is)* felvehetünk, az SBS csak korlátozott számú egyidejű ügyfélkapcsolatot tesz lehetővé, és ötven a maximális száma a létrehozható ügyfélkonfigurációknak is *(felhasználó+munkaállomás páros)*.

A telepítés

Az elődökhöz képest a telepítés kicsit megváltozott. Az SBS továbbra is közös programcsomagként telepíthető, azonban az operációs rendszer telepítését elválasztották a kiszolgálóalkalmazások telepítésétől. Ez nem jelenti azt, hogy az

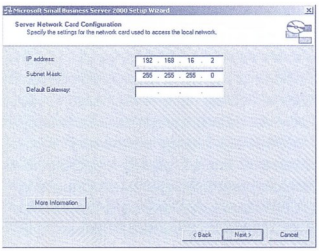
SBS telepítőkészlet nem tartalmazza a Windows 2000 Server-t, sőt, az SBS-nek muszály a saját CD-jéről W2K fészket rakni. Mindössze a telepítés menete bonyolódik egy kissé. A Windows 2000 Server telepítése a szokásos módon történik, az egyetlen különbség talán csak az, hogy a kódtáblák, nyelvi támogatás beállítása után a komponensek között felesleges matatnunk, az SBS telepítő a későbbiek során úgyis feltelepíti a hiányzó összetevőket. Ugyanez igaz a hálózati kapcsolatra is, a hálózati kártya beállításait – hacsak nincs rá a telepítés előtt szükségünk – hagyjuk alapértelmezésen. Miután a Windows 2000 Server elkészült, telepítsük a szükséges eszközmeghajtókat *(nyomtató, videókártya, stb.)*, majd az SBS CD-ről indítsuk el az SBS telepítőt. A főoldalon kattintsunk a Set Up Small Business Server linkre, erre elindul a telepítést segítő varázsló.

Az SBS telepítő varázsló figyelmeztet, ha hiányosságokat tapasztal a rendszerben

A továbbkattintás után megjelenő ablakban a varázsló figyelmeztet, ha a rendszer paraméterei nem érik el a minimális, illetve az ajánlott szintet. Ha a minimális korlátokat sem sikerül hiánytalanul teljesítenünk, a telepítés nem folytatódik. Érdekesség, hogy az elődökkel ellentétben most a modem nem, a hálózati kártya viszont feltétel az SBS telepítéséhez. A telepítés első lépése a Windows 2000 Server felkészítése az apaságra. Ennek érdekében települ majd a terminálszolgáltatás, valamint az Active Directory. Ne lepődjünk meg tehát, ha a telepítő varázsló az informatív adatok mellett címtárral, tartománnyal kapcsolatos kérdéseket is feltesz majd.

Az SBS és a kliensalkalmazások más-más licenckulcsot használnak. A második ábrán az automatikus bejelentkezés adatai láthatók

Mindenekelőtt azonban meg kell adnunk néhány más jellegű adatot: fogadjuk el a licencszerződést, írjuk be saját, vagy cégünk adatait, majd írjuk be illetve ellenőrizzük a licenckulcsokat *(az SBS önmaga és a kliensalkalmazások – MS Outlook – más-más kulcsot használnak)*. A telepítés során szükség lehet a rendszer újraindítására. A fenti ábra jobb oldalán látható ablakban a varázsló felajánlja segítségét: ha megadjuk a rendszergazda jelszavát, automatikusan bejelentkezik majd, és ott folytatja a telepítést, ahol abbahagyta; ellenkező esetben nekünk magunknak kell majd minden újraindítás után bejelentkeznünk a rendszerbe.



Hálózati konfiguráció – nincs túlbonyolítva

A következő érdemleges helyszín a hálózati beállítások panelja: az SBS telepítő minden hálózati kártyát letilt, és az egyiken saját alhálózatot hoz létre. A létrehozott alhálózat a privát 192.168.x.x címtartományba esik, bár ezt a telepítőben testreszabhatjuk. Általában érvényes az a tétel, hogy ahol az SBS-re van szükség, ott nagy valószínűség szerint teljesen mindegy, hogy a hálózat milyen IP címet és alhálózati maszkot visel, ezért érdemes elfogadni a telepítő alapértelmezését. A telepítés után ne felejtsük el újra engedélyezni a hálózati kártyákat *(bár erre a To Do List külön lépésben fel is szólít majd)*.

Néhány szó a SmallBusiness Kft-ről: cikksorozatunk áldozati báránya egy tipikus mai kis cég. A jelenlegi, egyértelműen javításra szoruló munkamenet a következő: a cég néhány tíz alkalmazottja egy irodaház 5. emeletén, öt-hat szobában dolgozik. A belső hálózat peer-to-peer alapú, az adatokat mindenki a saját *(vegyesen Windows 98, NT illetve Windows 2000 Professional)* számítógépén tárolja, szükség esetén bizonyos könyvtárakat megosztva biztosítja a többieknek a hozzáférést. A cég kifejezetten informatikus szakembert nem tart, inkább szükség esetén igénybe veszi egy MS megoldásszállító segítségét.

A cégnek egy ismert internetszolgáltatónál van néhány postaládája *(egy a cégnek, egy-egy a fontosabb alkalmazottaknak)*. Az internetszolgáltató üzemelteti továbbá a cég internetes honlapját is. A cég internetelérését a kivételezett számítógépekre aggatott modemek biztosítják, ugyancsak az internetszolgáltatónál fizetett hozzáférések segítségével.

A közös fax a titkárnő irodájában ontja a papírt, a bejövő faxokat igény szerint ő fénymásolja és osztja szét a címzetteknek. Ha valaki faxot szeretne küldeni, kinyomtatja, majd a faxhoz ballag, és elküldi *(esetleg megkéri a titkárnőt, hogy küldje el ő)*. A könyvelői csoport számítógépein található modemek a banki ügyfélterminál használatához szükségesek.

Az SBS tartomány

Az SBS tartomány *(ami nem más, mint egy Windows domain)* megválasztásánál kicsit óvatosabbnak kell lennünk. Nemcsak azért, mert a rendszer telepítése után a tartomány nevének megváltoztatására már nincs mód, hanem azért is, mert a Windows 2000 tartomány összenőtt az internetes névfeloldási hierarchiával *(DNS)*. Abban a pilla-

natban, amikor a cégünk az internetes világban is meg szeretne jelenni, saját tartománynevet kell majd regisztrálnia *(vagy talán már meg is tette)*. A Windows 2000 tartomány adatai is a DNS-ben találhatók, ez az információ viszont nem feltétlenül egyeztethető össze a külvilág által ismert tartománnyal, ráadásul sok esetben nem szükséges, mitöbb nem biztonságos, ha a tartományi információk a külvilág tudomására jutnak.

Esetünkben a következőről van szó: cégünk, a SmallBusiness Kft. régebben létrehozott weblapját egy internetszolgáltató saját szerverén üzemelteti, a www.smallbusiness.hu cím alatt. Ehhez a címhez tartozik egy DNS zóna, ami a smallbusiness.hu tartomány bejegyzéseit tartalmazza *(névkiszolgálók, levelező kiszolgálók, további tagok, pl. a www is)*. Mindez tőlünk függetlenül, az internetszolgáltató segítségével, a nagyvilágban működik.

Amikor az SBS kiszolgálót telepítjük, választanunk kell egy tartománynevet. Ha egyszerűen a smallbusiness.hu nevet választjuk, a tartomány adatai a smallbusiness.hu DNS zónába kerülnek bele, a zónát persze nem az internetszolgáltató DNS kiszolgálója, hanem az SBS saját DNS szolgáltatása kezeli. Ilyenkor két rossz megoldás közül választhatunk:

- „Behozzuk" a DNS zónát az internetszolgáltatótól – ekkor a világ felől érkező összes kérés az SBS kiszolgálóra esik be, terhelve a vállalati vonalat. Arról nem is beszélve, hogy ehhez állandó hálózati kapcsolat és fix IP cím szükséges *(tipikusan bérelt vonal vagy hasonló megoldás)*.
- „Kivisszük" a Windows tartomány adatait a szolgáltató DNS kiszolgálójára – ekkor a cégen belüli összes tartományi DNS lekérdezés az internetszolgáltatón keresztül fog zajlani, ami ugyancsak terheli a céges vonalat. *(Ha nem fix a hálózati kapcsolat, akkor az állandó lekérdezések miatt majd kvázi az lesz :-))*. Ráadásul az internetszolgáltatók által üzemeltetett DNS kiszolgálók nem biztos, hogy képesek a W2K által elvárt működésre *(dinamikus rekordfrissítés, SRV rekordok, stb.)*.

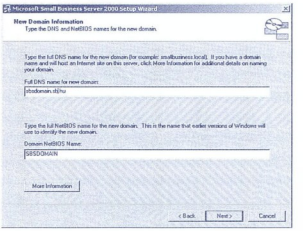
Bármelyik megoldást is választanánk, még ott lenne az a probléma, hogy a belső hálózatunk adatai *(belső IP címek, a belső felépítésre utaló adatok, stb.)* a replikáció során gyakorlatilag a világ összes DNS kiszolgálójára eljuthatnának. Ezek az adatok a tartományi felhasználókon kívül egy valakit érdekelhetnek: a fekete kalapos hackert, akinek egyetlen célja, hogy tönkretegye a konkurenciát.

Mi a megoldás?

Legjobban tesszük, ha az SBS tartományt a cég domain neve alatti aldomainként hozzuk létre, például így:
sbsdomain.smallbusiness.hu
A tartományi adatok ekkor nem közvetlenül a smallbusiness.hu alá kerülnek, hanem egy szinttel „lejjebb", az sbsdomain.smallbusiness.hu alá.

Hogy mi ebben a jó? A kulcs a delegáció, hölgyeim és uraim. A DNS-nek ugyanis jó alaptulajdonsága, hogy az óriási névtér egyes részeit más-más DNS kiszolgáló tartja karban. *(Még jó, különben nem is lenne képes működni.)* Amikor egy DNS zóna egy alzónáját valaki másnak „adjuk", ezt nevezzük delegációnak. Ilyenkor a mi DNS zónánkban az alzónánál egyetlen NS bejegyzés szerepel: „ha erről az alzónáról bármit tudni akarsz, őt kérdezd!".

Esetünkben tehát a smallbusiness.hu zóna marad az internetszolgáltatónál, ebben a zónában található az sbsdomain.smallbusiness.hu bejegyzés, ami egyetlen delegációs rekordban merül ki, és az SBS kiszolgálónkra mutat. Ha valaki az sbsdomain.smallbusiness.hu-ról érdeklődik, az SBS kiszolgálóhoz fordul majd, más kérdés, hogy a kiszolgáló válaszol-e neki *(természetesen csak akkor, ha az illető tartományi tag, azaz van köze az adatokhoz)*. Így már minden klappol: ami bent kell, az bennmarad, ami kint kell, az kinnmarad, és közben továbbra is minden működőképes.

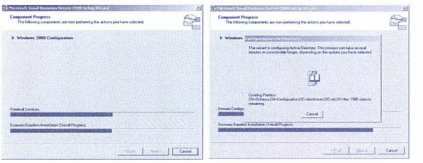


A tartománynév kiválasztásakor legyünk óvatosak és gondolkodjunk néhány évre előre!

A tartomány NETBIOS neve *(ami a régebbi klienseknek kellhet)* alapértelmezésben a tartomány DNS nevének első részéből készül, ebből a szempontból tehát az altartományként való létrehozás nem jelent változást; arról nem is beszélve, hogy mint az az ábrán is látható, a NETBIOS tartománynév az igazi tartománynévtől teljesen függetlenül is megadható.

A telepítés további menete

A tartománynév kiizzadása után meg kell adnunk a Directory Services Restore Mode rendszergazdai jelszavát *(lásd Windows 2000 Server alapismeretek: a címtár visszaállításához használatos boot üzemmódhoz szükséges bejelentkezési jelszó)*. Ezután áttekinthetjük a telepítés ezen szakaszának összefoglalóját *(hálózati és tartományi konfiguráció, terminálszolgáltatás telepítése és hálózati azonosító adatok beállítása)*, majd gombnyomásra megkezdődik a Nagy Reszelés *(mindenféle perverz felhang nélkül értendő!)* első szakasza.

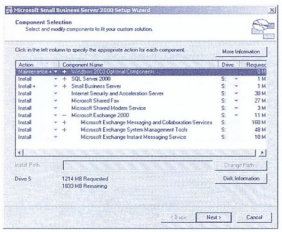


Az SBS telepítésének első fele: a kiszolgáló beállítása. Mint az ábrán is látható, „titokban" még a DCPROMO is lefut...

A telepítés befejezése után a kiszolgáló újraindítása következik. Az újraindulás után a számítógép már mint tartományvezérlő működik, és kezdődhet a valódi SBS komponensek telepítése.

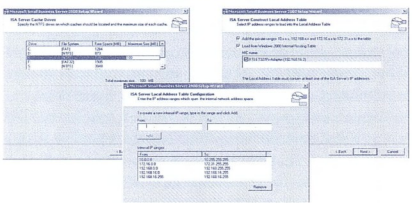
A telepítendő komponensek kiválasztása

A számítógép újraindulása után – ha kell, segítsünk a bejelentkezéskor – ismét elindul az SBS telepítő varázsló, és eljutunk a komponensek kiválasztására szolgáló képernyőhöz. Tüzetesen vegyük szemügyre a tartalmát:



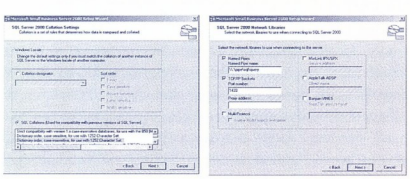
A telepítendő SBS komponensek kiválasztása

Vegyük észre az Install és a Maintenance felirat melletti kis + jeleket! Ez azt jelenti, hogy a csoport alatt további beállítások találhatók *(nyissuk ki egyenként a csoportokat a nagy + jelekre kattintva)*. Ellenkező esetben könnyen előfordulhat, hogy a főcsoportra kattintva például azt hisszük, hogy telepítettük az SQL Server-t, miközben összesen a dokumentációt sikerült felmásolnunk a kiszolgálóra. Az egyes komponensek kiválasztásakor az ablak alsó részében látható a telepítési útvonal, ami szükség esetén megváltoztatható. A továbbkattintás után a kiválasztott komponensek alapvető beállításait kell megadnunk.



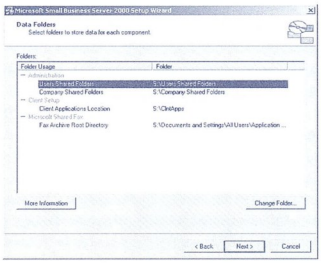
Az ISA Server beállításai

Ha például az ISA Server-t kiválasztottuk a telepítendő komponensek listájából, most meg kell adnunk, hogy mekkora gyorsítótárat szeretnénk, és hogy hova akarjuk azt helyezni. A gyorsítótár csak NTFS partícióra kerülhet, és az alapértelmezett 100 MB kezdésnek megfelelő lesz. Ezek az értékek a későbbiek során természetesen módosíthatók. Az ISA Server másik fontos beállítandó paramétere a helyi IP címeket tartalmazó táblázat, a LAT *(Local Address Table)*. Az ISA ennek alapján tudja ugyanis, hogy ki az, aki a tűzfalon belül, illetve kívül tartózkodik *(függetlenül attól, hogy melyik hálózati kártyán csatlakozik a géphez)*. A LAT összeállításában segít a varázsló: mielőtt kézzel szerkesztgetnénk a címtartományokat, kiválaszthatjuk, hogy szeretnénk-e a privát címeket, illetve a kiválasztott belső hálózati kártyánkról leolvasható alhálózat címeit felvenni a LAT-ba.



Az SQL Server beállításai. Ha ezekkel az ablakokkal a telepítés során nem találkozunk, kezdjünk gyanakodni!

Az SQL Server kiválasztása nem könnyű feladat, ugyanis a telepítő az SQL Server gyökerének kiválasztása során hajlamos az alatta található dolgokat üresen hagyni. Ennek az az eredménye, hogy SQL Server nélkül települ fel az SBS. Bár az SQL Server-t később is hozzáadhatjuk, de az külön macera – figyeljünk inkább oda, ne hagyjuk magunkat becsapni. Az SQL Server előzetes konfigurációja 4-5 dialógusablakon keresztül vezet *(ebből kettő látható fentebb)*, ezek hiányát nehéz nem észrevenni. Az ábra bal oldalán található dialógusablak az adatbázis alapértelmezett kódtáblájának kiválasztására szolgál *(hogy itt mit állítunk be, az függ az SQL Server későbbi felhasználásának módjától)*; a jobb oldalon pedig az SQL Server eléréséhez használható metódusokat állíthatjuk be *(például a TCP portot)*. Ezután még meg kell adnunk az SQL szolgáltatás futtatásához használt felhasználói fiókot és az alapértelmezett adatbázis helyét is.



Az alapértelmezett mappák helye

A telepítő utolsó érdemleges kérdése az alapértelmezett mappák helyére vonatkozik:

- User Shared Folders: a felhasználók saját mappáinak gyökérkönyvtára
- Company Shared Folders: a felhasználók *(illetve a cég)* közös mappája
- Client Applications Location: a kliensalkalmazások telepítőkészleteinek helye
- Fax Archive Root Directory: a faxszolgáltatás archívumának gyökérkönyvtára

Az adatok megadása után elkezdődik a mintegy másfél órás telepítés, aminek során sorra adagoljuk majd a CD-ket a telepítőnek. A telepítés végén – néhány újraindítás után – megjelenik a To Do List feliratú ablak, ami a kiszolgáló első munkába állásakor elvégzendő feladatokat tartalmazza szép sorjában – a következő hónapban innen…

Folytatjuk…

Fülöp Miklós
mick@netacademia.net

A cikkben szereplő URL-ek:
[1] http://www.microsoft.com/sbserver/

Whistler Server Beta 2

Whistler Server Beta 2 újdonságok

Az áprilisi tech.net számban Fóti Marcell tollából már olvashattunk némi előzetest a Whistler-ről. Most szintén csak szemezgetős jelleggel mutatnék be néhány újdonságot – ne feledjük, a termék még csak Beta 2 állapotban van, a végleges funkciókészlet még nincs rögzítve, ezért előfordulhat, hogy egy bétában meglévő funkció nem lesz benne a végleges termékben *(ezt már a Windows 2000-nél is tapasztalhattuk)*.

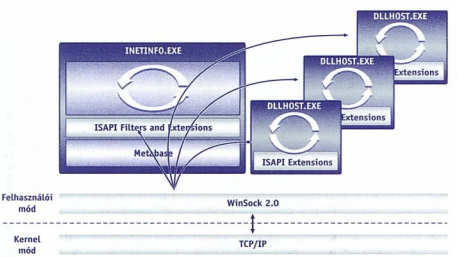
A munkaállomás család végleges neve Windows XP, erre hivatalos megjelenési dátumot is bejelentett már a Microsoft: 2001. október 25. *(a kód természetesen már hamarabb, augusztus végén lezárul – október végére lehet várni a magyar nyelvű változat elkészültét is)*.

A kiszolgálócsalád ennél később fog megjelenni, egyelőre nincs hivatalos dátum. Bár a Microsoft április végén egy sajtóközleményben bejelentette a Windows 2002 Server nevet, ezt később visszavonta – egyelőre tehát nincs végleges döntés, így csak kódnéven, Whistler Server-ként hívjuk a terméket.

Internet Information Service 6.0

A Whistler Server az IIS új, 6.0-ás verziójával jelenik meg, amely jelentős architekturális változásokat tartalmaz.

Az IIS 5.0 alatt az alkalmazások készítésénél döntenünk kellett az „in-process" és az „out-of-process" *(OOP)* futtatás között.



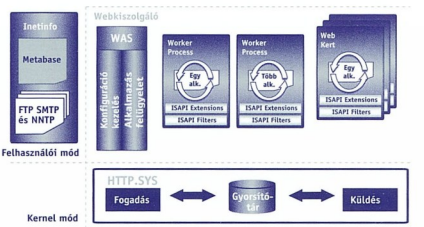
Az IIS 5.0 architektúrája

Az in-processz kifejezés azt jelenti, hogy az alkalmazás egy szálként *(thread)* az INETINFO.EXE processz memóriaterületén belül fut. Ennek előnye az, ami egyébként a szálak használatának az előnye: a szálak között gyors a kommunikáció és a környezetváltás *(context switching)*. Hátránya viszont, hogy az alkalmazás hibája a teljes webkiszolgáló processzt magával ránthatja.

Épp ezért a Microsoft is az OOP futtatási módot ajánlja, azonban sokan mégis in-processz írják az alkalmazásaikat, mert az OOP-nél csökken a teljesítmény, a felhasználói módú INETINFO és az egyes alkalmazások közötti inter-processz kommunikáció és az egyes processzek közötti környezetváltások miatt.

IIS 6.0 – Dedikált alkalmazásmód

Az IIS 6.0 ún. dedikált alkalmazásmódot használ, ahol a felhasználói webalkalmazások kódja teljesen izolált környezetben fut, az IIS 5.0 out-of-process futtatási módjának stabilitásával, de annak teljesítménykorlátai nélkül.



IIS 6.0 architektúra - dedikált alkalmazásmódban

Az IIS 6.0 új architektúrájában az alapvető webkiszolgáló funkciók, a kérések, a gyorsítótár, a konfiguráció kezelése és a webalkalmazások felügyelete teljesen elkülönül a külső, felhasználó által futtatott webalkalmazásoktól. Ezek a központi webkiszolgáló feladatok az új HTTP.SYS kernelmódú meghajtóban és az új Web Administration Service *(WAS)* felügyeleti modulban valósulnak meg. A külső, felhasználói alkalmazások egy ettől teljesen elszigetelt *(OOP)* futtató környezetet kapnak, egy-egy saját mini-webkiszolgálóval (dllhost, ISAPI szűrők, ISAPI bővítmények) együtt – egy ilyen elkülönített webalkalmazás neve „worker process". Az izolált mini-webszerver környezetben az egyes alkalmazások egymástól függetlenül leállíthatók, újraindíthatók, debugolhatók – az egyes worker processzek a LocalSystem fióknál alacsonyabb jogosultsági szinttel rendelkező fiók nevében is futtathatók, ami tovább növeli a megbízhatóságot. A webalkalmazások ún. alkalmazás pool-okba csoportosíthatók, mely az IIS 6.0 felügyeleti egysége: egy pool tartalma lehet egyetlen worker processz is, lehet webalkalmazások csoportja, vagy akár több webhely *(több névtér!)* is.

IIS 6.0 - Kernelmódú webgyorsítótár

A változás középpontja *(mint az a fenti ábrán is látszik)* a kernel módú HTTP.SYS meghajtó. Az IIS 5.0 a beérkező kéréseket az INETINFO-ban, felhasználói módban figyeli – a TCP/IP stack-et pedig a Winsock interfészen keresztül éri el. A Winsock gyors, nincs vele semmi baj, de mégiscsak felhasználói módban van, és egyébként is egy általános hálózati interfész. Az IIS 6.0-ban a HTTP.SYS közvetlenül az IP stack tetején csücsül, a beállított IP/port kombinációkat figyelve fogadja a kapcsolatfelvételi kéréseket, Winsock nélkül. Ennek előnye az, hogy a bejövő kérés nem a korábbi IP stack – Winsock – INETINFO – webalkalmazás utat jár-

ja be, hanem a jóval rövidebb IP stack – HTTP.SYS – webalkalmazás utat *(ami egy processzel, és két felhasználói módú környezetváltással rövidebb)*.
A HTTP.SYS az egyes webalkalmazásokhoz érkező kéréseket sorba állítja, minden alkalmazás pool-hoz önálló puffert kezel, melyek hossza alkalmazásonként korlátozható – ha a beérkező, sorba állított kérések száma a pufferben eléri a maximumot az adott alkalmazáshoz, HTTP 503 hibaüzenetet küld vissza a HTTP.SYS – de csak az adott alkalmazás pool-hoz érkező kérésekre.
Az alkalmazás pool-ok függetlenítése tehát már a kérések kezelésénél megtörténik. A kérések kezelésének központosításával a sávszélesség korlátozása is jobban megoldható: a HTTP.SYS párhuzamosan tudja kiszolgálni a kéréseket a pool-okra meghatározott sávszélességkorlátokkal, szemben az IIS 5.0 soros megoldásával *(egy kérés innen, kettő onnan...)*.
A HTTP.SYS másik nagy előnye a kernel módú webgyorsítótár kialakítása a HTTP válaszokhoz, hiszen találat esetén egy választ közvetlen kernelmódból, környezetváltás nélkül, azonnal visszaadhat *(ez magyarul egy kb. kétszeres szorzó a teljesítményben)*. Ez a gyorsítótár nemcsak statikus, hanem dinamikus válaszokra *(lapokra)* is vonatkozik! Az újdonság az ún. Universal Resource Identifier *(URI)* HTTP response cache módszer. Ennek lényege, hogy a HTTP.SYS nem kényszerít egyetlen központi gyorsítótár szabályrendszert az összes alkalmazásra – az egyes alkalmazás pool-ok mondhatják meg önállóan, külön-külön *(programozható felületen keresztül)* a HTTP.SYS-nek, hogy a saját HTTP válaszaikra milyen gyorsítótár kezelést használjon *(ez nyilván főleg a dinamikus válaszok kezelésénél fontos)*.

IIS 6.0 – Webalkalmazás felügyelet

A Web Administration Service *(WAS)* egy új monitorozó funkció az IIS 6.0-ban, mely a webkiszolgáló részeként fut, külön processzként, felhasználói módban, szintén elkülönítve a külső webalkalmazásoktól. Dedikált módban a WAS feladata rendszeres időközönként „megpingetni" az egyes worker processzeket. A „ping" itt nem IP alapú ICMP echo üzenet, hanem egy processzek közötti kommunikációs csatornán életjel lekérdezése az egyes worker processzektől. Ha egy worker processz nem válaszol egy ilyen oldalbalökésre, a WAS terminálja, és egy új példányt indít belőle. Mivel a kérések kezelése a HTTP.SYS-ben, és nem a worker processzben *(tehát magában a webalkalmazásban)* történik, a HTTP.SYS a processz terminálása után is fogadja az adott alkalmazáshoz érkező kéréseket, puffereli, majd amikor a WAS elindított a hibás processz helyett egy új példányt, annak adja tovább őket. A böngésző előtt ülő felhasználó tehát semmit nem lát abból, ha egy webalkalmazás meghal, csak kicsit később kapja meg a választ *(de nem kap HTTP 503-at)*.
Térjünk vissza a WAS-hoz: ez a szervíz nemcsak monitorozza a worker processzeket, hanem az életciklusukat is képes felügyelni: a Demand Start és az Idle time-out funkciókkal lehetővé teszi azt is, hogy egy nagy webalkalmazás egy-egy rész-processze csak akkor foglaljon erőforrásokat, ha azt valóban használják.
Demand Start használatakor az adott worker processzt csak akkor hozza létre a webkiszolgáló, amikor a legelső kérés beesik a hozzá tartozó sorba a HTTP.SYS-ben. A WAS-ban minden egyes worker processzre meghatározható egy időtartam – ha ezen ideig nem használja senki a worker processzt *(idle állapotban van)*, a WAS automatikusan terminálja azt. Ha ezután érkezik HTTP kérés hozzá, újra létre fog jönni egy példány *(de közben nem foglalta a memóriát!)*.

IIS 6.0 – Webkertek

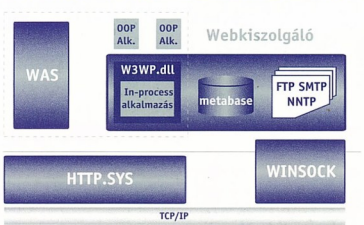
A Windows 2000-nél már meg kellett ismerkednünk néhány mezőgazdasági fogalommal, mint pl. a kiszolgálófarmok és fürtök. A Whistler-beli IIS 6.0 ezt tovább bővíti a Webkertek *(Web Gardens)* bevezetésével. A Web Garden egy olyan alkalmazás pool, amelyben több, de teljesen azonos funkciójú worker processz fut, melyek bármelyike kiszolgálhatja a pool-hoz érkező kérést. Ez tehát egyfajta „terheléselosztást" valósít meg processz szinten – jelentősége főleg többprocesszoros rendszereknél van: ha az egyik CPU-n zárolunk egy worker processzt *(egy ASP-lock miatt)*, akkor a másik CPU-n az ugyanazon alkalmazás poolhoz tartozó másik worker processz még kiszolgálhat kéréseket, nem kell megvárni a zárolás végét. Mindez jelentős teljesítménynövekedéssel jár.

IIS 6.0 - Metabase és konfigurációkezelés

Az IIS 6.0 lehetővé teszi kiszolgálófüggetlen metabase mentések, webhely- és alkalmazáskonfigurációs mentések készítését. A jelszóval titkosított mentéseket egy másik webkiszolgálón helyreállíthatjuk, így könnyebben migrálhatjuk és helyezhetjük át az egyes alkalmazásokat, vagy webhelyeket más kiszolgálóra. A metabase konfigurációs állomány természetesen egy XML fájl *(metabase.xml)*, amit akár notepad-del is szerkeszthetünk, a konfigurációk exportálása pedig szintén XML-be történik. Az új mentési, import és export műveletek új Admin Base Object API-n keresztül érhetők el, de ADSI-n és WMI-n keresztül szintén hozzáférhetők. Az IIS WMI provider szintén új az IIS 6.0-ban, segítségével minden olyan művelet elvégezhető WMI-n keresztül is, ami az IIS ADSI provider-en keresztül megtehető.
Újdonság, hogy a metabase-t az IIS futása közben szerkeszthetjük, nem kell a szervízt leállítani. A metabase változtatásaira az IIS 6.0 verziókövetést használ, egy metabase history mappát tart fenn, ahol a korábbi metabase.xml állományok verziószámmal vannak ellátva *(á la VMS)* melynek segítségével könnyen visszaállhatunk egy korábbi konfigurációra, ha valamit elrontottunk volna.

IIS 6.0 - Standard alkalmazásmód

A dedikált alkalmazásmód jelentősen növeli az alkalmazások teljesítményét és stabilitását – kompatibilitási okokból azonban az IIS 6.0 tartalmaz egy standard alkalmazásmódot is, amely teljesen megegyezik az IIS 5.0 futtatási módjával.



IIS 6.0 standard (kompatibilitási) alkalmazás mód

Ha egy IIS 5.0-ra már megírt webalkalmazásunk nem fut dedikált mód alatt, akkor is érdemes áttérnünk IIS 6.0-ra, mert a standard módú alkalmazásoknál is kihasználhatjuk a kernelmódú HTTP.SYS új lehetőségeit a kérések központosítására és a kernelmódú gyorsítótár kezelésére.
CGI-alkalmazásokat is érdemes áttenni IIS 6.0 alá: az IIS 5.0 szinkronban tud csak CGI alkalmazásokat futtatni *(a CGI folyamatot létrehozó szálat blokkolja az IIS 5.0, amíg a gyermek CGI folyamat vissza nem tér)*, míg az IIS 6.0-ban aszinkron a CGI végrehajtás, párhuzamosan futnak a folyamatok a szülő blokkolása nélkül.

Scalable Web Cache – SWC 3.0

Egy jó hír: a kernelmódú web gyorsítótár előnyeit már ma kihasználhatjuk, nem kell a Whistler Server-re várnunk! A Scalable Web Cache *(SWC 3.0)* az IIS 5.0-hoz *(Windows 2000 + SP1)* ingyenesen letölthető kernelmódú meghajtó **[1]**. Az SWC 3.0 *(hogy egy kollégám kedvenc kifejezésével éljek)* drámaian megnöveli az IIS 5.0 teljesítményét. A webkiszolgálóknál ipari szabványnak tekintett SPECWeb99 teljesítménytesztben épp a napokban hitelesítették egy 8 utas Windows 2000 Datacenter eredményét IIS 5.0-val és SWC 3.0-val. Ez az eredmény abszolút értékben harmadik, és ezzel az összes Linux-os eredményt maga mögé szorította *(még a szintén kernelmódú gyorsítótárat használó Red Hat TUX webkiszolgálókat is)*. A dobogós helyezettek most:

Cég	Webkiszolgáló	CPU#	Eredmény
Sun	Iplanet WebServer 6.0	12	8739
IBM	Zeus 3.3.7	12	8344
MSFT	IIS 5.0 + SWC 3.0	8	8001

A részletes eredmények a **[2]** címen tekinthetők meg. A táblázatból látszik, hogy az első két helyezett 12 processzoros konfiguráció – viszont másfélszer annyi processzorral nem hoznak másfélszer jobb eredményt, mint a harmadik helyezett 8 utas IIS 5.0. Ezek szerint egy 12 vagy 16 utas Windows 2000 Datacenter rendszernek elég jó esélyre lehet az első helyre *(természetesen egy gépen belül ő se fog lineárisan skálázódni, de a fentieknél valószínűleg jobb eredményt hoz majd)*. És ez még mindig az IIS 5.0, nem pedig a Whistler...

Active Directory újdonságok

Azt már a Tech.Net magazin levelezési listáin is megszokhattuk, hogy az Active Directory gondokra mindig ugyanaz a három válasz vonatkozik: DNS, DNS, DNS. A Whistler-ben találunk majd egy eszközt, amellyel ellenőrizhetjük és tesztelhetjük, hogy DNS-ünk megfelelően van-e konfigurálva az adott AD struktúránkhoz, és amely segít felderíteni a hibás beállításokat. Telephelyi tartománykiszolgálók telepítésénél jelent segítséget, hogy a kezdeti replikáció nemcsak hálózatról, hanem bármilyen médiáról *(szalag, CD, merevlemez)* is elvégezhető – így nem kell a lassú WAN vonalon fél napon át replikálni, illetve nem kell fizikailag elszállítani a telephelyi gépet a központba a telepítéshez.
Az AD talán legnagyobb újdonsága viszont a *(tessék megkapaszkodni!)*:

Forest trust

A forest trust-ot két erdő gyökér *(root)* tartománya között kell egyszer definiálni, és ezáltal egy tranzitív megbízotti kapcsolat jön létre a két erdő összes tartománya között. A forest trust egyirányú, és csak a megbízotti kapcsolat két oldalán álló két erdő tartományai szintjén tranzitív, az erdők szintjén nem. Ha tehát A erdő egy forest trust létrehozásával megbízik B erdőben, míg B erdő megbízik C erdőben, akkor A erdő nem fog megbízni implicite C erdőben, csak ha külön A és C között is létrehozunk egy forest trust kapcsolatot *(azt hihettük, hogy az AD kétirányú tranzitív trust kapcsolataival végre nem kell mindig átgondolni, hogy mit jelent megbízó és a megbízott félnek lenni – ez a házi feladat most visszajön az egyirányú forest trust-nál)*.
Az erdők közötti hálózati és interaktív bejelentkezés NTLM és Kerberos felhasználói azonosítással is működik, illetve működik a Kerberos megszemélyesítés is, ha egy többrétegű alkalmazás valamelyik rétege egy másik erdőben van.
Forest trust kapcsolatot nem lehet egymást átfedő névtérrel rendelkező erdők között létrehozni. Az erdők közötti névfeloldáshoz a megbízotti kapcsolat kialakításakor létrejön egy ún. Trusted Namespace *(Megbízott Névtér)*, ami tartalmazza az összes megbízott erdő tartomány-, felhasználó-, szolgáltatásneveit és a felhasználói SID-eket. Ha már az erdő Globális Katalógusa sem tud feloldani egy nevet, akkor egy új eljárást hív meg a Whistler, ami a fenti Megbízott Névtérben próbálja megkeresni az adott nevet.
A hozzáférésvezérlési listákon *(ACL-eken)* szintén szerepelhetnek objektumok a megbízott erdőkből is – az objektumok teljes nevét meg kell azonban adni, más erdő objektumait nem tudjuk kilistázni és nem használhatunk joker karaktereket sem a név megadáshoz *(a Megbízott Névtér ugyanis csak névfeloldást végez, katalógusszolgáltatást nem)*.
Cégek egyesítésénél, felvásárlásnál nagyon jól jön, de a többerdős modell továbbra sem célszerű tervezési szempont.

LDAP újdonságok

A Whistler LDAP fejlesztéseket figyelembe véve méginkább igaz a fenti állítás. Több erdő kialakításához eddig a legütősebb érv a különböző séma iránti szükséglet volt, amit egy erdőn belül semmiképp nem lehetett megoldani. A Whistler viszont támogatja az AD-ban a Dinamikus Külső Osztályokat, ahol a külső osztály által meghatározott új objektumtulajdonságok egyedi objektumpéldányokhoz rendelhetők hozzá – azaz meghatározhatjuk azon objektumok halmazát egy erdőn belül, ahol látni és használni szeretnénk a külső osztály tulajdonságait – a többi objektumnál ezek a tulajdonságok nem fognak látszani. A Windows 2000-nél ezt nem tudtuk megtenni – új tulajdonságot csak a séma bővítésével lehetett megadni, ahol a sémában az osztály definíciója módosult, és bővült ki új tulajdonságokkal, és így egy osztály összes példányára érvényes lett *(ezért statikus ez a módszer)*. A Dinamikus Külső Osztályok nem nyúlnak a sémához, csak úgy „röptében" rendelődnek hozzá bizonyos objektumpéldányokhoz. *(A következő lépés csoportházirendekkel szabályozni, hogy egy adott szervezeti egységen belül milyen objektumokat milyen Dinamikus Külső Osztályok bővítsenek ki – lehet, hogy ez is benne van a Whistler-ben, nem tudom, nem néztem...)*

A Dinamikus Külső Osztályokat *(Dynamic Auxillary Classes)* ne keverjük az ún. Dinamikus Bejegyzésekkel *(Dynamic Entries)* ami szintén új a Whistlerben, de egyébként az IETF RFC 2589 megvalósítása: egy dinamikus bejegyzéshez egy TTL-t rendelhetünk, aminek lejárta után a bejegyzés automatikusan törlődik a címtárból. *(Ha rosszmájú akarok lenni, akkor egy User objektum is lehet dinamikus, a TTL-jét a munkaszerződésének időtartamára állíthatjuk...)*

Az LDAP kapcsolatok TLS *(Transport Layer Security)* használatával titkosíthatók *(RFC 2830)*.

Virtual List View *(VLV)*: ezt néhányan félreértelmezték már, pedig ez is egy IETF által definált LDAP kiterjesztés – ha egy LDAP ügyfél nem akarja lehozni a kiszolgálóról egy LDAP lekérdezés teljes eredményét, mert az túl sok adat lenne, akkor a VLV segítségével „ablakozhat" az eredményben, kisebb részeket hozva csak át a hálózaton.

Talán mindannyian hallottuk már azokat a vádakat, miszerint az Active Directory LDAP megvalósítása nem szabványos. Ez főleg a Novell és a Netscape vesszőparipája volt, mondván, hogy az AD nem az inetOrgPerson osztályból származtatja a User objektumot, amint az az RFC 2798-ban írva vagyon.

Az igazságot mindannyian tudjuk: az RFC 2798 még csak draft állapotban volt a Windows 2000 elkészültekor, és egyébként sem tette kötelezővé az inetOrgPerson-ból való származtatást. A Windows 2000 ezért a már régen elfogadott és érvényben levő RFC 2254 szerint az OrganizationalPerson osztályból származtatta a felhasználókat. Ennyit arról, hogy ki a szabványos. A Whistler az alap AD sémában támogatja az inetOrgPerson-ból származtatott felhasználókat is, minden megszokott művelet elvégezhető rajtuk, grafikus felületről is, LDAP-on és ADSI-n keresztül is. *(Ez a „Na jó, legyen igazatok, csak hagyjatok már békén..." tipikus esete...)*

Csoportházirendek

160 új csoportházirendet tartalmaz a Whistler Beta 2 a Windows 2000-hez képest. Ez csak a legkisebb újdonság a csoportházirendek terén: talán a leginkább használható fejlesztés az ún RSoP *(Resultant Set of Policies, azaz Eredő Csoportházirend)*. Ez egy eszköz, mely a FullArmor cég által gyártott FAZAM2000 termékhez hasonlít, melynek korlátozott változatával a Windows 2000 SP1-en, funkciókészletével pedig e szám GPO cikkében találkozhatunk *(tessenek odalapozni!)*. Segítségével grafikus felületen elemezhetjük, hogy egy adott felhasználóra és egy adott gépre különböző szinteken előírt házirendek, öröklődéssel, blokkolással és minden egyéb furmánnyal megtűzdelve végül is milyen eredő beállítást fognak eredményezni. Az eszköz a csoportházirend beállítások megtervezésében is segít, hogy végül minden szinten, minden objektumra azt a hatást érjük el, amit szeretnénk.

Az egyes új beállítások közül érdekes a Netlogon szolgáltatás összes beállításának szabályozása a tartományvezérlőkön csoportházirenddel: dinamikus DNS regisztráció, telephelyfelderítés, stb., de ugyanígy szabályozható a terminálszolgáltatás minden aspektusa is. Na és most tessék figyelni: WMI szűrés csoportházirend objektumokon! Bizony! Hasonlóan, ahogy egy GPO-t biztonsági csoportokra tudunk szűrni a Jogosultságok fülön, egy új fül is meg fog jelenni, ahol WMI lekérdezések eredményeire szűrhetünk. Meg tudjuk pl. adni, hogy egy házirend csak arra a gépre legyen érvényes, amiben 100 Mb-es hálókártya van, vagy csak arra amiben hálókártya van... *(emlékeznek, mit írtam az áprilisi szám bevezetőjében a felügyeleti technológiák, az SMS és az Active Directory közeledéséről?)*

Terminálszolgáltatás

A legnagyobb újdonság itt az, hogy a munkaállomás, a Windows XP Professional is tartalmazza a beépített terminálszolgáltatást – ezért erről a Windows XP cikkben írunk majd... Mindenesetre említsük meg az átirányítási lehetőségeket *(hang, fájlrendszer, port, nyomtató, vágólap átirányítás, azaz az ügyféloldal ezen erőforrásai a terminál ügyfél munkamenetben (session) is látszanak és használhatók)*. Újdonság a Terminal Services Virtual Channel API, amivel az átirányítási funkciót egyéb erőforrásokra is kibővíthetjük saját alkalmazásunkból.

A hang átirányításnál a hangfolyam automatikusan a rendelkezésre álló sávszélességre lesz optimalizálva, nem kell külön állítgatnunk.

Smart card támogatás

A smart card azonosítás köre kibővül, pl. terminál kiszolgálóra is be tud jelentkeztetni a terminálügyfél jelszó és felhasználónév nélkül, csak a smart card használatával.

A bejelentkezésen kívül számos adminisztrációs eszköz is tudja majd a Rendszergazdát smart card-ról azonosítani, név szerint a grafikus DCPromo, „Run As" és „Map Network Drive", illetve a parancssori „run as" és a net.exe. *(Hogy mik ezek? Lapozzon a „Szerszámosláda" rovathoz! – a szerk.)*

Egyebek

A tervek szerint mind a Whistler Server, mind a Windows XP Personal és Professional tartalmaz majd egy beépített tűzfalat *(Personal Firewall)*, a LAN, VPN és betárcsázásos *(DUN)* kapcsolatok védelmére – és alapszintű behatolásvédelmet is tudni fog *(a portszkennelés elleni védelmet legalábbis)*. Igaz, hogy az ISA Server 2000 a hitelesített, ICSA minősítéssel rendelkező tűzfal, de a tech.net levelezőlistákon is gyakran felbukkan a kérés, hogy egy jó, olcsó *(lehetőleg ingyenes)* tűzfalat keresnének... Ez most az operációs rendszer része lesz. Hot-plug memória: ha a hardver is támogatja, a Whistler Server-ben működés közben bővíthetjük a memóriát, ami azt be is konfigurálja, az alkalmazások és az operációs rendszer újraindítás nélkül azonnal használhatják.

Horváth Tamás
Microsoft Magyarország
tamash@microsoft.com

A cikkben szereplő URL-ek:
[1] http://www.microsoft.com/TechNet/iis/swc3.asp
[2] http://www.spec.org/osg/web99/results/res2001q2

A csoportos házirend *(II. rész)*

Programtelepítés Group Policyval

Az előző részben eljutottunk a programtelepítés küszöbéig, ebben a hónapban ott folytatjuk, ahol májusban félbe hagytuk: ha nincs új típusú MSI fájl a telepítendő alkalmazáshoz, akkor a Veritas cég WinInstall szoftverével készíthetünk egyet...

WinInstall LE használata

A program egyszerűsített változata *(Light Edition)* megtalálható a Windows 2000 Server CD VALUEADD\3RDPARTY\MGMT\WINSTLE mappájában. Az alkalmazást új csomagok létrehozása mellett már meglévők módosítására is használhatjuk. A WinInstall LE működési alapelve a telepített alkalmazás által végzett módosítások *(fájl, registry stb.)* összegyűjtése. Ehhez „lefotózzuk" az operációs rendszer állapotát a telepítés előtt, majd elvégezve a telepítést a „fotó" alapján megállapíthatók a változások.

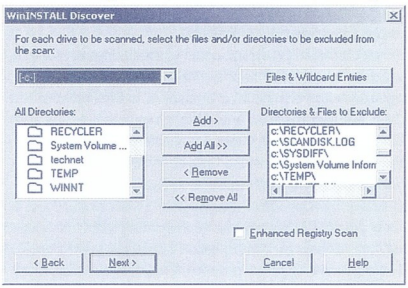
Új telepítési csomag elkészítéséhez két számítógépre van szükségünk. Az egyikre próbaként telepítsük fel azt az alkalmazást, amelyiket szeretnénk elterjeszteni a vállalatunknál. Fontos, hogy előzetesen erre a gépre az operációs rendszeren és a megfelelő szervizcsomagon kívül mást ne tegyünk fel. Így biztosítható ugyanis, hogy a próbatelepítés során a fájlrendszeren és a regisztrációs adatbázisban bekövetkező összes változást *(és csak azt!)* regisztrálni tudjuk. A másik számítógépre pedig a WinInstall LE-t telepítsük.

Mielőtt a referenciául szolgáló számítógépen elvégeznénk a próbatelepítést, le kell futtatnunk a WinInstall Discover *(DISCOZ.EXE)* elnevezésű segédprogramját *(before snapshot)* az alapállapot „lefotózásához".

A Discover elindítását követően először meg kell adnunk az elterjesztésre váró alkalmazás nevét, illetve az elkészítésre váró .msi fájl elnevezését és elérési útját. *(Ha nem adunk meg elérési utat, akkor a csomag a WinInstall könyvtárában jön létre.)* Ezt követően a Discover munkakönyvtárának meghajtóját kell kijelölnünk. A program – átmenetileg – itt hozza létre a DISCOVER.WRK nevű könyvtárat, amelybe a működéshez szükséges fájlok kerülnek. A teljesítmény növelése miatt lokális meghajtót válasszunk. Gondolnunk kell arra is, hogy a kiszemelt partíción legalább 250 MB szabad hely legyen.

A Next gomb megnyomása után feltűnő párbeszédablakon azt a meghajtót, illetve azokat a meghajtókat kell kijelölnünk, ahol a próbatelepítés során változások következhetnek be.

Végezetül lehetőségünk van olyan mappákat kivonni az ellenőrzés alól, amelyeknél biztosan nem várható módosulás.

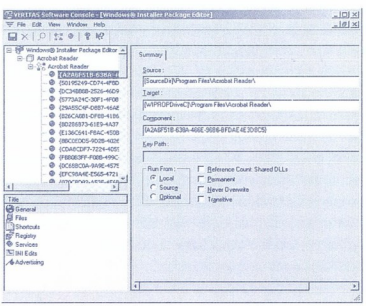


A hatékonyság növelése végett fájlokat és könyvtárakat vonhatunk ki az ellenőrzés alól

Miután a Discover feltérképezte az aktuális konfigurációt, következhet az elterjesztésre váró alkalmazás telepítése. A folyamat befejezéseként ismét le kell futtatnunk a Discover segédprogramot *(after snapshot)*. Ennek során a program rögzíti mindazokat a változásokat, amelyek a próbatelepítés során létrejöttek. A csomag és az alkalmazás elterjesztéséhez szükséges állományok abban a könyvtárban lelhetők fel, amelyet a Discover elindítása után adtunk meg az .msi fájl tárolóhelyeként.

Csomagok módosítása WinInstall Console-lal

Sok esetben előfordul, hogy a telepített programon utólag több beállítást is módosítani kell, hogy illeszkedjen a vállalatnál elterjedt konfigurációhoz. Abban az esetben, ha a telepítés számos munkaállomást érint, fontos szempont, hogy ezeket a változtatásokat automatizálni tudjuk. Szerencsére a Windows Installer csomagokban tárolt információkat könnyen módosíthatjuk a VERITAS Software Console jóvoltából.



A Windows Installer csomag tartalma a VERITAS Console-on keresztül szemlélve

Ezzel a módszerrel nagyon hatékonyan tudjuk a programtelepítést testre szabni:

- megváltoztathatjuk az összetevőket
- lehetőségünk van, hogy a programmal együtt új betűtípust is telepítsünk
- módosíthatjuk a telepítendő szervizek beállításait
- speciális jogosultságot rendelhetünk fájlokhoz
- megszabhatjuk, hogy a programhoz tartozó parancsikonok megjelenjenek-e a Start menüben, illetve a Munkaasztalon
- módunkban áll kulcsok és bejegyzések létrehozására és törlésére a regisztrációs adatbázisban

Bár az iménti felsorolás nem teljes, talán jól érzékelteti a csomagok módosításában rejlő lehetőségeket.

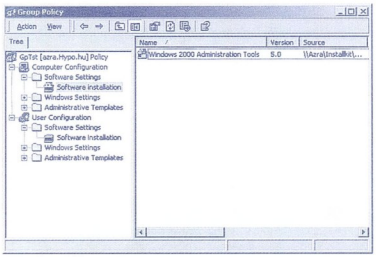
A VERITAS WinInstall LE teljes ismertetése túlnő ezen cikk keretein. A program működésével kapcsolatos leírás megtalálható a Microsoft honlapján **[1]**.

A csoportos házirend programtelepítési modulja

A Group Policy snap-in Computer Configuration\Software Settings, valamint a User Configuration\Software Settings mappájában található a programok telepítésére szolgáló modul. E két modul esetében különböző telepítési eljárásokat engedélyezhetünk:

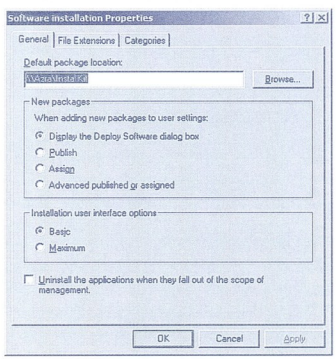
- Publish to Users: Lehetőségünk van a telepítést a felhasználók számára közzétenni. Ebben az esetben a felhasználónak kell gondoskodnia a telepítés elindításáról a Control Panel → Add/Remove Programs → Add New Program segítségével. Ha olyan házirendobjektumnál engedélyezzük a szoftvertelepítést, amely már érvényben van a felhasználóknál, akkor nem kell újra bejelentkezni a telepítés indításához. Fontos tudni, hogy a felhasználók bármikor újratelepíthetik az alkalmazást mindaddig, míg ezt le nem tiltjuk a Group Policy snapinben. Sőt, lehetőségük van a program eltávolítására is!
- Assign to Users: Az előző módszerhez képest további szolgáltatás is rendelkezésünkre áll: a telepítés a program első használatakor is elindítható. A telepítendő program ikonja ugyanis a felhasználó bejelentkezését követően megjelenik a Start menüben.
- Assign to Computers: A telepítés a számítógép újraindításához kötött, és a felhasználó beavatkozása nélkül megy végbe. Ilyenkor csak a lokális rendszergazdai jogosultsággal rendelkezők tudják a programot eltávolítani.

Első lépésként tehát azt kell eldöntenünk, hogy a három lehetőség közül melyik felel meg legjobban a céljainknak. Ezután hirdethetjük meg telepítésre a Windows Installer csomagot: jelöljük ki a megfelelő szoftvertelepítési modult, majd kattintsunk az Action menüben a New/Package parancsra. Az Open párbeszédablakban keressük meg azt a megosztott mappát, ahova a telepítendő állományokat másoltuk. Lényeges, hogy az elérési utat mindig az univerzális névkonvenciónak *(UNC)* megfelelően adjuk meg. Értelemszerűen ezáltal válik egyértelművé, hogy melyik szerver melyik megosztásáról történjen a telepítés.



A rendszergazdai eszközök telepítésének meghirdetése

Ha mindig ugyanazt a szervert és megosztást használjuk az automatikus telepítésre, akkor érdemes az elérési utat a modul tulajdonságait leíró párbeszédablakon *(Action menü Properties utasítás)* megadni.



A szoftvertelepítés alapértelmezett beállításai

Emellett más hasznos beállítási lehetőségünk is adódik. A New packages szekcióban meghatározhatjuk, hogy melyik legyen az alapértelmezett telepítési mód. Ha meghagyjuk a Display the Deploy Software dialog box beállítást, akkor minden új csomag létrehozásakor egy párbeszédablakon kell meghatároznunk a kívánt telepítési fajtát.

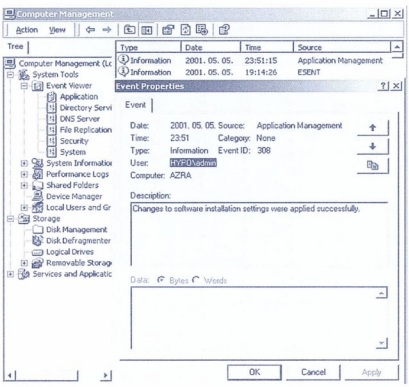
Az Installation user interface options-nál szabályozhatjuk, hogy a telepítés során mennyi tájékoztató üzenet, illetve beavatkozási lehetőséget biztosító párbeszédablak jelenjen meg a felhasználóknak. Alapértelmezésben *(Basic)* csak a progress bar lesz látható. Ha azt szeretnénk, hogy minden telepítési üzenet, párbeszédablak stb. megjelenjen, akkor válasszuk a Maximum rádiógombot.

Gyakori eset, hogy a szoftvertelepítés csak egy meghatározott szervezeti egységhez tartozó felhasználókat érint. Így, ha valamelyik dolgozó átkerül egy másik szervezeti egységbe, akkor a programhasználatra már nem jogosult. A gyakorlatban ez azt jelenti, hogy a szóban forgó munkatárs számítógépéről a szoftvert el kell távolítani. Erre a problémára kínál nagyon elegáns megoldást a Software Installation Properties párbeszédablak alján található beállítás. Az „Uninstall the applications when they fall out of the scope of manage-

ment" jelölőnégyzet engedélyezésekor – ha a csoportos házirendobjektum már nem érvényes a felhasználóra – az operációs rendszer eltávolítja a programot.
A szoftver automatikus eltávolításának nem ez az egyedüli módja. A Group Policy snap-inben meghirdetett csomagok törlésénél *(Action menü → All Tasks → Remove)* két választási lehetőségünk van. Egyrészt megszüntethetjük a csomag meghirdetését, vagyis megakadályozzuk a további telepítést *(„Allow users to continue to use the software, but prevent new installation" rádiógomb)*. Másrészt a csomag eltávolítása mellett utasítást is adhatunk a már telepített program eltávolítására *(„Immediately uninstall the software from users and computers" rádiógomb)*.

A hibakeresés lehetőségei

Az automatikus programtelepítés során előforduló főbb eseményekről bejegyzés készül az Application Event Logba. A telepítésre vonatkozó információk esetén az „Application Management" felirat jelenik meg az Eseménynapló Source oszlopában.



A Group Policy Software Installation bejegyzései az Application Event Logban

Módunkban áll az imént említettnél részletesebb naplózást beállítani *(Verbose Logging)*. A regisztrációs adatbázisban a

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\
  Windows NT\CurrentVersion\Diagnostics
```

kulcs alatt kell létrehoznunk az "Appmgmtdebuglevel" = 9b *(DWORD)* bejegyzést. A részletes diagnosztikai és tájékoztató bejegyzések *(például: melyik házirendobjektum érvényesül a felhasználók és számítógépek esetén, a telepítéssel kapcsolatos esetleges hibakód stb.)* a %windir%\debug\usermode\appmgmt.log-ban találhatók meg.
Még bővebb információhoz juthatunk a regisztrációs adatbázis másik bejegyzése révén:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\
  Windows\Installer\Logging="voicewarmup"
```

Ezt a beállítást a Group Policy révén is érvényre juttathatjuk *(Computer Configuration\Windows Components\Logging)*.
A bejegyzés értékének mindegyik karaktere különböző naplózási módot állít be:

- v: részletes naplózás
- o: a „lemezterület megtelt" üzenet rögzítése
- i: állapotüzenetek
- c: a felhasználói felületen megadott paraméterek
- e: az összes hibaüzenet naplózása
- w: figyelmeztetés a nem végzetes hibákról
- a: a telepítés fő lépéseinek naplózása *(például: parancsikonok elkészítése, komponensek telepítése)*
- r: a telepítés fő lépéseinek részletezése *(például a telepített komponensek felsorolása)*
- m: kevés memória miatti leállás naplózása
- u: User requests
- p: Terminal properties

A karaktereket tetszőleges sorrendben adhatjuk meg. Érdemes gondosan megválasztani a naplózás részletességét, mert a túl sok adat rögzítése értelemszerűen lassítja a programtelepítést.
A Windows Installer működésével kapcsolatos bejegyzések két külön állományba kerülnek. A felhasználó által kezdeményezett események *(például a telepítés indítása az Add/Remove Programs alól)* a %temp%\MSI*.log, míg a központi telepítéssel összefüggő információk *(például szoftvercsomag meghirdetése)* a %windir%\temp\MSI*.log fájlban találhatók.
A Windows 2000 Server Resource Kit is kínál eszközt a hibakeresésre. Az addiag.exe elnevezésű segédprogram az általános diagnosztikai adatok mellett a következő információkat gyűjti össze számunkra:

- a bejelentkezett felhasználó adatai *(felhasználói név, SID, a profil típusa, a bejelentkeztetést végző szerver neve stb.)*
- a telepített, illetve telepítésre meghirdetett programokkal kapcsolatos, a regisztrációs adatbázisból származó információk
- az Active Directoryban fellelhető, a meghirdetett szoftverre vonatkozó információk
- az Eseménynapló megfelelő bejegyzései

Az imént felsorolt módszerek kifejezetten a programtelepítés hibáinak feltárására szolgálnak. A csoportos házirend végrehajtásával összefüggő problémák felderítésére más eszközök is rendelkezésünkre állnak.

A csoportos házirend végrehajtásával kapcsolatos hibák feltárása

A problémák okának megállapításához sok esetben elegendő az Eseménynapló részletes naplózási funkciójának engedélyezése:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\
  CurrentVersion\Diagnostics\
  RunDiagnosticLoggingGroupPolicy=1 (DWORD).
```

Az operációs rendszer nem rögzíti az összes figyelmeztetést és hibaüzenetet az Event Logba. Sokkal több információhoz jutunk, ha az úgynevezett Userenv naplózást alkalmazzuk:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\
  CurrentVersion\Winlogon\
  UserenvDebugLevel=0x10002 (DWORD).
```

A csoportos házirend végrehajtásával összefüggő adatok a %windir%\debug\usermode\Userenv.log elnevezésű fájlban lelhetők fel. Ha az állomány mérete meghaladja az 1 MB-os méretet, akkor a rendszer újraindításakor az adatokat az operációs rendszer átmásolja a userenv.bak-ba, és egy új Userenv.log fájlt hoz létre. Ügyeljünk arra, hogy ha a csoportos házirendben gyakori frissítést állítottunk be, akkor a fájl mérete intenzíven növekszik. A usermode mappán érvényes biztonsági beállítások miatt a naplófájl elolvasásához rendszergazdai jogosultság szükséges.

A házirendobjektum szerkesztésekor előforduló hibák rögzítéséhez a regisztrációs adatbázis Winlogon kulcsa alatt a GPEditDebugLevel=0x10002 *(DWORD)* bejegyzést kell létrehoznunk. A hibaüzenetek lelőhelye a %windir%\debug\usermode\gpedit.log lesz.

A Windows 2000 Resource Kit is több hasznos alkalmazást kínál:

- Group Policy Verification Tool: A gpotool.exe többek között alkalmas a házirendobjektumok állapotának meghatározására, illetve replikációjuk ellenőrzésére.
- Active Directory Replication Monitor: A tartományvezérlőkön elhelyezkedő házirendobjektumok esetében az Active Directory nem megfelelő replikációja is számos hiba forrása lehet. Az AD replikációs hibáinak nyomon követésére nagyon jól használható a replmon.exe.
- Group Policy Results Tool: A gpresult.exe részletes információt szolgáltat a végrehajtott csoportos házirendobjektumokról, tájékoztat a végrehajtásban résztvevő tartományvezérlő nevéről, valamint a regisztrációs adatbázisban végbement változásokról.
- Network Connectivity Tester: Speciális esetben előfordulhat, hogy a házirend végrehajtásának útjában hálózati problémák állnak. Ennek megállapítására jól alkalmazható a netdiag.exe.

FAZAM 2000

A csoportos házirend szolgáltatásainak széles körével nagyban megkönnyíti a rendszergazdai feladatokat. Hatékony kezeléséhez viszont alapos ismeret szükséges, amelynek megszerzése időigényes folyamat. A FullArmor Corporation **[2]** FAZAM 2000 *(FullArmor Zero Management for Windows 2000)* néven olyan eszközt fejlesztett ki, amellyel csökkenthető a csoportos házirend adminisztrációjára fordított idő. A FAZAM 2000 segítségével ugyanis olyan feladatokat is elláthatunk, amelyeket a Group Policy beépített eszközeivel nem. Például:

- a házirendstruktúra grafikus megjelenítése, amely nagy segítséget nyújt a tervezésben és a hibaelhárításban
- házirendobjektumok archiválása és visszaállítása a szervezeti egységekhez fűződő kapcsolatuk figyelembevételével
- egyes rendszergazdai feladatok automatizálása scriptek segítségével
- házirendobjektumok replikálása tartományok és erdők között
- hibakeresés lehetősége távoli munkaállomásokon

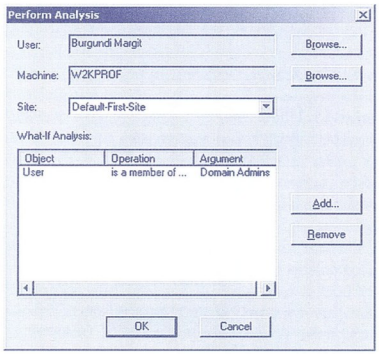
Az imént felsoroltakon kívül a FAZAM 2000 képes elemezni például egy új házirendobjektum bevezetésének várható eredményét:

- az új GPO érvényesülésekor pontosan milyen beállítások lesznek érvényesek valamely felhasználóra
- az új beállítások nem kerülnek-e konfliktusba a korábbiakkal

Az elemzés során három tényezőt tudunk megvizsgálni:

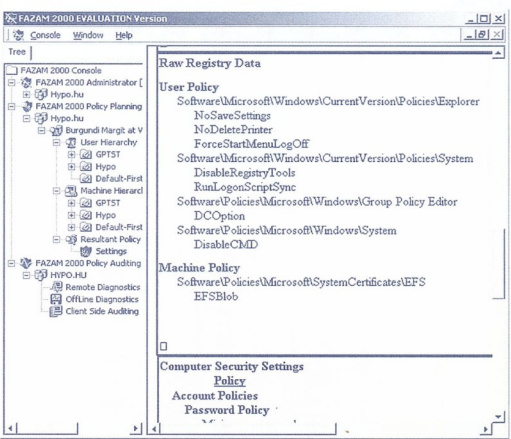
- a felhasználó tagja valamelyik csoportnak
- a felhasználó nem tagja egy bizonyos csoportnak
- a felhasználó átkerül egy másik szervezeti egységbe

A tényezők hatásának vizsgálatakor felhasználói azonosítón kívül mindig meg kell határozni a munkaállomás nevét is.



Mi történik, ha Burgundi Margit a Domain Admins csoport tagjává válik?

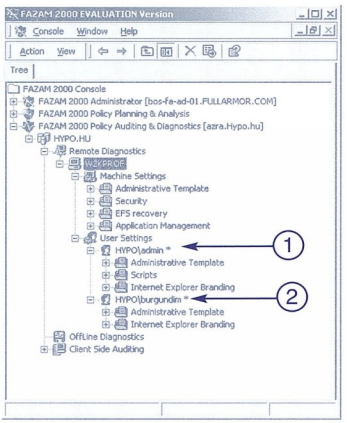
Erre azért van szükség, mert a FAZAM 2000 a lokális házirendbeállításokat is figyelembe veszi. Az elemzés az öröklődési szabályok figyelembevételével készül, így teljes képet kapunk a beállítások várható hatásáról.



Az elemzés végeredményének egy részlete

A FAZAM 2000 nagyon hasznos szolgáltatása, hogy a hibakeresés céljából távoli munkaállomásokon is le tudjuk futtatni az elemzést. Ezt a funkciót csak azután vehetjük

igénybe, hogy a munkaállomásokon telepítettük a FAZAM 2000 kliensoldali komponensét. A vizsgálatra kiszemelt számítógéphez a FAZAM 2000 Policy Auditing & Diagnostics alatt található Remote Diagnostics modul révén kapcsolódhatunk hozzá *(Action menü Select Machine parancs)*. Az elemzés nemcsak a számítógép, de a szóban forgó munkaállomásról bejelentkezett összes felhasználói azonosító beállításait érinti. Így vizsgálatunk tárgyát képezheti például valamelyik futó szervizhez rendelt felhasználói azonosító házirendje is.



A Group Policy beállítások távoli elemzése a W2KPROF nevű munkaállomáson

① A számítógépen dolgozó munkatárs házirendje
② A munkaállomáson futó szervizhez rendelt felhasználói azonosító házirendje

Arra is van lehetőségünk, hogy az elemzést helyben, a problémás számítógépen végezzük el, és az adatokat később értékeljük ki az Offline Diagnostics szolgáltatás segítségével. Az adatgyűjtést a FAZAM 2000 fadiag.exe nevű segédprogramja végzi, amelyet azon a számítógépen kell lefuttatnunk, amelyiken a hibát tapasztaltuk. A programot parancssorból érdemes indítani, mert paraméterként meg kell adnunk egy fájlnevet, amelybe az összegyűjtött információk kerülnek. A FullArmor ajánlása alapján a fájlnak .dgn kiterjesztést adjunk, hogy jól megkülönböztethető legyen egyéb diagnosztikai információkat tartalmazó állományoktól. Az adatok értékeléséhez az Offline Diagnostics modulnál kell megadnunk a .dgn fájl elérési útját *(Action menü Select File parancs)*.
Lényeges különbség a Remote Diagnosticshoz képest, hogy az Offline Diagnostics által szolgáltatott adatok csak a számítógép, illetve a munkaállomáson éppen dolgozó munkatárs házirendjére vonatkoznak. Ezzel a módszerrel tehát nem tudjuk elemezni a szervizhez rendelt felhasználói azonosító beállításait. Mint látható, a FAZAM 2000 a Group Policynál bővebb szolgáltatási körrel rendelkezik. Ennek tükrében különösen fontos hangsúlyozni, hogy a FullArmor célja nem a csoportos házirend kiváltása, hanem egy olyan könnyen kezelhető eszköz megalkotása volt, amely a tervezéstől a hibaelhárításig hatékonyan segíti a rendszergazdák munkáját.
A FAZAM 2000 egyszerűsített változata utólag bekerült a Windows 2000 Resource Kit alkalmazásai közé. A telepítőkészlet letölthető a Microsoft honlapjáról **[3]**.

Tomasz Balázs
balazs.tomasz@hu.hypovereinsbank.com
pszichológus

A cikkben szereplő URL-ek:
[1] http://www.microsoft.com/windows2000/library/ planning/management/veritas.asp
[2] http://www.fullarmor.com
[3] http://www.microsoft.com/windows2000/techinfo/reskit/ tools/existing/fazam2000-o.asp

Runas és Whoami

Ha valaki nem olvasta volna az előző részt *(egy hónap kihagyás történt)*, vagy memóriáját már felszabadította azóta: cikksorozatunkban a Windows 2000-hez kapcsolódó hasznos Microsoft segédprogramokat részletezem. Az alábbi forrásokból emelek ki alkalmanként néhány fontosabb eszközt:

- Windows 2000 Server
- Windows 2000 Support Tools
- Windows 2000 Server Resource Kit

Az adott források elérhetőségét az első részben részletezetem, ebből is látszik hogy érdemes visszamenőleg beszerezni a magazinszámokat :).

Másodlagos bejelentkezés

Elérhetőség: X:\WINNT\SYSTEM32\runas.exe

Forrás: Windows 2000 Server (és Professional)

Ez a kombináltan parancssoros és grafikus, beépített Windows 2000 parancs lehetővé teszi például egy rendszergazda számára, hogy a felhasználó aktuális bejelentkezéséhez tartozó jogosultságoktól eltérő jogosultságokkal futtasson egy eszközt vagy programot anélkül, hogy a júzernek ki kellene jelentkeznie. Ezt a UNIXban az su parancs és testvérei oldják meg. Egy biztonsági jótanács, amelyben a runas sokat segít: mivel sokan a rendszergazdai munkaállomáson egyéb tevékenységet is végzünk *(pl. levelezés, web)*, ajánlott mindezt egy „halandó" felhasználói kontextusban tenni, a különféle biztonsági kockázatok miatt *(trójai és féregvírusok, stb)*. Természetesen fordítva is alkalmazható – mint bármilyen fegyver – a rendszergazda egyszerűen tudja a korlátozott környezeteket finomhangolni a felhasználók számára, mivel szintén futtathatja az adott alkalmazásokat az ő nevükben. Ezekben a helyzetekben használható a runas parancs, mivel a szélesebb jogosultságot igénylő feladatokat is végezhetjük a megszokott felhasználói környezetünkből, felesleges ki-bejelentkezések nélkül.

A parancssorból *(és a parancsikonokból)* futtatható runas az alábbi szintaktikával működik:

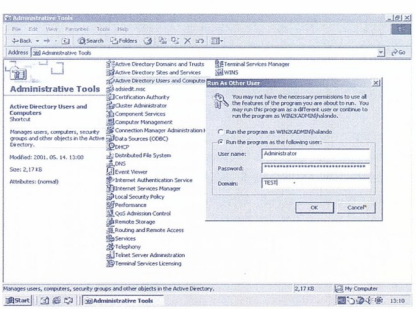
```
runas [/profile] [/env] [/netonly]
/user:Felhasználói_fiók_neve "program"
```

Egy egyszerű példa:

```
runas /user:DOMAIN\Administrator "cmd"
```

Ez a példa a rendszergazda nevében indít egy konzolt. A felhasználónevet két formátumban is megadhatjuk: a hagyományos tartomány\felhasználónév, és a felhasználónév@tartomány *(UPN)* formában. Az **[1]**-es címen található KB cikk leírja azt a bizonyos hibajelenséget, amely a username@domain.stb *(UPN)* formátumú felhasználónévmegadás esetén jöhet elő a runas használatakor SP1-es környezetekben.

A kapcsolók közül a /profile jelzi, hogy ne az eredeti felhasználó profilját értelmezze a rendszer – tehát ez dönti el, hogy az alapértelmezett temp, My Documents mappák közül melyik felhasználóét veszi figyelembe. A /env a környezeti változókkal machinál *(SET TEMP, SET WINDIR stb.)*. Megadhatjuk, hogy az aktuális változókat használja a parancs a felhasználó helyi környezete helyett. Szintúgy a /netonly kapcsoló, amellyel beállíthatjuk, hogy csak távoli kapcsolatokra legyen érvényes a runas – így kisebb az esélye, hogy a lokális felhasználó gépén véletlenül kárt teszünk. A jelszavakat meglehetősen biztonságosan kezeli ez a parancs: eltárolni nem hagyja őket – a parancssoros alkalmazásnál a végrehajtás közben kérdezi meg azt. Ha munkaállomásról szeretnénk a teljes Windows 2000 tartományt kezelni, további lépésekkel fel lehet telepíteni a teljes Windows 2000 adminisztrátori eszközkészletet. Ezek egy része már megtalálható a Windows 2000 Professional-ben, az Administrative Tools *(Rendszerfelügyeleti eszközök)* között, de pl. az Active Directory Users and Computers, és más hasonló tartományi eszközök csak az adminpak.msi telepítése után érhetőek el. Ezt a javítócsomagok *(SP1 és SP2)* megfelelő állományából telepíthetjük a jobbklatty után az install menüpont kiválasztásával. Ha a gyárilag a Professional változatban szereplő eszközöket *(pl. Computer Management)* a Control Panel/Administrative Tools felől közelítjük meg, a megnyíló ablakban az adott eszköz ikonján a jobbklikk után megjelenő kis menüben megtalálható a „Runas" lehetőség is. Más parancsikonoknál további lépésekre van szükség, hogy a jobbkattintással ezt a funkciót is elérhessük, így az adminpak.msi telepítéséből származóaknál is. Tehát: ha a jobb-kattintás közben a SHIFT billentyűt nyomvatartjuk, akkor bármely parancsikonon elérhető lesz a Runas opció is *(ilyenkor a /profile kapcsolót használja a rendszer)*. Ilyenkor egy ablak ugrik elénk, ahol a megcélzott felhasználó paramétereit *(név, jelszó, tartomány vagy gép)* kell megadnunk.



Halandóból Administrator – csak egy jobbklikk

A runas parancssori paraméterezése egyértelműen felhasználható parancsikonok létrehozásánál. Ilyenkor egy adott program *(MMC, stb)* helyett a már felparaméterezett runas-t tároljuk le a hivatkozásban. Alapfeltétel ezen parancs használatához, hogy a Runas szolgáltatás fusson – amelyet célszerű az alapbeállításon, azaz az automatikus induláson hagyni.

DNS kiszolgáló vezérlése konzolról

Elérhetőség: X:\Program Files\ Support Tools\dnscmd.exe
Forrás: Windows 2000 Support Tools
A DNS kiszolgálók és adatbázisaik hagyományos tartozéka *(pl.: BIND)* a text zónaállományok és a parancssoros kezelés. A kedvenc „visszatérés a gyökerekhez" mozgalom eredményeként található ez a segédprogram az ingyenes Support Tools-ban. Szinte minden feladat elvégezhető vele, anélkül hogy a bűnös GUI-val kéne találkoznunk. Csak Windows 2000 DNS kiszolgálókkal működik együtt – az NT4 Resource Kit-ből ismert DNSStat utódjaként.

```
dnscmd kiszolgáló parancs [paraméterek]
```

A kiszolgáló címzése három módon is megoldható: 1. IP cím, 2. DNS név *(teljes tartománynév, FQDN)*, 3. NETBIOS név alapján. Az első kettő TCP/IP-n keresztüli RPC hívásokkal operál, míg a harmadik Named Pipes-on keresztül RPC-zik. A helyi kiszolgálót is elérhetjük. A jogosultságok szempontjából a bejelentkezett felhasználót veszi csak figyelembe.
A parancsok nagyon sokrétűek: csak képzeljük el, hogy a teljes grafikus DNS menedzser funkcionalitását kellett belesűríteni ebbe a 83 Kbyte-ba. A munka szintjeként választhatjuk a teljes DNS kiszolgálót, bizonyos zónákat, és tartományokat. Az általános kiszolgáló-lekérdezést a /Info kapcsolóval, a zónalekérdezést /Zoneinfo kapcsolóval tehetjük meg. Itt jegyezném meg, hogy a paraméterek mind betűméret-érzékenyek – nem hajtódnak végre a lekérdezések, beállítások, ha nem a szintaxisban leírtak szerint választjuk meg a nagybetű-kisbetű kombinációkat. Ezen parancsok további paramétereket is támogatnak – főleg mivel a zónákkal való műveleteknél a parancs mögött meg kell adni a zóna nevét teljes tartománynév *(FQDN)* formában.
A parancsok *(nem teljes)* listája:

- ResetForwarders – a továbbkérdezéshez használt DNS kiszolgálók adhatóak meg
- ResetListenAddresses – a kiszolgáló IP címei közül melyiken működjön a DNS szolgáltatás
- ZoneAdd – új zóna létrehozása
- ZoneResetType – a zóna typusának változtatása *(elsődleges, másodlagos, AD integrált)*
- RecordAdd – új bejegyzés felvétele *(A, SOA, NS, CNAME, MX, SRV)*
- Restart – a DNS kiszolgáló-szolgáltatás újraindítása

A komolyabb, scriptalapú DNS manipulációkat a Resource Kit Perl eszközeivel *(DNSZones, DNSRecord, DNSServer)* csináljuk. Akik még meredekebben, egyéni eszközök, scriptek létrehozását célozták meg, a DNSPROV segíthet *(szintén a Resource Kit-ből)*, mivel segítségével WMI szolgáltatóként a DNS kiszolgáló a WMI Platform SDK alapján kezelhetővé válik.

Apróságok

Ki is vagyok én? Ezt a filozofikus kérdést néha elmormolja a rendszergazda, miközben egy felhasználó gépén próbál életet lehelni egy jogosultságokon elhasalt programba. Hol vagy jó öreg Novell Netware, hol a Whoami parancs? Ez a programocska a Resource Kit lakója. Paraméterezése végtelenül egyszerű. Ha paraméter nélkül futtatjuk, a válasz a parancsot lefuttató felhasználó neve és tartományi hovatartozása lesz. Ha a /all kapcsolót használjuk, megkapjuk többek közt a csoporttagságokat, és a rendszerszintű jogokat *(pl. rendszer leállítása távolról)* is. A SID – azaz a Windows 2000 felhasználók, objektumok egyedi biztonsági azonosítója is kiíródik, ha valaki ez alapján szeretne mondjuk AD-t buherálni, debugolni.
(Ha csak a bejelentkezett felhasználóra vagyunk kíváncsiak, és nincs kéznél a Whoami, akkor a set /find "username" paranccsal is célt érhetünk el. A szerk.)

Floppy meghajtó zárolása

A felhasználók kontárkodási esélyeit radikálisan megnyirbálhatjuk, ha a Resource Kit FloppyLock szolgáltatás fut a munkaállomáson. Ilyenkor csak a Rendszergazdák és a Kiemelt Felhasználók *(Administrator és Power Users)* tudnak a floppyhoz hozzáférni. Server esetében ez csak a Rendszergazdákra vonatkozik. A FloppyLock szolgáltatást telepítenünk kell minden megvédendő gépre. A telepítéshez a szintén Resource Kit lakó grafikus felületű Service Installation Wizard, vagy a parancssoros instsrv.exe használható. A telepítése során a „FloppyLock" nevet, az állomány elérési útját *(A resource kit könyvtárában a floplock.exe)*, és a szolgáltatás típusát *(service)* kell megadni. Ha a kedves felhasználó megpróbál ezek után floppyhoz nyúlni, a jól ismert „Hozzáférés megtagadva" üzenettel fog szembesülni.

Terminálkapcsolatban a kliensgép meghajtóinak automatikus megosztása

A hagyományos terminálkliens funkció bővíthetőségét már az előző részben is tárgyaltam – akkor a kliens és szerver közötti állományműveleteket kivitelezését boncolgattam. Most egy másik, szintén hasznos funkcióbővítést vettem elő, amely lehetővé teszi, hogy a terminálkapcsolat létrejöttekor a kliensgép összes meghajtója automatikusan felcsatolva ott legyen a terminálkiszolgáló Windows felületében. Ezáltal az állományátvitel még egyszerűbben zajlik, amihez mindössze egy-egy állományt kell bemásolnunk a rendszerekbe, és egy-egy regisztrációs adatbázis-módosítást kell automatizáltan megtennünk. Első lépésként a terminálkiszolgáló SYSTEM32 könyvtárába másoljuk be a drmapsrv.exe-t, majd illesszük be a regisztrációs adatbázisba a drmapsrv.reg-et a jobbklatty-merge kiválasztásával. A kliensen a drmapclt.dll-t kell a Terminal Service Client könyvtárba bemásolni, majd a drmapclt.reg-et a regisztrációs adatbázisba beolvasztani. Ezek után a manuális megosztás-felcsatolás helyett a kánaáni automatizáció előnyeit élvezhetjük.

Radványszki Gábor
gabor@radvanszki.net
MCP

A cikkben szereplő URL-ek:
[1] http://support.microsoft.com/support/kb/articles/Q272/4/72.asp

ISA Server – az alapok *(II. rész)*

Előző havi cikkemben a tűzfalakról írtam. Az ott leírtak sokszor vissza fognak köszönni a cikksorozatban, és – ha el nem felejtem – próbálok is majd utalni rájuk. Az e havi cikk, ígéretemtől eltérően, még nem tartalmaz konkrét „így kattintsunk" jellegű dolgokat, erre a következő cikkig még várni kell. A most következő oldalakon próbálom megismertetni a Kedves Olvasót az ISA Server-rel. Ez több szempontból is fontos. Egyrészt azért, mert a későbbiekben minden igyekezetem ellenére sem fogom tudni az összes funkciót a teljes részletességgel leírni, de szükségesnek tartom, hogy bemutassam *(legalább dióhéjban)* az ISA-ban rejlő összes lehetőséget *(vagy legalábbis ezek nagy részét)*. Másrészt szeretném, ha mindenki elkülönítené az ISA Server-t a Proxy Server 2.0-tól. Az ISA Server 2000 egy nagy teljesítményű tűzfal és web gyorsítótár, melyet úgy terveztek, hogy nagyvállalati környezetben is megállja a helyét, és még a legkényesebb igényeket is kielégítse. Természetesen az ISA is a Windows 2000-re épül, és maximálisan kihasználja az operációs rendszer adta lehetőségeket *(például biztonsági beállítások, rendszerfelügyelet és címtár)*, hogy megvalósítható legyen az internetelérés házirendalapú felügyelete, és a rendelkezésre álló sávszélesség optimális kihasználása.

- Biztonság: az ISA Server tűzfalként megakadályozza a hálózatok jogosulatlan elérését, védi az internetről elérhető kiszolgálókat a külső támadásoktól és természetesen ellenőrzi az összes rajta átmenő forgalmat.
- Optimális sávszélességkihasználás: az ISA is, mint általában a proxy tűzfalak, tartalmaz gyorsítótárat. Ennek segítségével gyorsabb webelérést biztosít, mert a kérések egy részét az internet helyett a helyi gyorsítótárból szolgálja ki. A hálózati forgalom csökkenése miatt csökkenthetők az internetelérésre fordított költségek. Reverse proxy-ként működve csökkenti a webkiszolgálók terhelését.
- Rendszerfelügyelet: a felhasználók internethozzáférésének szabályozása házirendek kikényszerítésével. A cég szempontjából hasznos alkalmazásokra és weboldalakra korlátozható az Internet használata, ezzel növelhető a termelékenység *(mert ha a mélyen tisztelt dolgozó nem nézegetheti az XXX-es oldalakat, az így keletkezett „szabadidőben" akár dolgozhat is :))*. A sávszélesség üzleti érdekek szerint osztható el a felhasználók között, s az internethasználatról jelentések készíthetők. Ezek pedig nagyon fontosak a jövőbeni fejlesztések meghatározása szempontjából.
- Testreszabhatóság: az ISA Server tűzfal és gyorsítótár-összetevői a vállalat igényeitől és hálózatának felépítésétől függően külön gépekre is telepíthetők *(de természetesen futhatnak ugyanazon a gépen is)*. A fizikai kiépítéstől függetlenül – a Windows 2000 képességeit kihasználva – az ISA Server-ek egységes rendszerként képesek működni *(például ugyanazt a házirendet kényszerítik ki)*.

A fent említetteken túl az ISA Server-hez gazdag SDK és API készlet tartozik, melyek segítségével például a következő területeken bővíthető az ISA funkcionalitása: felügyelet, vírusszűrés, tartalomszűrés, bizonyos webhelyek elérésének megakadályozása, alkalmazásszűrők, valósidejű megfigyelések, stb.

A .NET nagyvállalati kiszolgálók

Jogos a kérdés, hogy hogyan is illeszkedik az ISA a .NET „képbe". Jól! Sőt, merem állítani, hogy az ISA kulcsfontosságú tagja a .NET családnak. Miért is? Mint tudjuk, a .NET kiszolgálók alkotják a Microsoft minden igényt kielégítő alkalmazáskiszolgáló családját. E kiszolgálók felhasználásával skálázható, integrált, könnyen felügyelhető webalapú megoldások és szolgáltatások készíthetők és helyezhetők üzembe. Mindez azonban fabatkát sem érne a biztonság nélkül. Én például nem bíznám a személyes adataimat, ne adj' isten a pénzem egy olyan kiszolgálóra, amelyet csak úgy „rádugtak az Internetre".

Egyszerű rendszerfelügyelet + átlátszóság a felhasználók számára = Alacsonyabb TCO

A biztonság és a gyorsítótár külön felügyelete általában különféle eszközöket, infrastruktúrát és tapasztalt rendszergazdákat igényel. Ezzel nő a rendszer összetettsége, üzemeltetési költsége – és kompatibilitási problémák adódhatnak. Az ISA Server egységes, házirendalapú felügyeleti eszköze segít a rendszergazdáknak egy központi helyről elvégezni a beállításokat, így javul a hálózat átláthatósága, és csökken a teljes birtoklási költség. A vállalatoknak előnyösek a tűzfal és gyorsítótár ellentmondásmentes szabályai. A Windows 2000 által biztosított integrált felügyeleti rendszer biztosítja ezen szabályok együttes megjelenítését, így a tűzfal és gyorsítótár infrastruktúra egyszerre felügyelhető.

A kiváló felügyeleti lehetőségek mellett a vállalatok a könnyű üzembehelyezhetőséget is igénylik. Az ISA Server-rel csak a tűzfal és a gyorsítótár kiszolgálók beállítására van szükség, így egyszerűsödik a kiszolgálók közzététele, a tűzfal és a gyorsítótár beállítása. Az ISA Server biztonságos címfordítási *(Secure Network Address Translation – SecureNAT)* képességének használatával a rendszergazdáknak nem kell további szoftvert telepíteni az ügyfélgépekre vagy a közzétett kiszolgálókra a tűzfal vagy a gyorsítótár használatához. Az ISA Server láthatatlan az ügyfélgépek és más kiszolgálók számára, így csökkenthetők a rendszerfelügyelet összetettsége és költségei.

És egy kis realitás:

Ne feledkezzünk el arról sem, hogy a könnyű felügyelhetőség – bár igen fontos – nem pótolhatja a szakértelmet, és nem egyenlő a biztonsággal.

Az ISA Server üzemmódjai

Az ISA Server háromféle üzemmódban működhet: tűzfal, gyorsítótár és integrált *(tűzfal és gyorsítótár ugyanazon a gépen)* módban.

Internet tűzfal

Az ISA Server üzembehelyezhető tűzfalként, amely az Internet és a belső ügyfélgépek közti biztonságos átjáróként működik. Az ISA Server a kommunikációban résztvevők számára észrevétlen. A felhasználó egész addig nem is veszi észre, hogy a tűzfal jelen van, amíg nem próbál olyan szolgáltatást vagy webhelyet elérni, aminek az ISA tiltja a használatát. A biztonsági házirendek beállításával megelőzhető a hálózat illetéktelen használata, a veszélyes dolgok hálózatra kerülése, és letiltható a kifelé irányuló forgalom, például a felhasználó vagy felhasználói csoport, az alkalmazás, a kapcsolat célja, a tartalom típusa vagy a napszak szerint.
Az ISA Server ezt megvalósító fő szolgáltatásai:

- többrétegű tartalomfigyelés – csomag-, kapcsolat- és alkalmazásszűrők
- „Intelligens", adatfelismerő alkalmazásszűrők
- beépített behatolásészlelés
- a rendszer megerősítése a Windows 2000 biztonságosabbá tételéért
- beépített virtuális magánhálózat *(VPN)* lehetőségek

Biztonságos kiszolgáló közzététel

Az ISA a belső hálózat biztonságának veszélyeztetése nélkül biztosítja a szolgáltatások Interneten való közzétételét. Beállíthatók a web és más kiszolgálók közzétételének szabályai, melyek eldöntik mely kérések továbbítódnak az ISA Server mögött található kiszolgálók felé. Ezzel biztosított a belső kiszolgálók biztonsága. Például egy Microsoft Exchange kiszolgáló elhelyezhető az ISA Server mögött, és beállíthatók a kiszolgáló közzétételének szabályai, melyek lehetővé teszik az Interneten való közzétételt. Az Exchange Server-re bejövő leveleket elfogja az ISA Server, amely az ügyfelek számára levelező kiszolgálónak látszik. Az ISA Server képes megszűrni a forgalmat, majd továbbítani az Exchange Server-nek. Magát az Exchange Server-t nem is láthatják a külső felhasználók, így az végig biztonságos környezetben marad.
Az ISA Server kiszolgáló közzétételt elősegítő szolgáltatásai:

- könnyen használható kiszolgáló közzététele varázslók
- SecureNAT
- széleskörű protokoll-támogatás *(például HTTP, FTP, H.323, SMTP, folyamatos átviteli szolgáltatások)*

Belső ügyfeleket kiszolgáló gyorsítótár

Az ISA Server telepíthető belső ügyfeleket kiszolgáló gyorsítótárként, amely azoknak Internet elérést biztosít. Az ISA Server az Internetről letöltött objektumokat egy központi gyorsítótárba helyezi, amelyet bármely webböngésző elérhet. A tartalom helyi meghajtóról való „szállítása" természetesen sokkal kevesebb feldolgozási műveletet és időt igényel. Ez növeli az ügyfél böngészőjének teljesítményét, csökkenti a válaszidőt, boldogítja a felhasználót és csökkenti az internetkapcsolat használatát.
Az ISA Server fő gyorsítótár szolgáltatásai:

- a gyorsítótár egy része a rendszermemóriában helyezkedik el
- időzített letöltések
- elosztott és hierarchikus gyorsítótár láncok létrehozásának támogatása
- a népszerű webhelyek tartalmának előzetes letöltése *(Active Caching)*

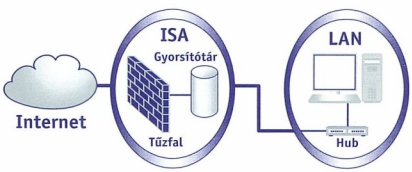
Külső ügyfeleket kiszolgáló gyorsítótár *(reverse proxy)*

Az ISA Server telepíthető a webes alkalmazásokat futtató webkiszolgálók elé. A bejövő webkérésekre az ISA webkiszolgálóként válaszol, az ügyfél kéréseit pedig a saját gyorsítótárából szolgálja ki, csak azokat a kéréseket továbbítva a webkiszolgáló felé, amelyeket nem tud gyorsítótárából kiszolgálni.
Az ISA Server fő reverse proxy szolgáltatásai:

- webközzététel varázslók
- a gyorsítótár egy része a rendszermemóriában helyezkedik el
- az összes ügyfél számára láthatatlan
- elosztott gyorsítótár kialakítása a Cache Array Routing Protocol *(CARP)* segítségével

Integrált tűzfal és web gyorsítótár kiszolgáló

Ez tulajdonképpen a fenti funkciók összevonása egyetlen kiszolgálóra. Elsősorban kis és közepes méretű vállalatok számára ajánlott megoldás, a nagyvállalati rendszerekben – teljesítményszempontok miatt – ajánlott a funkciók szétválasztása.



ISA Server integrált üzemmódban

Felügyelet – a házirendek és szabályok alapjai

A Microsoft ISA Server-t az egyszerű üzemeltetés érdekében kiváló felügyeleti képességekkel ruházták fel. A gyorsítótárak és tűzfalak központi helyről történő felügyelete növeli a biztonságot, a szabályozások következetességét, és leegyszerűsíti a rendszer felügyeletét. Ez ugyanúgy igaz a Small Business Server-ben található ISA Server-re, mint a több fiókiroda között létrehozott kiszolgálóláncokra, vagy akár egy Internet szolgáltatónál kiépített ISA Server tömbre.
Az alábbiakban a legfőbb eszközök áttekintése következik, melyek segítségével biztonságosan szabályozhatók a befelé és kifelé irányuló kérések, és az azokra érkező válaszok.

Házirendek

Beállítható szabályok, melyek a helyi hálózat és az Internet közti kommunikációt szabályozzák. Ezek a szabályok vállalati vagy kiszolgálótömb szintű házirendekben adhatók meg, és központilag tárolódnak az Active Directoryban.

Kiszolgálótömb házirend vagy vállalati házirend?

Tömbszinten webhelyre, tartalomra, protokollra, webközzétételre vonatkozó szabályok, és IP csomagszűrők készíthetők. E szabályok együttese alkotja a tömb házirendjét. A tömb házirendje meghatározza, hogy az ISA Server ügyfelek hogyan kommunikálnak az Interneten, és mely kommu-

nikáció van számukra engedélyezve. A tömbházirend csak az adott tömbben levő ISA Servereken érvényesül.
Készíthető vállalati házirend is, mely webhelyre, tartalomra és protokollra vonatkozó szabályokat tartalmaz. A vállalati házirend bármely tömbre érvényesíthető, és kibővíthető a tömb saját házirendjével. Ez teszi lehetővé a fiókirodák és osztályok rendszergazdáinak a vállalati házirendek átvételét. A tömbházirendek csak szigorúbbak lehetnek a vállalati házirendnél, ami azt jelenti, hogy a tömbszintű házirendek segítségével finomhangolható a vállalati házirend, de csak további tiltásokat tartalmazhatnak.

Az ISA Server szabályainak bemutatása

Az ISA Server egyéni biztonsági igényeket is kielégíthet olyan szabályok megadásával és beállításával, melyek eldöntik, hogy mely felhasználók, szolgáltatások, portok és tartományok hozzáférése engedélyezett a helyi hálózat, vagy az Internet számítógépein. Az ISA Server-ben háromféle szabály adható meg:

- a hozzáférések házirendjének szabályai
- sávszélességszabályok
- közzétételi szabályok

A hozzáférések házirendjének szabályai

Az ISA Server használható a hozzáférések házirendjének beállítására, mely webhely-, tartalom- és protokollszabályokból valamint IP csomagszűrőkből áll:

- A webhely- és tartalomszabályok megadják, hogy mely webhelyek érhetők el az ISA mögött elhelyezkedő ügyfelek számára. Ezek alkalmazásszinten kerülnek feldolgozásra.
- A protokollszabályok megadják az ISA Server mögötti felhasználók által használható protokollokat. A protokollszabályok alkalmazásszinten kerülnek feldolgozásra.
- Az IP csomagszűrők engedélyezik vagy megakadályozzák a beállított IP címek között megadott protokollt, és portokat használó kommunikációt. Az IP csomagszűrők csomagszinten kerülnek feldolgozásra.

Sávszélességszabályok

Az ISA Server sávszélességszabályai a Windows 2000 QoS *(Quality of Service, garantált szolgáltatásminőség)* képességeire épülnek, hogy meghatározhassák, mekkora sávszélesség osztható ki egy adott Internet kérésnek. A sávszélességszabályok alkalmazásszinten kerülnek feldolgozásra.

Közzétételi szabályok

A kiszolgálóközzétételi szabályok szűrik az összes bejövő és kimenő kérést. A bejövő kéréseket a megfelelő ISA Server mögötti kiszolgálóhoz rendelik hozzá. Például az Exchange Server 2000 a külső felhasználók számára érzékelhetetlenül tehető elérhetővé az ISA Server-en keresztül. A webközzétételi szabályok pedig a bejövő kéréseket a megfelelő ISA Server mögötti web kiszolgálókhoz rendelik hozzá.

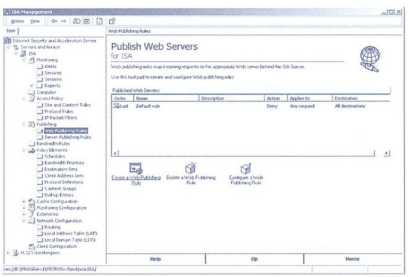
A házirend alkotóelemei

A házirend alkotóelemei a szabályok. Aprólékos szabályozást tesznek lehetővé nemcsak telephelyenként vagy felhasználónként, hanem például a sávszélesség elosztásán, megadott protokollokon és tartalomtípus szerint is. Mint az ISA Server legtöbb része, az egyedi igények kielégítése érdekében a házirend elemei is bővíthetők. A vállalatok például vásárolhatnak tiltott URL-eket tartalmazó listákat erre szakosodott cégektől, melyek beilleszthetők a házirend alkotóelemei közé. Az egyes elemek a következők:

- az időzítések meghatározzák, hogy mely időpontokban engedélyezett vagy tiltott az ügyfelek hozzáférése.
- a sávszélességelsőbbségek meghatározzák a befelé és kifelé irányuló forgalom súlyozását az elérhető sávszélesség jobb kihasználása érdekében
- a célállomás csoportok távoli webhelyeket definiálnak IP cím vagy URL alapján
- az ügyfélcím csoportok belső ügyfeleket határoznak meg IP cím, vagy a tartomány felhasználói fiókjai, felhasználói csoportjai alapján
- a protokolldefiníciók protokollok szerint finomítják a szabályokat
- a tartalomcsoportok a leggyakoribb fájltípusok logikai csoportjai *(például video, audio, képek)*

Az ISA Server konzol

Minden felügyeleti, megfigyelő és jelentéskészítő eszköz a felügyeleti alkalmazásból érhető el. Ez a program a közismert Microsoft Management Console-t használja. Azt hiszem, az MMC használatát nem is kell részleteznem.



„Valahol már láttam...“ :)

Windows 2000 integráció és felügyelet

Az ISA Server kihasználja a Windows 2000 és Active Directory által biztosított felügyeleti lehetőségeket. Ezzel egyszeri bejelentkezés válik lehetővé. Lerövidül a változtatások felhasználókhoz való eljutásának ideje, és a felhasználói fiókban tárolt információk felhasználhatók arra, hogy az ISA Server segítségével engedélyezzék vagy letiltsák a felhasználó bizonyos Internetes szolgálattásokhoz való hozzáférését.
Az operációs rendszerrel való integráltság kihasználásának másik példája a Windows 2000 Server Performance Monitora, mely számos ISA Server jellemző valósidejű mérését lehetővé teszi. A Windows 2000 eseménynaplója segít követni az ISA Server-rel kapcsolatos eseményeket, és segít a hibaelhárításban. A távoli adminisztráció pedig az MMC és/vagy a Terminal Services használatával valósítható meg.
Az ISA Server rugalmas, jól testreszabható felügyeleti modult tartalmaz, így csökkentheti a tűzfal, a gyorsítótár és a sávszélességszabályzó eszközök felügyeletének bonyolultságát.

SecureNAT ügyfelek

Azokat az ügyfélszámítógépeket, melyekre nincs tűzfal ügyfélszoftver telepítve, SecureNAT ügyfeleknek nevezzük. A SecureNAT ügyfelek használhatják az ISA Server legtöbb funkcióját, így például a hozzáférést szabályozó funkciók nagy részét is, kivéve a magasszintű protokolltámogatást és a felhasználónkénti hitelesítést. A SecureNAT használatakor az ügyfélszámítógépek semmilyen beállítást nem igényelnek, így az ISA Server üzembe helyezése és menedzselése a felhasználók számára észrevétlen, a rendszergazdák számára pedig sokkal egyszerűbb.
Bár a SecureNAT ügyfelek semmilyen különleges szoftvert nem igényelnek, az alapértelmezett átjárót úgy kell rajtuk beállítani, hogy az összes Internet felé irányuló forgalom az ISA Serveren haladjon át. Az ISA Server is beállítható alapértelmezett átjáróként, de ez nem követelmény.

A SecureNAT és a Windows 2000 NAT

Az ISA Server az ISA Server házirendek SecureNAT ügyfelekre való kikényszerítésével bővíti ki a Windows 2000 címfordítási *(NAT)* képességeit. Más szóval, a SecureNAT nagyobb biztonságot és jobb szabályozhatóságot nyújt, mert a tartalmak áthaladnak az alkalmazásszűrőkön, valamint a házirendek és a sávszélességszabályzás is érvényesül rajtuk. Minden ISA Server szabály *(protokollhasználat, célállomások és tartalom típusa)* alkalmazható a SecureNAT ügyfelekre, annak ellenére, hogy a Windows 2000 NAT nem tartalmaz beépített hitelesítési eljárást.
Mivel a SecureNAT ügyfelek kéréseit főként a tűzfalszolgáltatás kezeli, a SecureNAT ügyfelek a következő biztonsági előnyökkel bírnak:

- Az alkalmazásszűrők képesek megváltoztatni az adatfolyamot, hogy átjuthassanak az összetett protokollok. A Windows 2000 NAT-nál ez NAT szerkesztők *(NAT Editor)* használatával valósítható meg, melyeket kernelmódú NAT szerkesztő meghajtóként írtak meg a Windows 2000-hez.
- A tűzfalszolgáltatás átadja a HTTP kéréseket a web proxy szolgáltatásnak, amely a gyorsítótárat kezeli, és biztosítja a webhely és – tartalom szabályok megfelelő alkalmazását.

Tűzfal ügyfelek

A tűzfal ügyfélprogram telepítése lehetővé teszi a hozzáférési házirendek hitelesített felhasználókra való alkalmazását is *(nemcsak az ügyfélszámítógépek IP címeire)*. Így például hozzáférési és sávszélességszabályokat érvényesíthetünk bizonyos Windows NT vagy Active Directory tartományi felhasználókra és felhasználói csoportokra, amelyek NTLM vagy Kerberos jegyek segítségével lettek hitelesítve.
A tűzfal ügyfél támogatja a WinSock alkalmazásokat is. A tűzfal ügyfél telepítése nem változtat a WinSock alkalmazások beállításain, hanem ugyanazt a WinSock dll fájlt használja, amit az alkalmazások. A tűzfal ügyfél elfogja az alkalmazások hívásait, és eldönti, hogy továbbítsa-e a kérést az ISA Server számítógépnek.
A tűzfal szolgáltatás az 1.1-es és 2.0-s WinSock verziót használó alkalmazásokat támogatja. Mielőtt egy WinSock alkalmazás hozzáférhetne az ISA Server-en keresztül az Internethez, a kiszolgálót is be kell állítani, hogy engedélyezze a felhasználónak a szükséges protokoll és a szükséges szolgáltatás portok használatát.
A tűzfal ügyfél Windows 95, Windows 98, Windows NT 4.0 vagy Windows 2000 operációs rendszert futtató gépekre telepíthető.

A SecureNAT ügyfelek és a tűzfal ügyfelek

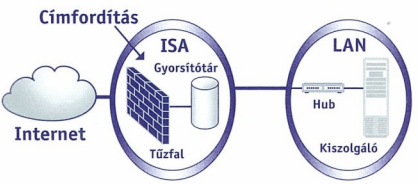
Az ISA Server biztosítja az összes tűzfal és SecureNAT ügyféllel folytatott hálózati kommunikáció biztonságosságát, emellett minden ügyfél élvezi az ISA Server gyorsítótárának előnyeit.
Mind a SecureNAT, mind a tűzfal ügyfelek védhetők alkalmazásszűrőkkel, melyek tartalomellenőrzést és más biztonsági tevékenységeket végeznek. Ha az IP továbbítás elérhető, a jobb teljesítmény és késleltetés eléréséhez mindkettő használhatja a kernel módban futó „adatszivattyút" *(data pump)*.

Biztonságos web- és kiszolgálóközzététel

Az ISA Server a belső hálózat biztonságának veszélyeztetése nélkül biztosítja az Internetre való közzétételt. A web- és kiszolgálóközzétételi szabályok beállíthatók arra, hogy eldöntsék mely kéréseket kell beküldeni az ISA Server mögött levő kiszolgálónak. A közzétett kiszolgáló külvilág felé való megszemélyesítésével az ISA Server a nagyobb biztonságot és a gyorsítótárba helyezett webtartalom gyorsabb elérését biztosítja. A közzétett belső kiszolgálón nem szükséges további szoftvereket telepíteni, mert az ISA Server a SecureNAT-ot használja a felhasználó számára észrevétlen kommunikációhoz.

Kiszolgáló közzététel

Egy belső Microsoft Exchange Server például SMTP leveleket küld és fogad az Internetről. Mivel az SMTP mind a bejövő, mind a kimenő kommunikációhoz a 25-ös portot használja, az Exchange Server az ISA Server külső címének 25-ös portjához van kapcsolva. Ily módon az Exchange Server felé irányulhatnak bejövő kapcsolatok. A közzététel ezen módjának megvalósításához készíteni kell egy vagy több kiszolgáló-közzétételi szabályt, melyek megadják, hogy mely belső kiszolgálóknak engedélyezett az Interneten való megjelenés. Az ISA Server egy vagy több kiszolgáló nevében figyeli a kéréseket, és átirányítja azokat a megfelelő kiszolgálóhoz.

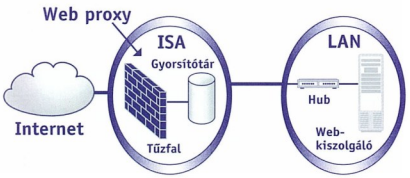


Kiszolgáló közzététele

Webközzététel

A webkiszolgáló az ISA Server mögé helyezhető. A webközzétételi szabályok teszik lehetővé a webkiszolgáló Interneten való „megjelenítését". A webkiszolgáló felé irányuló bejövő kéréseket az ügyfelek felé webkiszolgálóként megjelenő ISA Server a gyorsítótárából szolgálja ki, és csak akkor továbbítja a kéréseket a belső kiszolgáló felé, ha azok a gyorsítótárból nem szolgálhatók ki. Eközben a webkiszolgáló biztonságos környezetben van, és elérheti a belső hálózati szolgáltatásokat.

Az ISA Server megszemélyesíti a webhelyet, mivel az ő IP címe az egyetlen, amelyhez DNS név *(Host rekord)* tartozik. A webközzétételi szabály továbbítja a kéréseket a belső hálózat megfelelő webkiszolgálója felé. Az ISA Server átveszi a webtartalom feldolgozásának egy részét, biztonságot jelent, központi SSL kulcsfelügyeleti helyet biztosít és lehetővé teszi több kiszolgáló egy webhelyként való közzétételét.



Webkiszolgáló közzététele

A rendszer megerősítése

A tűzfalnak biztonságosnak kell lennie, hiszen minden támadás során ő a „pofozózsák". Nyilvánvalóan védi saját magát is, de azért sosem árthat, ha az operációs rendszer, amin a tűzfal fut, mentes a biztonsági résektől. Ezt segíti elő az ISA-ban található System Hardening Wizard, amivel az ISA szerepkörének megfelelő biztonsági mintát húzhatjuk rá tűzfalunkra. Vigyázat! Ez nem csodaszer! A biztonsági minták tartalmazzák ugyan a Windows 2000 biztonsági szolgáltatásainak megfelelő beállításait, de a folyamatosan megjelenő hibajavításokat nem. Ha valahol, hát az ISA-nál igazán fontos ezek folyamatos követése, és a kiszolgáló naprakészen tartása.

Az ISA, mint webgyorsítótár

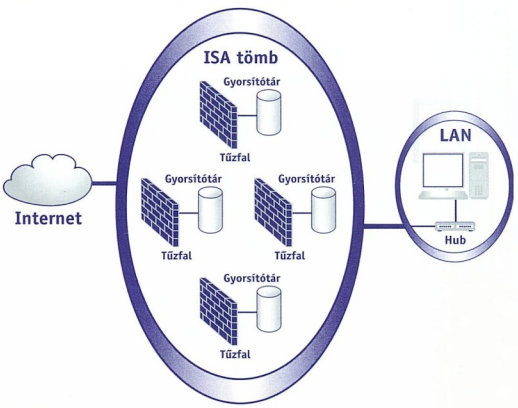
A legtöbb webböngésző támogatja az objektumok helyi gyorsítótárba helyezését, ami azt jelenti, hogy az ügyfélgépen tárolódnak a lekért weblapok. Ha ezt egy kicsit továbbgondoljuk, szinte adja magát az ötlet, hogy legyen egy olyan központi gyorsítótár, amely lehetővé teszi, hogy az Internet felől jövő dróton ezek az anyagok csak egyszer menjenek át, ne kelljen mégegyszer megtennünk ezt – ha már egyszer letöltöttük őket. Ez persze így túl egyszerű volna. Egyrészt, mert idővel az egész Internet a merevlemezeinken lenne :), másrészt egy idő után rengeteg fölösleges és elavult dolgot találnánk itt. Mint az előző részben már említettem, az alkalmazásszintű ellenőrzés eléggé processzorigényes feladat, viszont a HTTP és FTP objektumok biztosítása az ISA Server memóriájából vagy merevlemezéről sokkal kevesebb feldolgozást igényel, mint az Internetről jövők *(mert hát minek is ellenőriznénk újra meg újra azokat a dolgokat, amiket egyszer már ellenőriztünk?)*. Akár a befelé, akár a kifelé irányuló kéréseket kiszolgáló gyorsítótárként alkalmazzák, az ISA Server gyorsítja a böngészést, csökkenti a válaszidőket és optimalizálja a sávszélesség kihasználását.

A gyorsítótár tartalmának időzített letöltése

Az ISA Server képes a gyorsítótárba meghatározott tartalmakat időzítetten letölteni. Egy háttérben futó folyamat előre beállított időzítés szerint mindig akkor tölti le a tartalmakat, amikor az ISA Server éppen nem kezel web proxy ügyfelektől érkezett kéréseket.
A gyorsítótár tartalmának időzített letöltése lehetővé teszi, hogy az ISA Server a várhatóan hamarosan kért HTTP tartalommal frissítse a gyorsítótár tartalmát *(ilyenek lehetnek például a hír-weblapok)*. Ha okosan használják, az időzített letöltés értékes sávszélességet takarít meg, és az átbocsátóképesség befolyásolása nélkül növeli a gyorsítótár teljesítményét.
A gyorsítótár tartalmának időzített letöltése egy Windows 2000 szolgáltatás *(service)*. Éppen úgy meg lehet állítani, elindítani, szüneteltetni, mint bármely hasonló szolgáltatást.

Elosztott gyorsítótárak

Az ISA Server egyik leghasznosabb képessége az elosztott gyorsítótárak használatának támogatása. Ez a képesség teszi alkalmassá az ISA Server-t a nagyvállalatok és Internet szolgáltatók rendkívül nagy igényeinek kielégítésére. A gyorsítótár objektumainak terheléselosztása növeli a gyorsítótár teljesítményét, és emellett hibatűrést is biztosít. Az elosztott gyorsítótárak használata gyorsítótár tömbök vagy láncok, vagy a kettő kombinációjának létrehozásával lehetséges.
Az elosztott gyorsítótárak használata azért fontos, mert így a gyorsítótárak közelebb kerülnek a felhasználókhoz. Egy nagyvállalaton belül a gyorsítótár láncok például a vállalati hálózat határától *(egy központi helytől)* egészen a fiókiroda, vagy munkacsoport szintig lenyúlhatnak. Egy Internetszolgáltató hálózatán belül a gyorsítótárak a szolgáltató helyi megjelenési pontjaitól a központi megjelenési pont felé húzódnak.
A gyorsítótárak felhasználókhoz való közelebb helyezésével csökkenthető a hálózati forgalom és a költségek, és növelhető a teljesítmény.
A Microsoft ISA a több ISA Server-ből álló tömbök létrehozását a Cache Array Routing Protocol használatával támogatja. Ez a gyorsítótár objektumok terheléselosztásával javítja az aktív és passzív gyorsítótárak teljesítményét.



ISA Server tömb

Cache Array Routing Protocol — skálázhatóság

A Microsoft azért fejlesztette ki a CARP-ot, hogy ennek segítségével kommunikáljanak egymással az ISA Server számítógépek egy láncban vagy egy tömbben, és hatékony, skálázható gyorsítótárak együttesét alkossák. A CARP megoldja a gyorsítótárak használatának hatékonysági problémáit. Más gyorsítótár megoldásokkal ellentétben a CARP elosztott irá-

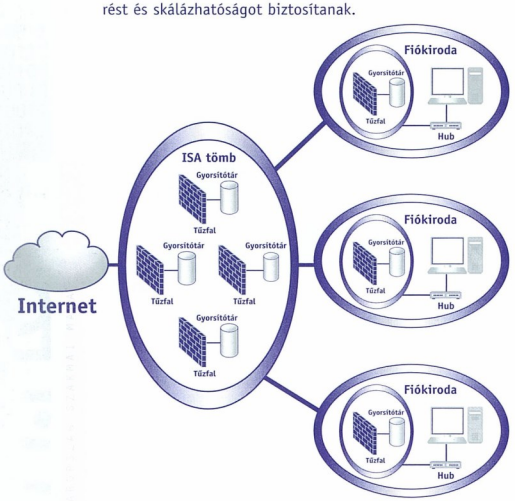
nyítási eljárást használ a lekérdezések helyett a tárolt Internet objektumokra irányuló kérések teljesítéséhez. Ez biztosítja a gyors elérést, az alacsony fenntartási költségeket és a tárterület hatékony kihasználását. Mindegyik internetobjektum a tömb egyetlen ISA Server-én található csak meg, így a tömbök egyetlen logikai gyorsítótárként működnek. Természetesen, a CARP biztosítja a skálázhatóságot is, vagyis ahogy egyre több ISA kerül a tömbbe, úgy nő a teljesítmény is.

Gyorsítótár láncok *(hierarchikus gyorsítótárak)*

A lánc az ISA Server-t futtató számítógépek hierarchikus kapcsolata. Az ügyfelek kérései felfelé haladnak a gyorsítótár-hierarchiában, egészen addig, amíg meg nem találódik *(csak magyarosan! :))* a keresett objektum. Egy fiókirodában levő ügyfél kérése például először a fiókiroda ISA Server-éhez megy, majd a területi vállalati központhoz, mielőtt kijutna az Internetre.

A gyorsítótár lánc tagja lehet egy ISA Server számítógép, vagy akár egy ISA Server tömb is. A láncok kialakítása is hatékony mód a kiszolgálók terhelésének elosztására és a hibatűrés kialakítására. Secure sockets layer *(SSL)* láncok kialakítása is támogatott.

Az elosztott gyorsítótárak és a hierarchikus láncok kialakításával az ISA Server akár a legnagyobb nagyvállalat igényeinek megfelelően is skálázható. A CARP hatékony tömbök kialakítására szolgál, melyek jó teljesítményt, hibatűrést és skálázhatóságot biztosítanak.



ISA Server tömb és láncok együttese

Riasztások

Az ISA Server testreszabható rugalmas riasztási szolgáltatása segít a hálózat elleni támadások és a rendszerelemények megfigyelésében, és azok megfelelő kezelésében. A riasztási szolgáltatás biztosítja előre definiált cselekvéssorozatok elindításának támogatását *(automatikus levélküldés a rendszergazdának, szolgáltatások elindítása és megállítása, egyedi szkriptek vagy programok futtatása)*. A riasztási szolgáltatás eseményelosztóként és -szűrőként működik. Ez felelős az események elfogásáért, az adott feltételek teljesülésének ellenőrzéséért és a megfelelő cselekvések végrehajtásáért.

Az eseményalapú riasztások és a testreszabható cselekvéssorozatok végrehajtásának támogatásával az ISA Server olyan rugalmas felügyeleti eszközt biztosít, mely segítséget nyújt a biztonságos tűzfal és hálózat fenntartásában. A vállalati biztonság egyik fő eleme a támadásokról való értesülés és a gyors ellenlépés végrehajtásának képessége. Ez tényleg nagyon fontos. Valljuk be őszintén, hogy ami jól működik, arról az „Egy gonddal kevesebb." felkiáltás kíséretében hajlamosak vagyunk elfelejtkezni. Közben meg lehet, hogy valaki már két hete vígan „hekkelgeti" a tűzfalunkat.

Jelentések készítése

Az ISA Server-ben előre elkészített jelentésminták találhatók, melyek nagyban megkönnyítik a biztonsági problémák, és az internethasználat elemzését. Bár maga az ISA csak alapvető jelentések készítését támogatja, kiterjedt jelentéskezelő API-készletet tartalmaz, melynek segítségével külső gyártók jelentéskészítő és -kezelő eszközeit is alkalmazhatjuk.

A jelentéskészítő mechanizmus ISA Serverenként egy adatbázisba gyűjti a naplókat, majd a jelentés készítésekor ezen naplók vizsgált időtartamra vonatkozó bejegyzései egyetlen adatbázisba kerülnek, és ennek alapján készül el a jelentés. Ez az adatbázis egy adott ISA Server-en helyezkedik el, és a jelentések is csak ezen a számítógépen nézhetők meg.

Természetesen az összegyűjtött adatok alapján rendszeres időközönként, és automatikusan összefoglalók készíthetők vállalatunk internetfelhasználásáról.

Előre definiált jelentések

Az ISA-ban az alábbi előre definiált jelentések találhatók meg:

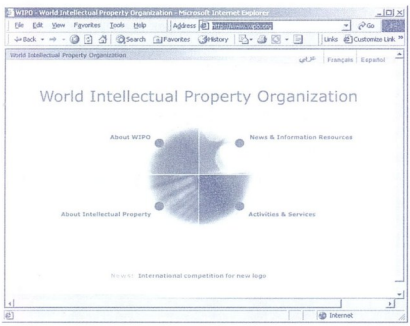
- összesített jelentések
- a webhasználatról szóló jelentések
- az alkalmazások használatáról szóló jelentések
- forgalmi jelentések és sávszélességkihasználtság jelentések
- biztonsági jelentések

Hát ennyi lett volna az e havi cikk. Jövő hónapban *(istenbizony)* felépítünk egy tesztrendszert. *(Tehát aki tényleg ki akarja próbálni a leírtakat, kezdje gyűjteni a PC-ket. Jövő hónapban még csak háromra lesz szükség, később négyre. Ja, és az egyikben legyen három darab hálózati kártya. Szükség lesz rá.)*

PB
MCSE

Domain vitarendezési eljárások

Korábban már ugyanezeken az oldalakon szóltam a magyar domainregisztrációs rendszerről és ott nagy vonalakban kitértem a hazai vitarendezési eljárásra is.
Ezúttal azonban a gTLD *(Generic Top Level Domain)* körében az ICANN **[1]** által meghatározott szabályok szerint folyó, a WIPO **[2]** keretein belül működő sajátos vitarendezési eljárásokat szeretném ismertetni.



A WIPO-n megtalálhatók a korábbi vitás ügyek...

A generikus kódok *(gTLD)* az eddigi használat szerint a .com, .net, .org, de mint tudjuk, számos új elfogadása már megtörtént és bevezetése a közeli jövőben várható. E körben a domainregisztráció szabályait az ICANN állapította meg. Itt nem működik semmilyen különösebb előzetes vizsgálat, a „first step first served" elve szerint az igénylőnek – ha az adott domain nem foglalt – automatikusan regisztrálják a kért nevet. Akinek jogát sérti az adott regisztrálás, nem marad más hátra a rosszhiszemű cyber-tér foglalóval szemben, mint valamilyen jogorvoslati úton eltiltatni a domain használatától a jogsértőt és megszerezni a maga számára a kért domaint.
A hagyományos bírói út a világ más részein, a Lajtán és az Atlanti óceánon túl is éppoly lassan nyújt eredményt, mint a Kárpát medencében. Az Internet világa viszont éppen ezt az egyet nem tűrheti, hiszen az eleve a minden eddiginél gyorsabb információáramlásra épülő rendszerben a bírói döntés megszületéséig olyan mérvű jogsértés történhet, amelynek jóvátétele már szinte alig lenne lehetséges, és a jogosult részéről is érdekmúlás következhet be. Fel kell tehát kínálni egy olyan alternatív utat, amely gyors megoldást nyújthat, még ha ideiglenes jelleggel is – még ha csak néhány szempontra helyezzük is a vizsgálódást.
Valamennyi gTLD regisztrátor elfogadta a hosszú egyeztetések, nehéz küzdelem eredményeként megszületett egységes vitarendezési eljárást *(Uniform Domain Name Dispute Policy ill. Rules for Uniform Domain Name Dispute Resolution Policy)*. Ez azt jelenti, hogy ha egy adott védjegy jogosultja a rosszhiszemű domainigénylővel szemben az e rendszer keretében a WIPO mellett felállított testület döntését kéri, a döntést a regisztrátor végrehajtja, és ha az a domain átruházását rendeli el, úgy a kérelmezőre átruházza.
Az eljárás nem választott bíróságot hozott létre. A választott bíróság ítélete a törvény erejénél fogva jogerős, végrehajtható, az állami végrehajtási rendszer útján kikényszeríthető. Ellene jogorvoslatnak helye nincs, és csak nagyon szűk körben, súlyos eljárási szabálysértés esetén van arra mód, hogy az ítéletet hatályon kívül helyeztessék.

Az adminisztratív panel

Az ICANN egységes eljárásában a fő szerep az ún. adminisztratív panelnek jut, amely a felek választása szerint 1-3 főből áll. Döntését a regisztrátor végrehajtja, kivéve, ha a fél igazolja, hogy a döntés kézhezvételétől számított 10 napon belül bírósághoz fordult. Itt tehát nem jogerős és állami erővel végrehajtható határozat születik, hanem egy olyan döntés, amelynek végrehajtását az önkéntes alávetés garantálja, és amely után még a rendes vagy választott bírói út a maga teljes egészében nyitva áll a felek előtt.
Az adminisztratív panel tagjait egy listáról jelölik ki. Az eljárás írásban folyik, nincs személyes meghallgatás, és a kérelem beérkezésétől számított 45 napon belül befejeződik. A határidők rendkívül szorosak és betartásukat a döntés adminisztrációját intézők nagyon szigorúan veszik. Így például a kérelmezett fél által elkésetten előterjesztett védekezést már nem veszik figyelembe a döntés során.
Az eljárás kérelemre indul, amelynek az előírt tartalmi jegyekkel rendelkeznie kell. A kérelemből megállapíthatónak kell lennie, hogy a kérelmező jogosult az adott domain névre.
Az eljárás díja 1000 – 3000 USD közötti összeg, amely a magyar szokásokhoz képest meglehetősen magas, de ha az egyes domainek piaci értékét, és a döntés gyorsaságát nézzük, akkor megéri megfizetni, és amint ezt a statisztika bizonyítja sok esetben meg is teszik. Ez különösen abból a szempontból érdekes, hogy – a bírósági eljárásokkal szemben – itt a „vesztes" fél nem köteles megtéríteni utóbb az eljárási díjat a kérelmezőnek. Közvetve következtethetünk tehát arra, hogy az egyes domainek átruházásáért mekkora ellenértéket kérnek a piacon.
A kérelmező javára szóló döntéshez az alábbi feltételeknek kell együttesen fennállniuk:

- a domainnév azonos vagy összetéveszthetőségig hasonló egy áru vagy szolgáltatási védjegyhez, amelynek a Kérelmező a jogosultja, és
- a regisztrált használónak nincs jogcíme vagy jogszerű igénye *(érdeke)* a domain névhez, és
- a domainnevet rosszhiszeműen regisztráltatta és használta

E felsorolásból láthatjuk, hogy ez az eljárás csak akkor vehető igénybe, ha kifejezetten védjegyjogot sért a domainregisztráció. Nem járható út tehát ha a versenyjogi szabá-

lyok, személyiségi jog, stb. sérelmére alapítja valaki a kérelmét. Ez esetben tehát marad a rendes bírói út.

Ugyanakkor a védjegyjogosult jogával szemben eredményes védekezés lehet, ha a kérelmezett a domainnevet:

- már annak nyilvántartásba vétele előtt használta a saját áruja vagy szolgáltatása körében, vagy erre bizonyíthatóan előkészületeket tett
- ha a domainnév megegyezik azzal a névvel amelyen már korábban magánszemélyként vagy gazdálkodó illetve más szervezetként ismerték *(pl. cégként ezen a néven bejegyezték)*
- a domainnevet nem kereskedelmi jelleggel, rendeltetésszerűen használja, anélkül, hogy célja lenne a védjegy jogosultjának potenciális vásárlóinak eltérítése, vagy a szóban forgó védjegy megkülönböztető jellegének elhalványítása

A vitás ügyek lezárása

A döntés a következő változatok egyike lehet:

- az adminisztratív panel megállapítja, hogy a domaint rosszhiszeműen regisztrálták, és annak törlését rendeli el
- az adminisztratív panel megállapítja, hogy a domaint rosszhiszeműen regisztrálták és annak a kérelmező javára történő átruházását rendeli el
- az adminisztratív panel a kérelmet elutasítja

Külön figyelmet érdemel, hogy az egységes eljárási szabályok mit tekintenek a rosszhiszeműség bizonyítékának. Önmagában megállapítható a rosszhiszeműség, ha annak átruházása ellenértékeként akkora összeget kérnek, amely a regisztrálással ténylegesen felmerülő költségeket meghaladja. Ugyancsak rosszhiszemű a regisztrálás, ha annak célja az volt, hogy a védjegy jogosultját megakadályozza abban, hogy a saját védjegyét domain neveként feltüntesse, vagy ha a cél a versenytárs üzletének megzavarása volt, vagy ha a domainnevet a kérelmezővel való összetéveszthetőség céljából a piaci előny megszerzéséhez kívánták használni úgy, hogy a honlapon vagy más on-line lapon az azonosság, összefonódás látszatát keltették.

Sok esetben megállapítják a rosszhiszeműséget akkor is, ha a kérelmező csak azt igazolja, hogy a domain azonos vagy az összetéveszthetőségig hasonló az ő védjegyével, és a kérelmezett fél nem terjeszt elő semmilyen védekezést. A bizonyítás ugyanis a kérelmezett felet terheli. Ugyanakkor ismertek olyan döntések is, amelyeknél egy ügyes védekezéssel el lehetett érni a panasz elutasítását.

Mindenképpen érdemes egy .com, .org. vagy .net domain jogvita kezdeményezése előtt, vagy ha abba kérelmezettként kerül az ember, hozzáértő ügyvéddel konzultálni. A magyar érdekeltségű ügyek közül ismert olyan, ahol az ügyes ügyvédi trükk segített megtartani a nyilvánvalóan jogsértő domaint. Itt a kérelmező a Casino Monte-Carlo volt, amelynek védjegyéhez nagyon hasonló domaint egy magyar betéti társaság regisztráltatott. Az eljárás megindítását követően azonban a domaint átruházta egy másik társaságra, amelynek tevékenységi körében szerencsejáték-szervezés is volt, és amely arra hivatkozott, hogy e cím alatti honlapon szerencsejátékot kíván szervezni. Így a rosszhiszeműséget nem tartotta megállapíthatónak az eljáró panel és a kérelmet elutasította.

Az eddigi statisztikák szerint egy magyar kérelmező fordult a panelhez, a Zwack Unicum Rt., amely a zwackunikum.com domain átruházását kérte. A kérelmezett nyilvánvalóan rosszhiszemű eljárását az is bizonyította, hogy az említett domainnév alatt a Jägermeister – a közismert versenytárs – reklámja volt látható.

Az ismertetett eljárás tehát a nemzetközi domainpiacon működik. Nem terjed ki azonban a ccTLD-k alatt regisztrált domainekre, így a .hu alattira sem. Ennek oka nyilvánvalóan abban rejlik, hogy amíg a nemzetközi jogviták esetében azok jellemzője a jogok közötti állapot, vagyis az alkalmazandó jogot az ide-oda utaló szabályok alapján kell megállapítani, addig az országnevek alatt egyértelműen a hazai jog szerint kell a jogvitát eldönteni. Itt viszont a regisztrátorok felelőssége, az igények jó és rosszhiszeműsége más elvárások és más szabályok szerint kerül elbírálásra.

A jövő nyilván e területen is a közelítés, a harmonizáció lesz. Addig azonban be kell érnünk az ahány ház, annyi szokás megoldással.

Sértettek és bitorlók

A **[2]** címen végzett kutatás érdekes eredményeket hozott a tavalyi év összesen 1841 ügyének megoszlásában. Vannak ugyanis csaló, domainbitorló országok. A sértettek tizes toplistáján *(egy kivétellel)* természetesen a legfejlettebb országokat találjuk *(USA, Egyesült Királyság, Franciaország, Spanyolország, Németország, Ausztrália, Japán, Svájc és kakukktojásként: India)*, addig a bitorlók listáján a negyedik helyet Dél-Korea foglalja el, a hatodik Kína. Kis hazánk a becsületes népek táborát gyarapítja, mivel míg sértettként a 45. helyen áll, a bitorlók listáján már csak ötvenedik.

Dr. Mayer Erika
Nádas & Mayer Ügyvédi Iroda
mayer@nadas-mayer.hu

A cikkben szereplő URL-ek
[1] http://www.icann.org
[2] http://www.wipo.org

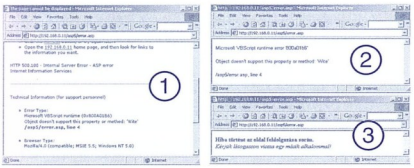
ASP suli – Hibakezelés *(V. rész)*

Hibátlan szoftver – mint tudjuk – nem létezik. Ez a tétel igaz az ASP alkalmazásokra is. Ma az ASP programozás során fellépő hibák hatékony kezeléséről és elkerüléséről lesz szó.

Rendezzük a felszínt

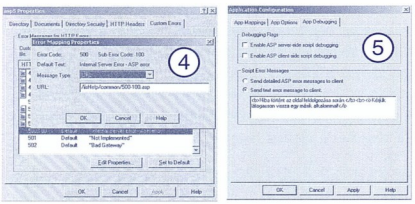
Az ASP alkalmazásokban fellépő hibák rendszerint kellemes hibaüzenet formájában jelennek meg a felhasználónál. Természetesen nem feltétlenül egészséges, ha egy-egy ilyen hibaüzenet, esetleg kódrészlet nyilvánosságra kerül, ezért megvan a módja annak, hogy – miközben mi a hiba kezelését végezzük – a felhasználót megkíméljük a kínos részletektől.

Az IIS alapértelmezése szerint az alábbi ábrán ① jellel jelölt hibaüzenet-oldal jelentkezik a felhasználónál, ami egyrészt praktikus, mert szép és színes, másrészt kellemetlen lehet, hiszen tartalmazza a hiba komplett leírását. Ezt az oldalt egyébként a \winnt\help\iisHelp\common\500-100.asp állítja elő.



Egy hiba három arca. Az ① jelű az alapértelmezés, a ② egy köztes állapot, a ③ végfelhasználóknak szánt változat

Miközben nyilván nagyon hasznos a részletes hibaüzenet, éles környezetben ez nem feltétlenül igény. Az ASP scriptek végrehajtása során fellépő hibák esetén a teendőt a virtuális könyvtár tulajdonságlapjának Custom Errors oldalán az 500;100 sorhoz tartozó beállítása határozza meg. Amint az az alábbi ábrán is látható ④, alapértelmezésben a fentebb említett hibakezelő oldalra adódik a vezérlés.



Hibakezelési beállítások a webalkalmazásban. Az ⑤ beállítás csak akkor érvényesül, ha a ④-et alapértelmezésre (Default) állítjuk

Ezt a beállítást átirányíthatjuk valami saját magunk által készített hibakezelő oldalra *(ami mondjuk naplózza is a felmerült hibákat, ilyet fogunk létrehozni később)*, vagy választhatjuk az alapértelmezést is *(a Default beállítás, illetve Set to Default... gomb segítségével)*. Amikor az 500;100 hibához nincs beállítva külön hibakezelő fájl, a webalkalmazás beállításai között, az App Debugging oldalon található beállítás ⑤ az irányadó. Ha ezt a beállítást az első lehetőségen hagyjuk *(„Send detailed error messages to client")*, akkor a ② példához hasonló hibaüzenetek jelennek meg a böngészőkben. Ezek nem kevésbé bőbeszédűek, mint az alapértelmezett hibakezelő oldal, viszont jóval rondábbak: egyszerűen a kliens arcába vágják a hibát.

Ha viszont a másik opciót választjuk, megadhatunk egy saját hibaüzenetet *(ami tartalmazhat HTML elemeket is, ld. az ábrán)*. Végfelhasználói környezetben ez az egyik legegyszerűbb és legjobb megoldás, hacsak nem akarunk saját hibakezelő oldalt írni.

Írjunk saját hibakezelő oldalt!

Ehhez minden támogatást megkapunk az Internet Information Server-től: egyrészt a hiba esetén történő átirányítás lehetőségét *(ezt az előbb már láthattuk)*, másrészt pedig a hiba felismeréséhez, feldolgozásához szükséges információkat, az ASPError objektum formájában.

Miután a megfelelő könyvtárban beállítottuk a hibakezelő oldalt a sajátunkra, kezdjünk neki a munkának. Mindenekelőtt, ha a pufferba már írtak, amikor a hiba bekövetkezett, itt az ideje, hogy kiürítsük, és a hibakezelő oldal tiszta lappal indulhasson:

```
<%
  If Response.Buffer Then
    Response.Clear
```

Azután: a válasz státuszát állítsuk be hibaüzenetre, nehogy valaki azt higgye, hogy ez a valódi oldal:

```
    Response.Status = "500 Internal Server Error"
```

Majd állítsuk be az oldal tartalmát HTML-re, a lejárati időt 0-ra *(így a gyorsítótárak nem fogják az oldalt letárolni)*:

```
    Response.ContentType = "text/html"
    Response.Expires = 0
  End If
```

A következő lépés a hibát leíró ASPError objektum előcsalogatása, amit a következőképpen tehetünk meg:

```
Set objASPError = Server.GetLastError
```

Miután az objektum megvan, belekezdhetünk a jellemzők lekérdezésébe. Az egyes jellemzők jelentése a következő:

- .Category: a hiba kategóriája, szöveges jellemző, pl. „Microsoft VBScript compilation"
- .Number: a hiba kódja, hexadecimálisan megjelenítve sokatmondó lehet *(rá lehet keresni a tudásbázisban)*
- .Description: a hiba leírása, maga a hibaüzenet
- .File: a fájl neve, ahol a hiba keletkezett
- .Line, .Column: a hiba felismerésének sora és oszlopa *(ez az adat nem mindig áll rendelkezésre, ilyenkor az egyes jellemzők értéke –1)*. Vigyázzunk, a hiba felismerésének helye nem feltétlenül egyezik meg a hiba helyével!
- .Source: a hibát okozó sor kódja. Nem mindig áll rendelkezésre
- .ASPCode, .ASPDescription: a hiba ASP hibakódja és leírása – viszonylag ritkán kapnak értéket

Állítsunk össze ezekből az adatokból egy hibaüzenetet, egyenlőre egy szöveges változóban, mint ahogy az az errhandler.asp példafájlban is látható **[1]**. Azért ne írjuk ki a képernyőre, mert egyáltalán nem biztos, hogy a felhasználóra tartozik a hiba leírása, jellege és főleg a helye. Miután a hibaüzenetet összeállítottuk, eldönthetjük, hogy kiírjuk-e a böngészőbe, vagy egy illendő bocsánatkérő szöveg keretében elintézzük a dolgot a színfalak mögött. Én azt a megoldást választottam, hogy a felhasználó láthatja a hibaüzenetet, de csak akkor, ha a böngésző a kiszolgálón fut, azaz a kliens és a szerver IP címe megegyezik. Az ellenőrzés pillanatában a hibaüzenet szövegét már az sErrorText változó tartalmazza:

```
<%
  If Request.ServerVariables("LOCAL_ADDR") = _
  Request.ServerVariables("REMOTE_ADDR") Or _
  Request.ServerVariables("REMOTE_ADDR") = _
  "127.0.0.1" Then
%>
  <HR>
  <PRE>
<% Response.Write Server.HTMLEncode(sErrorText) %>
  </PRE>
<% Else %>
  <P>A hibát naplóztuk. Kérjük, próbálkozzon újra
néhány perc múlva.</P>
<% End If %>
```

Vegyük észre a hibaüzenet kiírásánál használt Server.HTMLEncode() függvényt. Mint mindig, ismeretlen szöveg kiírásánál használnunk kell, különben az esetleg *(scriptkódban nagy eséllyel)* előforduló speciális karakterek megzavarhatják a HTML kódot – gondoljunk csak néhány ártalmatlan kacsacsőrre, ami scriptkorábban még matematikai műveletet jelképezett, most pedig félkész HTML elemeknek értelmezněnk őket.
Következő lépésként a korábban bemutatott módszerrel a hibát eltároljuk az Eseménynaplóba:

```
Set oShell = Server.CreateObject("WScript.Shell")
oShell.LogEvent 1, "ASP HIBA!" & vbCRLF & sErrorText
Set oShell = Nothing
```

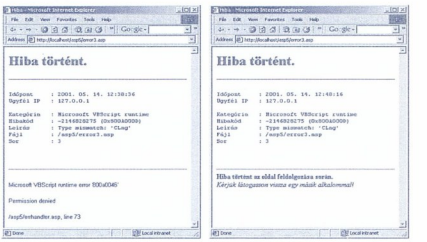
Végül, biztos, ami biztos, a hibát naplózzuk le egy közönséges szövegfájlba is:

```
Set oFSO = Server.Createobject
↳ ("Scripting.FileSystemObject")
Set oFile = oFSO.OpenTextFile
↳ ("C:\asperror.txt", 8, True)
oFile.Write sErrorText & vbCRLF & vbCRLF
oFile.Close
Set oFSO = Nothing
```

Ezzel készen is vagyunk. Felmerülhet *(és remélem, van már, akiben a kód olvasása során fel is merült)*, hogy a kód így nem teljes: mi történik például akkor, ha a naplófájlt valamilyen okból nem lehet megnyitni *(például mert éppen más valaki ír bele)*? Ilyenkor természetesen hiba keletkezik.

Amikor a hóhért akasztják

A hibakezelő rutinban keletkező hiba megoldása klasszikus feladat. Ilyenkor a hibakezelő oldal már nyilván nem képes a szálakat tovább a kezében tartani, szükség van egy felsőbb „hatalom" beavatkozására. Esetünkben ez a felsőbb hatalom maga az IIS, aki ilyenkor legjobb tudása szerint a felhasználó elé hinti a hibát, valahogy így:



Hiba a hibakezelőben (a képek alján): ilyenkor az IIS hibakezelési beállítása érvényes (ld. az előző oldalon: ⑤)

Hibakezelés futás közben

A hiba nem mindig végzetes: az esetek többségében fel tudunk készülni arra az esetre, ha valami „baj" történne – a lényeg csak a hiba biztonságos felismerése. A scriptnyelvek éppen ezért általában beépített hibakezelési elemeket tartalmaznak, amelyeket használva a kisebb hibákat még a nagyobb problémák bekövetkezte előtt kivédhetjük. A futás közbeni hibakezelő eszközök közös tulajdonsága, hogy a hiba kezelése után a program futtatása folytatódik.
A futás közbeni hibakezelés már eléggé scriptnyelv-specifikus, cikkünk keretében a két beépített nyelv, a VBScript és a JScript elemeivel ismerkedhetünk meg.
Kezdjük a VBScript-tel: itt a futás közbeni hibakezelést mindenekelőtt be kell kapcsolni, ha ezt nem tesszük meg, mindenképpen hiba keletkezik. A hibakezelés bekapcsolására a következő parancs használható:

```
On Error Resume Next
```

Azaz kb. „Hiba esetén folytasd a végrehajtást". Ezután rajtunk áll, hogy kijavítjuk-e a hibát, vagy egyáltalán foglalkozunk vele. Mindaddig amíg ez a parancs érvényes *(márpedig abban a procedúrában ahol kiadtuk, valamint az abból hívott procedúrákban, érvényes)*, a hiba miatt a VBScript nem fog leállni. Éppen ezért a fenti parancsot globálisan kiadni nem túl ésszerű, a hibakezelés bekapcsolását korlátozzuk kényes helyekre –

oda, ahol fel tudunk készülni a hibák javítására.
A hiba kezelésére *(és felismerésére)* az Err objektum használható, amelynek fontosabb jellemzői a következők:

- .Number – a hiba kódja, az Err objektum talán legfontosabb jellemzője. Értéke 0, ha minden rendben van, ettől eltérő, ha hiba történt.
- .Description – a hiba szöveges leírása
- .Source – a hibát „okozó" objektum azonosítója

Az előző példát tehát ki kell bővíteni:

```
On Error Resume Next
  '… hibát okozó rész
If Err.Number <> 0 Then
  ' Hiba a hibakezelőben…
  Err.Clear()
End If
```

Miután egy hibát kezeltünk, az Err objektumot vissza kell állítani a kezdőértékekre, különben a hibát a következő ellenőrző rutin is elkapná. Az Err.Clear() metódust pontosan erre találták ki. Mi is meghívhatjuk *(célszerűen a hibakezelő rutin végén)*, de a VBScript automatikusan meghívja a következő utasítások végrehajtása során:

- On Error Resume Next
- Exit Sub
- Exit Function

... azaz, a procedúrákból, függvényből történő visszatéréskor, valamint a hibakezelés bekapcsolásakor.
Ha elegünk volt a kánaánból, a futás közbeni hibák kezelését visszakapcsolhatjuk az

```
On Error Goto 0
```

paranccsal. Ilyenkor hiba esetén ismét a jól ismert hibakezelési megoldások veszik át az irányítást.
Az objektumnak van még egy érdekes metódusa, ez pedig az Err.Raise(). Ez a függvény arra való, hogy hibát váltsunk ki a rendszerben. Az Err.Raise() által generált hibák teljes értékűek: próbáljuk ki a raise.asp példaprogram segítségével!

try...catch...finally

A JScript hibakezelése másképp működik, inkább hasonlít a C nyelvben található try...catch utasításpárra. A szintaxis a következő:

```
try {
  // hibát okozó művelet
}
catch(hibaobjektum) {
  // hibakezelés
}
finally {
  // mindenképp lefutó utasítások
}
```

JScriptben tehát minden hibát okozó művelet*(-csoport)* esetén külön fel kell készülnünk az esetleg előforduló hibák kezelésére, ami azt is jelenti, hogy minden egyes esetben külön meg kell írnunk a hibakezelő rutint is. A gyanús sorokat a try{} blokkba kell írni. Ha az utasítások végrehajtása során bármilyen hiba keletkezne, a végrehajtás a catch{} blokkban folytatódik. A catch paramétereként megadott változó tartalmazza a hiba leírását, egy Error objektumot. Példaképpen lássuk, hogyan kaphatjuk el a FileSystemObject által visszaadott hibát *(jserror1.asp)*:

```
var oFSO = new ActiveXObject
↳ ("Scripting.FileSystemObject");
try {
  oFSO.CopyFile("qweqweq.txt", "eqwewqe.txt");
}
catch(e) {
  Response.Write("Error: " + e);
  Response.Write("<br>Error Number: " +
↳   (e.number & 0xFFFF) );
  Response.Write("<br>Description: " +
↳   e.description);
}
```

A hiba oka nyilvánvaló: az aktuális könyvtárban valószínűleg nincsen „qweqweq.txt" nevű fájl, amit másolhatnánk, ezért hiba történik. A catch-ben megkapott változó egy Error objektumot tartalmaz, erről már az első sorban kiírt [object Error] is tájékoztat minket. A JScript Error objektumának két jellemzője van: az Error.number a hiba kódját, az Error.description a hiba leírását tartalmazza. A hibakód előcsalogatásához az Error.number által visszaadott számot „dekódolnunk" kell, erre való a példában látott *(Error.number & 0xFFFF)* kifejezés.
A try...catch...finally hármas utolsó tagjának az az értelme, hogy olyan utasításokat is megadhassunk, amelyek lefutnak függetlenül attól, hogy a try blokkban történt-e hiba, vagy sem. Hiszen gondoljunk bele: hiba esetén a try blokkból azonnal kiugrunk, az esetleg hátralévő utasítások mennek a süllyesztőbe. Ez a helye az erőforrások, például adatbáziskapcsolatok felszabadításának, lezárásának.
JScriptben is idézhetünk elő hibát: a throw() függvény hívásával hasonló eredményt érünk el, mint a VBScript-beli .Raise metódussal. A throw() paramétere a hiba maga, ami lehet egy hibakód, egy szöveg, vagy egy előzőleg előállított, és megfelelően „kitöltött" Error objektum is *(jserror2.asp)*:

```
var oErr = new Error();
oErr.number = 1234;
oErr.description = "User Error";
try {
  throw(oErr);
}
catch(e) {
 …
```

Ha a catch blokkban, a hiba kezelése során zsákutcába jutunk, és feladjuk a harcot, a munkát tovább passzolhatjuk az eggyel magasabb szinten található hibakezelőnek a throw() függvénnyel. Paraméterként adjuk neki a catch()-ben átvett változót, valahogy így *(jserror3.asp)*:

```
catch(e) {
   Response.Write("Error: " + e);
   Response.Write("<br>Error Number: " +
   ↳   (e.number & 0xFFFF) );
   Response.Write("<br>Description: " +
```

```
    e.description);
  throw(e);
}
```

A második throw() hívás segítségével a hibát tovább passzoltuk az IIS-nek, küzdjön vele ő. A küzdelem eredménye a beállításoknak megfelelő IIS hibaüzenet lesz, mintha csak nem is lett volna a try...catch blokk.

Hibakeresés – debugolás

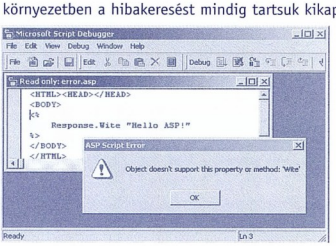
A futás közbeni hibák néha váratlanul jönnek, nem kerülnek elő egyszerűen, de az is előfordulhat, hogy a fejlesztés során szeretnénk akár soronként szemmel kísérni a kód futását, a változók értékét. Régebben ezt legfeljebb úgy tehettük meg, ha teleaggattuk a kódot debug üzenetekkel, a program köpte a funkciótól független információkat, csak győztük kiválogatni, mi a valóság és mi a debug által megjelenített adat.

Fejlettebb programozási környezetben szinte mindenhol megtalálható a debugger, mellyel végrehajtás közben ellenőrizhetjük a futási paramétereket, követhetjük a program menetét, sőt, még bele is avatkozhatunk a történésekbe. A Microsoft Script Debugger minden új Windows része, segítségével lehetőségünk van kiszolgálóoldali .asp, böngészőoldali .html, valamint parancssorból futtatott scriptek debugolására is, függetlenül a használt scriptnyelvtől.

Vannak persze sokkal fejlettebb eszközök *(gondoljunk csak a Visual Studio kifejezetten erre a célra fejlesztett komponensére, a Visual InterDev-re)*, de ez mindig rendelkezésre áll, ráadásul nem kerül külön pénzbe. Fontos, hogy a Script Debugger telepítése után az adott webalkalmazás beállításai között engedélyezzük a kiszolgálóoldali hibakeresést.

A hibakeresés engedélyezése a webalkalmazások beállításai között

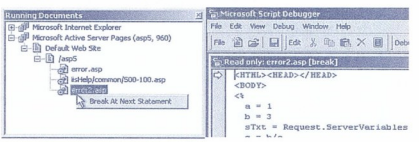
A Script Debuggert háromféleképpen állíthatjuk munkába. Mindenekelőtt, a hibakeresés engedélyezése után az oldalban történt hiba esetén nem a megszokott hibakezelő oldalak futnak le, hanem a kiszolgálón elindul a Script Debugger, és a végrehajtás mindaddig nem folytatódik, amíg valaki a dologra áldását nem adja – ezért fontos, hogy éles környezetben a hibakeresést mindig tartsuk kikapcsolva.



Hiba esetén már indul is a Script Debugger

A Script Debugger a hiba felderítésére, és nem a script szerkesztésére való, ezért senkit se lepjen meg a fejlécben olvasható „Read only" felirat. A debugger – a script kódját tartalmazó mellett – három különböző ablakot tartalmaz még. Az egyik a Call Stack ami a hívásverem aktuális tartalmát jelzi *(gyakorlatilag azt, hogy hogyan kerültünk a hibához)*. A második, talán kicsit hasznosabb ablak a Command Window, amibe futás közben parancsokat gépelhetünk *(például új értéket adhatunk a változóknak, meghívhatunk függvényeket, stb.)*. Ha egy változó vagy függvény értékére vagyunk kíváncsiak, a kifejezés elé írjunk egy ?-et *(ami a klasszikus BASIC-ben egyébként a Print parancs rövidítése volt)*. Ha a hibát JScript-ben keressük, a ? használatára nincs szükség. Ha nemcsak hiba esetén, hanem általában is szeretnénk a debuggert használni, két lehetőségünk van: az első, hogy a debugger indítására szolgáló utasítást helyezünk el a kód belsejében. Ez a parancs VBScript esetén a Stop, JScript kódban pedig a debugger; *(ld. debug1.asp, debug2.asp)*. A parancs hatására a script végrehajtása felfüggesztődik, elindul a debugger, és a kurzor a parancs sorára áll.

A másik lehetőség az, hogy a Script Debugger elindítása után a Running Documents ablakban kijelöljük a kívánt scriptet, majd ráuszítjuk a Break at Next Statement parancsot. Amikor legközelebb a végrehajtás az adott scriptre kerül, elindul a debugger és kezdhetjük a munkát.



Árgus szemekkel várunk a scriptre

A Running Documents ablakba azok a scriptek kerülnek be, amelyek a böngésző *(kliensoldali)* illetve az IIS *(szerveroldali)* motorjába már betöltődtek. Az ASP hibakeresést értelemszerűen a Microsoft Active Server Pages csomópont alatt keressük. Ahhoz, hogy egy script itt megjelenjen, be kell tölteni, a hibakeresés módja tehát a következő:

- Internet Explorer-be töltsük be a kívánt oldalt
- indítsuk el a Script Debugger-t, keressük meg a scriptet és adjuk ki a Break parancsot
- a böngészőben hívjuk le újra az oldalt *(Refresh)*

A Refresh hatására újra betöltődő .asp scriptet pedig már képes elkapni a debugger és – mint az az ábrán is látszik – a végrehajtás rögtön a script elején a debugger kezébe kerül.

Fülöp Miklós
mick@netacademia.net

A cikkben található URL-ek:
[1] http://technet.netacademia.net/feladatok/asp/5

MIME – levelezés az interneten *(II. rész)*

Az első részben láthattuk, hogy az ősrégi SMTP szabvány és formátum képezi az Internetes levelezés alapját. Ezek a szabványok annyira jól sikerültek, hogy – bár jelentős mennyiségű bővítmény megjelent már azóta, de – a mai napig nem szakítottunk az SMTP hagyományaival. Az alapokat már megismertük, következhetnek a későbbi fejlesztések.

Multipurpose Internet Mail Extensions

Szó esett már arról, hogy a tisztán szöveges levelek küldése mellett egyre nagyobb lett az igény az internetes levelezés más célú felhasználása iránt is. A UUENCODE/UUDECODE ugyan rövid ideig megoldásnak tűnt, de csak arra volt képes, hogy az amúgy bájtonként nyolc bites adatot át tudjuk tuszkolni az SMTP hagyományosan 7 bites csatornáján. Ez a feladat azóta is áll, de megjelentek olyan igények is, amelyek kielégítésére a UUENCODE már nem volt képes. Például:

- üzenetek küldése nem US-ASCII kódolással *(ékezetes betűk, ne adj' Isten távol-keleti nyelvek használata – hiszen Internet ott is van)* – sőt, esetleg az üzenet különböző részeiben különböző kódtáblák használata
- nem szöveges üzenetek *(de legalábbis nem tiszta szöveg, hanem RTF, SGML, esetleg HTML)*
- több részből álló üzenetek *(esetleg ugyanaz az üzenet több formátumban, amiből mindig csak egyet kell megjeleníteni)*
- nem ASCII kódolás használata a fejlécmezőkben *(például ékezetes betűk az üzenet témájában, a feladó és címzett nevében)*

Mindezt úgy kellett megoldani, hogy a – nyilván kódolás utáni – eredmény egy-az-egyben RFC 822-kompatibilis, azaz szigorúan 7 bites, ASCII karakterek, rögzített sorhosszúság és más paramétereknek megfelelő legyen.

Sikerült. Az eredmény a MIME *(Multipurpose Internet Mail Extensions, azaz kb. többcélú internetes levelezési bővítmények)* szabványcsalád, jónéhány RFC-ben megfogalmazva. A MIME egyébként „csak" egy jól megtervezett keretrendszer, ami azóta is bővül, de a tervezői profizmusára utal, hogy a mai napig minden bővítményt bele lehet csomagolni.

Fejléc-paraméterek

Az előző részben már bemutattuk az SMTP fejléc felépítését *(név: érték)*, a fejléc több sorba tördelésének lehetőségét és módját. Egy „apróság" azonban nem került sorra, mégpedig az, hogy a fejlécérték több paramétert is tartalmazhat, pontosvesszővel elválasztva, valahogy így:

```
X-MyFejlec:fejlecertek;
        param1=parametererertek1;
        param2=parametererertek2
```

A fenti példában tehát az X-MyFejlec értéke *fejlecertek*, az így kialakuló *fejlecertek* mező *param1* paramétere *parametererertek1*, *param2* paramétere pedig *parametererertek2*. A paramétereket most külön sorba választottuk el *(ezt az RFC 822 szabályai szerint megtehetjük)*, de ez ne zavarjon meg senkit, ettől a példa mindhárom sora az X-MyFejlec értékét jelképezi.

A Content-Type fejléc

A MIME egyik legelső és legfontosabb jellemzője az üzenetek fejrészében megjelenő Content-Type fejléc. E fejléc értéke határozza meg, hogy a levél milyen tartalmat hordoz, hogyan kell azt feldolgozni. A fejléc értéke mindig egy úgynevezett MIME típus *(MIME type)*, valamint a típushoz esetleg hozzátartozó paraméterek. A leggyakoribb MIME típusokat mi is bemutatjuk, de az IETF által regisztrált típusok az RFC 2046-ban találhatók meg. A regisztrációs procedúráról az RFC 2048 és RFC 2049 szól. Saját MIME típusokat is definiálhatunk, ezek az X-akármi névre hallgatnak. A saját típusok használata az Interneten nem tilos, csak arról kell*(ene)* gondoskodnunk, hogy a levél feladója és címzettje egyaránt ismerje az adott tartalomtípust.

A charset paraméter

A MIME egyik újdonsága, mint tudjuk, a különböző kódtáblák használata volt. A kódtáblák kialakulásáról, használatuk szükségességéről most nem szólnék, akit érdekel, az Interneten utánanézhet *(de érdemes ezügyben beleolvasni a tech.net magazin márciusi számában található ASP suli 3. cikkbe)*. A lényeg, hogy a világon szerte használt kódtáblák közül nekünk a közép-európai, más néven ISO-8859-2, illetve a Unicode karakterek kódolására használt UTF-8 a kedves. Szóba jöhet még a nyugat-európai *(ISO-8859-1)* is, de ebben a karaktertáblában a szép magyar ő és ű betűk helyett csak a hullámos/kalapos változatot találjuk meg.

A charset paraméter tehát bárhol jelenik meg, a tartalom KÓDTÁBLÁJÁRA utal – ez nem keverendő össze a tartalom 7 bitessé konvertálását végző kódoló ALGORITMUSOKKAL.

A Content-Transfer-Encoding fejléc

A Content-Encoding fejléc a tartalom kódolásának módját jelzi. Az eredeti RFC 822 ugyanis – mint tudjuk – a hét bites ASCII kódkészleten kívül mást nem nagyon hajlandó átvinni, ezért ha ennél többet szeretnénk, kódolásra kényszerülünk. Hiába használnánk például közép-európai kódtáblát, az üzenetet nem írhatjuk meg „csak úgy", hiszen az ékezetes karaktereink jóval a hét bites határ felett vannak. Ugyanez a helyzet a bináris adatokkal, képekkel, futtatható fájlokkal: ezeket mind-mind kódolnunk kell. Kódoláshoz többféle algoritmus is ismert, a Content-Transfer-Encoding tehát a következő értékeket veheti fel:

- 7bit = Az alapértelmezés, az adat kódolás nélkül kerül az üzenetbe *(ennek feltétele, hogy ne tartalmazzon 127 feletti karaktereket és a 00-t)*. Soronként legfeljebb 1000 karaktert küldhetünk.
- 8bit = 8 bites, kódolás nélküli átvitel. Ez lenne a tel-

jesen kézenfekvő, az SMTP gyökerei miatt azonban az Interneten használhatatlan. További korlátozás, hogy érvényes a soronkénti 1000 karakteres határ.
- binary = 8 bites, kódolás nélküli átvitel, soronkénti karakterhatár nélkül. Az Interneten értelemszerűen nem használható
- quoted-printable = ld. később
- base64 = ld. később
- x-akármi = saját kódolás, a Content-Type-hoz hasonlóan csak akkor használható, ha a feladó és a címzett egyaránt felismeri. Épp emiatt ez a megoldás széles körben nem terjedt el.

A fentiek alapján látható, hogy az alapértelmezett, ámde butácska 7 bites kódolás mellett két közismert alternatíva létezik: a quoted-printable, és a Base64. Bármi is a levél tartalma, MIME típusa, a 7 bitre kódolás e két módszer egyike lesz!
- **bináris adatok *(például képek)* hatékonyan Base64 kódolással változtathatók 7 bitessé**
- **szöveges jellegű infók tipikusan quoted-printable kódolást kapnak**

Sokakat megzavar a kódtábla *(például ISO-8859-2)*, és a kódolási algoritmus *(például Base64)* hasonló neve. Most akkor mi történik? Ha ISO-8859-2 kódolást „alkalmazok", akkor nem kell a levelet 7 bitesre konvertálni *(pl. Base64-gyel)*? De. A kódtáblák ugyanis nem alakítják 7 bitessé a levelet, sőt!, a Unicode egyenesen bináris bájtsorozattá varázsolja szövegünket! A kódtáblák a célalkalmazásnak segítenek az adatok megjelenítésében.
És tekintsünk úgy a Base64 és quoted-printable algoritmusokra, melyek bármit hálásan 7 bitessé konvertálnak: Unicode, ISO-8859-2 kódtáblájú szöveg GIF, HTML – egyre megy.

A quoted-printable kódolás

A quoted-printable rendszer nem minden karaktert kódol, csak azokat, amelyek nem esnek az ASCII karakterek tartományába. Az ASCII karakterek kódolatlanul, tisztán kerülnek be az üzenetbe, ezért a szöveg „nagyjából" olvasható. Az előző mondat például így fest kódolva:

```
Az ASCII karakterek k=F3dolatlanul, tiszt=E1n
 ker=FClnek be az =
=FCzenetbe, ez=E9rt a sz=F6veg "nagyj=E1b=F31"
 olvashat=F3.
```

Ugye ismerős? Talán kivehető, hogy a fenti egysoros *(sorvégjelet nem tartalmazó)* példából két sor lett. Ez azért van, mert a quoted-printable kódolás eredményeképpen keletkező adat soronként legfeljebb 76 karaktert tartalmazhat. A sortörést egy önmagában álló = jelzi. A kódolás lényege, hogy a nem ASCII karaktereket hexadecimális kódjukkal helyezi a szövegbe, ami elé egyenlőségjel kerül. A kis hosszú ő-ból így lett például =F3. Az egyenlőségjelet pedig =3D jelképezi. A NetAcademia kiszolgálójáról, a **[2]** címről letölthető egy demonstrációs script, ami képes quoted-printable kódolásra és dekódolásra is. A kódolás többlet „költsége" függ a valójában kódolt karakterek számától, ezért ez a kódolás tiszta bináris adat esetén pazarló *(szövegben nagyobb eséllyel találunk kódolatlanul elküldhető karaktereket, ezért ezt a kódolást leginkább szöveges jellegű adatokon használják)*.

A Base64 kódolás

A Base64 kódolás egészen más jellegű. Ilyenkor a kódolandó szöveg minden 3 bájtját *(3x8=24 bit)* négy darab hatbites részre bontjuk *(4x6=24 bit)*, majd a kapott négy értékhez kikeressük a hozzá tartozó jelet egy táblázatból *(ezt Base64 ABC-nek hívják)*. A táblázat 2^6, azaz 64 elemű, és csak olyan jeleket tartalmaz, amelyek egy SMTP levélben szerepelhetnek *(például ASCII karaktereket, de nincs köztük a fejléceket esetleg megzavaró kettőspont, stb.)*. Az „árvíztűrő tükörfúrógép" szöveg például Base64 kódolással így fog kinézni:

```
4XJ27Xp0+3L1IHT8a/ZyZvpy82fpcA==
```

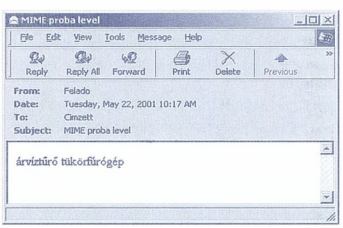
Az eredeti 22 karakterből 32 lett, hiszen minden három betűből négy készült. A kerekítések azért szerepet játszanak, de összességében el lehet mondani, hogy a Base64 kódolt adat hossza 4/3-szorosa az eredeti hossznak *(ezért csodálkozunk, ha a levelező kiszolgáló 1MB-ra állított korlátja nem engedi át a levélhez csatolt 1MB-os word dokumentumot...)*. Miután itt a költség állandó, szöveges jellegű adatok esetén jobb a quoted-printable kódolást választani. A **[3]** címről egy demonstrációs Base64 kódoló/dekódoló scriptet tölthet le a Kedves Olvasó.

Az első MIME példa

A fenti információk alapján már összeállíthatjuk az első példalevelünket. A NodePad tökéletesen alkalmas erre a célra. Készítsünk egy .eml kiterjesztésű fájlt *(az ilyen fájlokra kattintva elindul az alapértelmezett levelezőprogramunk és megjeleníti a levelet)*. A levélbe *(továbbra is a NotePad segítségével)* írjuk a következőt:

```
From: "Felado" <sender@mail.hu>
To: "Cimzett" <addressee@mail.hu>
MIME-Version: 1.0
Content-Type: text/plain; charset=iso-8859-2
Content-Transfer-Encoding: base64
Subject: MIME probalevel

4XJ27Xp0+3L1IHT8a/ZyZvpy82fpcA==
```



Ki hitte volna, hogy valaha Notepaddel írunk MIME levelet?

A fejlécek szerint ez egy tiszta szöveget tartalmazó levél, ISO-8859-2 kódtáblában írva, Base64 kódolással. Egy új fejlécről még nem beszéltünk:

```
MIME-Version: 1.0
```

Ez a fejléc szerepel minden MIME levél fejrészében, a levelezőprogram innen tudja, hogy a tartalmat a MIME szabályainak megfelelően kell feldolgoznia. Mielőtt rátérnénk a különböző tartalomtípusokra, maradjunk még egy kicsit az ékezeteknél: hogyan érjük el, hogy a fejlécekben *(feladó, címzett neve, levél témája)* se kelljen lemondani az ékezetekről?

Kódtáblák a fejlécekben

Az SMTP fejlécekben kicsit bonyolultabban kellett megoldani a kódolást. A kódolási megoldások persze megmaradtak *(Base64, Quoted-Printable)*, de ki kellett találni egy formátumot, aminek alapján az olvasó felismerheti, hogy kódolt formátummal áll szemben, és ami nem ütközik az SMTP fejlécekre vonatkozó szabályokkal. Arról nem is beszélve, hogy most fejlécenként meg kell mondanunk, hogy a következő adat milyen kódtáblában, milyen kódolással került bele a fejlécbe. *(Erről a témáról egyébként az RFC 2047 szól)*.
Mindenekelőtt: a kódolandó szöveget legfeljebb 75 karakteres részekre bontjuk, és minden részre külön értelmezzük az alábbiakat. A kódolt szövegrészeket azután egymás után, sortöréssel elválasztva *(a több sorba tördelt fejlécek szabályait betartva)* írjuk bele az üzenetbe.
A kódolt szövegrész a következő formátumú, legfeljebb 75 karakter hosszú:

```
"=?" charset "?" encoding "?" encoded-text "?="
```

... ahol a charset a használt kódtábla típusa *(pl. iso-8859-2)*, az encoding a kódolás azonosítója *(b vagy B a Base64, q vagy Q a quoted-printable jele)*, végül az encoded text maga a kódolt adat. A kódolt szövegben nem szerepelhet kódolatlanul a szóköz, tabulátor, vagy sorvégjel. A quoted-printable kódolásban a szóközt a _ *(alulvonás)* is helyettesítheti. Szilíciumvölgyi Fű Benő például így néz ki, ha levelet küld:

```
=?iso-8859-2?q?Szil=EDciumv=F6lgyi_F=FB_Ben=F5?=
```

A példára pillantva rögtön látszik, hogy a használt kódtábla a közép-európai *(ISO-8859-2)*, a kódolás quoted-printable *(q)*, valamint látható, hogy a nevek közötti szóköz alulvonássá változott. Ezt a kódolási módszert a címzésekben *(feladó, címzett, másolat, stb.)*, valamint a levél témájában használhatjuk.

Tartalomtípusok

Mint tudjuk, a Content-Type fejléc tartalma jelzi, hogy az üzenet *(vagy üzenetrész)* milyen tartalmat hordoz magában. A típusazonosítók két részből állnak, amelyet / jel választ el egymástól. A jel előtti rész a főcsoport, a jel utáni az alcsoport azonosítója. Jelenleg hét főcsoport létezik, az alcsoportokat értelemszerűen nem érdemes számolni. Most felsoroljuk a leggyakrabban használt tartalomtípusokat, illetve azok leírásait, a teljes lista szerte az RFC-k között található meg *(pl. RFC 2046)*:

- text/plain: egyszerű szöveg
- text/html: HTML tartalom, az egyre elterjedtebb HTML alapú levelek azonosítója. A charset paraméter itt is használható, bár a HTML nyelv önmaga is tartalmaz a kódtáblák kezelésére való elemeket. Lásd még: RFC 1866.
- image/gif, image/jpeg, image/bmp: Képek, grafikák.
- audio/basic, audio/32kadpcm, video/mpeg: Multimédia audio és videó, ma már inkább HTML levélbe ágyazva találhatók meg *(bár csatolt fájlként még találkozhatunk ezekkel a típusazonosítókkal is)*.
- application/msword, application/vnd.ms-excel: Az application főcsoportba tartoznak a különböző alkalmazások által használt formátumok, mint a példában említett word, illetve excel. Az „alternatív" formátumok általában ebbe a főcsoportba kerülnek.
- application/octet-stream: 8 bites adat
- application/x-pgp-keys, application/x-pgp-signature, application/x-pgp-encrypted: a PGP titkosítással ill. digitális aláírással ellátott üzenet részei *(RFC 2015)*
- application/x-pkcs7-MIME, application/x-pkcs7-signature, application/x-pkcs10: az SMIME titkosítással illetve digitális aláírással ellátott üzenet összetevői *(RFC 2311)*
- message/rfc822: teljes SMTP levél, fejlécekkel, tartalommal együtt. Ez a típusazonosító leggyakrabban akkor látható, ha a levelünkhöz másik levelet csatoltunk, illetve, ha...:
- message/delivery-status: ez bizony egy kézbesítési üzenet azonosítója. Ilyen üzenetet a jólnevelt levelezőkiszolgálók küldenek, amikor valami hiba történik a levél kézbesítése, továbbítása során *(például: címzett nem található)*. Az üzenet egy, az SMTP fejlécekhez hasonló adathalmaz, ami a hiba vagy jelenség jellemzőit tartalmazza. A Reporting-MTA a hibaüzenetet küldő levelezőkiszolgáló, a Received-From-MTA pedig az a kiszolgáló, akitől a fenti a hibát okozó üzenetet kapta. Az Arrival-Date az üzenet érkezésének időpontja. A kiszolgálókra vonatkozó információk után egy üres sor, majd a címzettekről szóló rész következik: láthatjuk az eredeti, valamint a valós címzettet *(Original-Recipient, Final-Recipient)*, és például az üzenet lényegét, az eseményt/műveletet *(Action: failed)*. További információk, illetve a fentiek részletes leírása az RFC 1894-ben található.
- message/disposition-notification: Ez a típusa az olvasásról kért visszajelzésként érkező üzenet informatív részének. Formátuma az előzőhöz hasonló, fejléc-jellegű, olvasható benne a címzett neve *(Final-Recipient)*, valamint az is, hogy mi történt az üzenettel *(Disposition: displayed, dispatched, processed, deleted, denied, failed)*, valamint, hogy a visszajelzőüzenet automatikusan, vagy a címzett előzetes megkérdezésével érkezett *(manual-action, automatic-action)*, és még néhány más adat, amiről az RFC 2298-ban részletesen is bárki olvashat.

Az utolsó fő típus-csoport pedig a multipart, ami a MIME igazi erejét adja. A több részből álló MIME üzenetekről a következő számunkban szólunk.

Folytatjuk ...

Fülöp Miklós
mick@netacademia.net

A cikkben található URL-ek:

[1] RFC xxxx: http://www.ietf.org/rfc/rfcxxxx.txt
[2] Quoted-Printable kódoló/dekódoló demonstrációs script: http://download.netacademia.net/tools/b64.vbs
[3] Base64 kódoló/dekódoló demonstrációs script: http://download.netacademia.net/tools/qp.vbs

.NET? .NET!

Bevezetés

.NET. Visszhangzik a fejünkben ez a három betű, amelyet egy évvel korábban még csak top level domain névként ismerhettünk, most pedig a legváltozatosabb fórumokon, – mind a Microsoft, mind a konkurencia részéről - csak erről hallunk. Mi lakik e három betű *(4 karakter)* mögött? Valóban óriásit alkotott a Microsoft, vagy csak leporolta a Windows DNA-t? Mit takar a fogalom? Marketingfrázis, vagy valódi tartalommal bíró technológia? A cikkben ezekre a kérdésekre keresem a választ.

Viszlát COM!

Fejlesztői szemmel nézve talán nem volt még ekkora változás a Microsoft alapú fejlesztések történetében, mint aminek most vagyunk a szemtanúi és előbb-utóbb aktív alkalmazói. Annak idején a COM *(akkori nevén OLE)* megjelenése forradalmi volt, mert lehetővé tette különböző programnyelvek együttműködését olyan komponenseken keresztül, amelyet bármely COM-ot támogató nyelvben fel lehetett használni. Ez nagy előrelépés volt a modularizált rendszerek fejlesztésében, mert így minden részfeladatnál könnyedén ki lehetett választani a megfelelő nyelvet a megfelelő feladatra.

A COM legnagyobb evangelistája Don Box. ő az elmúlt 10 évben szinte kizárólag a COM oktatásának szentelte az életét. Nemrég megjelent egy cikke „Is COM dead?" címmel *(MSDN magazin, 2000. decemberi szám)*. Ha Don Box-tól ilyen írást látok COM ügyben, akkor minimum elgondolkodom. Az elmúlt évtizedben a COM volt az alapköve Microsoft technológiáknak, s most az atya megkérdőjelezi a jövőjét? S mit ír Box válaszul a címbeli kérdésére? „Az attól függ, hogyan definiáljuk a COM-ot." Jó válasz, érezzük a lényegét. A COM, mint technológia jó volt, de most tovább kell lépni, mert itt kopogtat valaki, aki sokkal többet ígér: a Microsoft.NET.

Szia .NET!

Az előző fejezetben önkényesen kiemeltem egy pontot, a COM kérdését. Ez nem azt jelenti, hogy a .NET egyfajta COM++, vagy akár stílusosan COM#, csak érzékeltetni akartam, hogy a .NET nem egyszerű továbbfejlesztése valaminek, hanem a háttér teljes újragondolása, és az alapoktól történő újraépítése. Nem szeretem a forradalmi jelzőt, mert annak van egy visszalépésre is utaló mellékíze, de ami most a Microsoftnál történt, az tényleg szenzációs. Nem forradalom, hanem továbbfejlődés, evolúció. A Microsoft átgondolta az eddigi technológiáit, megnézte, melyek azok, amelyek jó irányba indultak el, és melyek azok, amelyeket az évek folyamán addig fejlesztgetett tovább, hogy már nincs értelme tovább bonyolítani, inkább érdemes teljesen az alapoktól, a menetközben megváltozott igényeknek megfelelően a nulláról újra kidolgozni. Így járt például az ADO és az ASP technológia. Mindkettőnek csak a neve maradt meg *(természetesen .NET kiegészítéssel)*, mögöttük azonban valaki teljesen más lakik, mint az elődjük.

A váltás fő kiváltó oka az internetes világban zajló változásokban keresendő. Az eddigi egymástól elszigetelt rendszereket egymással automatikusan kommunikálni képes rendszerek fogják felváltani, függetlenül azok operációs rendszerétől vagy hardverétől. Ehhez természetesen mindenki által elfogadott szabványok kellenek, amelyek első verzióját a World Wide Web konzorcium már véglegesen kidolgozta: az adatcserére XML, illetve rá épülve távoli szolgáltatások meghívására a SOAP.

A .NET-et minden szinten áthatja az XML. Az adatbázisokból felolvasott rekordokat közvetlenül XML-ként lehet elmenteni, XML-t visszaolvasni, formázni, összerakni, transzformálni, visszaírni adatbázisba. A SOAP az alapja a Web Szervizeknek, amely a Microsoft víziója az internetes szolgáltatások jövőjéről. Érdemes vele foglalkozni, hamarosan minden fejlesztő találkozni fog vele.

S mi történt a programozási nyelvekkel? Azt nem merem mondani, hogy a Visual Basic-kel mi történt, mert annak a sorsa még nagyon képlékeny. Elindult azon az úton, ami „A" programozási nyelvek szűk klikkjébe vezet, ám félúton, a Béta1 és a júliusban várható Béta2 között a VB fejlesztők nyomására egy kicsit kezdik visszaállítani a „régi" VB szintaktikáját. Nyilvánvaló, hogy erre a már meglevő kódok migrációja miatt van szükség, azaz, hogy a VB6-ban megírt kódok fussanak VB7.NET alatt is.

De van egy rossz hírem. A régi és az új Basic csak a szintaktikájában hasonlít egymásra, másban sem. Attól, hogy a Microsoft enged a nyomásnak, és a tömbök újra 1-től lesznek indexelve, mint régen, nos, ez csak egy apróság *(persze sok más is változik)*. A lényeg, a háttér alapjaiban megváltozott, és a programok nem ilyen apróságok miatt nem fognak futni, minthogy félremegy egy index.

Döntenie kellett a Microsoftnak. Kompatibilitás a régi rendszerekkel, vagy olyan lehetőségek, amelyekről még csak nem is álmodtak a VB programozók. Mindenképpen fájdalmas a döntés, mert aki már sok munkát belefektetett egy VB6-os projektbe, az kompatibilis VB.NET-et akar, aki pedig új szoftvert kezd fejleszteni, az egy olyan nyelvet szeretne, ami „mindent" tud. Úgy tűnik az utóbbi szándék az erősebb.

Azonban nem egyszerűen arról volt szó, hogy dönteni kellett a kibővített funkcionalitás és a kompatibilitás között. A háttérben ott ül egy kényszerítő erő, az összes .NET-es nyelv alapja, amelyet úgy hívnak:

Common Type System

És lassan elérkezünk a Microsoft.NET alapjaihoz. Milyen adattípusok voltak a Basic-ben? Volt Long, Integer, String és még sokan mások, de nem volt például előjel nélküli egész szám. Mi volt C++-ban? Voltak mindenféle egész számok, lebegőpontos számok ... és nem volt String. Natívan, nyelvi szinten nem volt olyan típus, hogy String. Mit történt, ha egy Basic programnak ki kellett olvasni egy C-ben deklarált LPCSTR *(Long Pointer to Constant String)* által hivatkozott szöveget, mondjuk egy függvényhívás során? Ter-

mészetesen ki tudta olvasni, csak tudnia kellett, hogy a kapott címen egy nulla bájttal lezárt karaktertömböt találhat. S mi van, ha egy Delphi-ben deklarált sztringet akart kiolvasni? Akkor tudni kellett, hogy a Delphi hogyan tárolja a sztringeket. Ráadásul a helyzetet csak bonyolítja, hogy használunk Unicode karaktereket is.

Jól látható, hogy a nyelvek között együttműködés egyik fő akadálya a különbözőképpen deklarált típusokban keresendő *(emellett még persze a hívási konvenciók és ezer más apróság is a képbe jön)*. Ez a szó, a típus kulcsfontosságú alapfogalom a .NET-ben. A típus leírja egy objektum tárolásának módját és a típusból létrehozott változókon végrehajtható műveleteket. Ha a Visual Basic, a C++ és a Delphi is ugyanolyan módon értelmezné a sztring típust, akkor könnyedén érthetnék egymás változóit.

Mi volna, ha már az elején meghatározhatnánk a legfontosabb primitív típusokat, és a nyelvek fordítóprogramjainak *(compiler)* kötelező lenne a típus által előírt adatstruktúrákat létrehozni, az objektumokat szabványosan lefoglalni és felszabadítani? Ekkor a nyelvek végre megértenék egymást, és ... elérkeztünk a .NET alapjához, a Common Type System-hez *(CTS)*. A CTS-ben leírt típusokat közvetlenül ismeri a .NET futtató rendszere, és az összes .NET-es nyelv fordítóprogramja. Ennek köszönhetően válnak lehetővé olyan együttműködési lehetőségek, amelyekről eddig szó sem lehetett. Például egy VB.NET osztály öröklődhet egy C++-ban vagy C#-ban leírt komponensben definiált osztályból! Ezen érdemes egy kicsit elgondolkodni, mert ez már nemcsak egyszerű együttműködés a nyelvek között, hanem valódi összeépülés, integráció. Köszönjük CTS!

A Common Language Runtime és a Class Library

Van közös típusrendszerünk, így a nyelvek közötti integráció lehetővé vált. Azonban szükségünk van olyan rendszerre, ami valóban összehozza a különböző nyelveken megírt részeket, menedzseli a kódunkat, és szolgáltatásokat nyújt a szoftvereinknek. Ez a Common Language Runtime *(CLR)*. Az előző fejezetben leírt CTS is a CLR része.

A választott programnyelvünkben a fordítóprogramok bocsátják a rendelkezésünkre a CLR funkcionalitását. A .NET-es fordítóprogramok olyan kódot generálnak, ami közvetlenül együttműködik a CLR-el. Az ily módon írt kódot menedzselt kódnak hívjuk. Csak a menedzselt kód tudja kihasználni a nyelvek közötti együttműködést, valamint a CLR biztonsági, nyomkövetési és még számtalan egyéb előnyét.

A CLR mögött áll egy több ezer*(!)* osztályból álló kódkönyvtár, a Class Library. Az impozáns méretén kívül mi ebben a nagyszerű? Egy kicsit gondolkodjunk el a múlton. A C++ programozók mögött többféle kódkönyvtár is állt. MFC az asztali alkalmazásokhoz, ATL a COM komponensekhez. STL, ha szükségünk volt szabványos algoritmusokra.

Mi volt a Basic programozók mögött? A Basic Runtime, ami teljesen más megközelítésben írta le egy ugyanolyan típusú alkalmazás szerkezetét.

Mi történt, ha egy Visual Basic programozót átültettek, hogy írjon alkalmazást C++-ban? Káosz, memory leak, general protection fault és társai. Nem a nyelv szintaktikája miatt, hanem azért, mert C++-ban sokkal több mindent megtehetett, mind Basic-ben *(és meg is tett)*. Emellett a nulláról meg kellett tanulnia, hogyan kell létrehozni, például egy ablakot, mondjuk MFC-ben.

A nyelv egy eszköz, amellyel a természetes nyelvi problémát le tudjuk fordítani valamilyen programozási nyelvre. Olyan nyelvet választok, amihez értek, és amellyel a legtermészetesebben le tudom írni a problémát. Ez az elmélet. Ezzel szemben általában a nyelvek mögötti kódkönyvtár által nyújtott szolgáltatások döntöttek egy nyelv használata mellett vagy ellen. De ennek vége! A Visual Basic.NET programozó pontosan ugyanazt a háttérkönyvtárat használhatja a programjaiban, mint a C++, C#, JScript *(Perl, Python, Eiffel, Cobol, ...)* fejlesztő. A CLR és a Class Library szolgáltatásait nyelvtől függetlenül ugyanúgy el lehet érni bármely programnyelvből.

Mindig is voltak kínos kérdések Visual Basic-ben. Többszálú programozás, registrykezelés, valódi többszálú komponensek írása *(free vagy neutral threaded)*. Ezek a funkciók mind natív módon benne vannak a CLR-ben, így például egy többszálú programot ugyanúgy meg lehet írni Basic-ben, mint C++-ban, csak a formátum más, a mögötte legenerálódó kód ugyanaz, ha C++-ben is csak menedzselt kódot írunk! A C++-nak ezután is meglesz az a kiváltsága, hogy szabadon azt tegyünk, amit akarunk, azaz kiléphetünk a menedzselt kód biztonságos keretei közül egy kis ámokfutásra. Más kérdés, hogy érdemes-e?

Azután voltak kínos kérdések még C++-ban is *(meg persze olyanok, amelyek csak C++-ban voltak kemények, mint a COM programozás)*. Teljesítménymutatók figyelése és létrehozása. Az NT biztonsági rendszere, ACL, ACE, SID és társai programozása. Titkosítás, digitális aláírás készítése. *(Hogy megemlítsek hármat a több tucatból.)* A Windows 2000 és elődei tele vannak-voltak olyan lehetőségekkel, amelyeket vagy nem ismertünk, vagy csak a szabványos API-kon keresztül voltak elérhetőek, de akkor meg már megbántuk, hogy megismertük, mert olyan bonyolultak voltak. Ennek is vége!

A CLR mögötti osztályhierarchia, a Class Library felfogható egyfajta objektumorientált API-nak is. Ebben az a szenzációs, hogy az összes nyelvből könnyedén el lehet érni az összes szolgáltatást, így nincsenek többé kínos kérdések. A Class Library magába olvasztja szinte az összes, eddig csak API-n keresztül elérhető szolgáltatást, meg még több ezer egyéb feladatot is. Vannak benne klasszikus algoritmusok *(rendezések, keresések, halmok kezelése, satöbbi)*, valamint rengeteg gyakori feladat kész megoldása *(például web-es autentikáció, állapotkezelés)*. Azaz mielőtt nekikezdenénk valamilyen funkció implementálásába, érdemes szétnézni, nincs-e készen már a Class Library-ben.

Common Language Specification

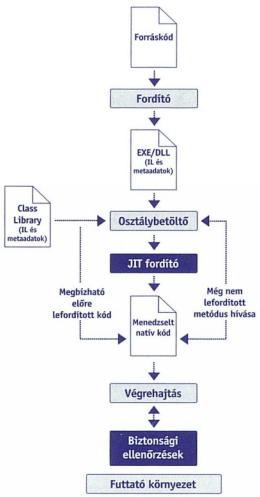
Beszéltünk arról, hogy a nyelvek integrációjához szükség van egy közös alaptípus rendszerre, ez a CTS. Kellett egy háttér és kódkönyvtár, ez volt a CLR és a Class Library. Azonban mi történik, ha az egyik nyelven megírt kódomban valami olyan konstrukciót használok, amit nem ismer egy másik nyelv, ami szeretne együttműködni velem? Baj, nagy baj, megbukna az együttműködés. Hogy ez ne következhessen be, minden .NET nyelvnek támogatnia kell egy minimális funkcióhalmazt, de ezen felül még tartalmazhatnak plusz szolgáltatásokat is. Ezt a minimális funkcionalitást írja le a Common Language Specification.

Ebben a „minimális" halmazban benne vannak olyan fogalmak, mint metódusok felüldefiniálása, öröklődés, vagy az

események natív támogatása. Nem csoda, hogy a Visual Basic-et ki kellett gyúrni, hisz mit értünk volna egy olyan közös nevezővel, mint ami a Visual Basic 6 volt?
A CTS-ben sokkal több lehetőség van, mint amit a CLS megenged. Így abban az esetben, ha a nyelvek közötti együttműködés másodlagos szempont, ki lehet használni, ha valamelyik nyelv több mindent implementál a CTS-ből, mint a másik. És itt jön be az elv, miszerint egyenlők között mindig vannak még egyenlőbbek is. Ha a CTS szinte teljes tárházát ki akarjuk használni, akkor használjuk a C# *(ejtsd: szí sárp)* nyelvet. A legteljesebb módon ezen a nyelven keresztül lehet hozzáférni a CLR szolgáltatásaihoz.

A közös nyelv

Valami trükk kell, hogy legyen a háttérben, hogy annyira különböző nyelvek olyan jól szót értenek egymással. A COM technológiában a közös nyelv a gépi kód volt. Bármilyen programozási nyelven is írtunk egy COM komponenst, a végén x86-os gépi kódra fordult le. Ez szükséges feltétel volt a nyelvek közötti együttműködéshez. Ez a bináris kódmegosztás azonban egy platformra szögezte a komponens használhatóságát. A .NET stratégia alapköve az információk elérése bármilyen eszközről. Első körben mindenkinek a PC jut eszébe, azonban az nem megy velünk mindenhová. Vannak más eszközök is, hogy a legkézenfekvőbbet említsem: a mobiltelefonok, amelyek egyre okosabbak, és egyre több információt tudnak elénk varázsolni olyan helyeken is, ahol a PC-knek nincs létjogosultsága. Milyen processzor van az ilyen kézi készülékekben, mint a telefonban, vagy a hand-held PC-kben? Milyen nyelven programozzák őket? Ki tudja...
És mi van, ha azt mondom, hogy az Önök által írt, például Visual Basic kód futni fog egy mobiltelefonon? Gondolom, körberöhögnek. Persze, ha hoznak egy száz form-ból álló programot, akkor én is elmosolyodom. Vannak dolgok, amelyek természetüknél fogva csak egyféle típusú gépen fognak futni.
És mi van, ha azt mondom, hogy az Önök által írt 500 form-ból álló VB program futni fog Linux-on, anélkül, hogy Önöknek ehhez bármi erőfeszítést kellene tenni? Nos, akkor lehet, hogy már érdekesnek találják, amiről beszélek.
A Microsoft az információ mindenhol, mindenkor mottót komolyan gondolja. A .NET környezet által előállított futtatható kód ugyanis nincs hozzászögezve egy platformhoz sem, futhat bármi olyan eszközön, ami képes ellátni azokat a funkciókat, amelyeket a program elvár tőle, minden módosítás nélkül. Hogyan tudja ezt megtenni?
A titok nyitja az Intermediate Language-ben van. Amikor egy fordítóprogram lefordítja mondjuk a C# kódunkat, akkor – ellentétben az eddigi compiler-ekkel – nem gépi kódra, hanem egy közbenső nyelvre, az Intermediate Language-re *(IL)* fordít. Magyarán, ha például Windows 2000 alatt lefordítunk egy hello.cs programot, akkor a hello.exe nem egy x86-os bináris kód lesz, hanem egy közbenső kód, IL.
Mi történik, ha elindítunk egy ilyen exe-t? A .NET framework, azon belül a CLR Virtual Execution System betölti a közbenső kódot, és a szükséges függvényeket, objektumokat lefordítja az adott platform gépi nyelvére. Ezt az igény szerinti, futásidő alatti fordítást Just in Time compilation-nek hívják. A jitter *(így becézik a JiT compiler-t)* nem fordítja le egyszerre az egész programot, csak a hivatkozott részeket, így gyorsabban indulnak el az alkalmazások. Látszólag ez nagy teljesítményveszteség a bárhol futtatható kódért cserébe. Azonban tudnunk kell, hogy az IL egy elég alacsonyszintű kód, amelyet gyorsan át lehet fordítani gépi nyelvre. A futtatás módját *(valamivel részletesebben, mint ahogyan leírtam)* az alábbi ábrán lehet közelebbről szemügyre venni.



Kétféle fordítót is lehet írni. Az egyik - figyelembe véve az aktuális futásidejű körülményeket - optimalizál, például észreveszi a processzorcsaládot, az aktuális memóriafelhasználást, satöbbi. Így akár az egyik gépen PIII-as processzorra, más gépen csak Pentium utasításokat használva generál kódot. Egy ilyen fordító kifejlesztése elég sok időt igényel, ezért nyilván csak olyan eszközökre éri meg megírni, amelyek várhatóan elég tartósak lesznek, például a PC-k.
Ezzel szemben egy hűtőgép böngészőjét futtató processzorra nem biztos, hogy megéri fél évig fordítót fejleszteni, mert addigra már úgyis másik processzor lesz benne, eladják a gyárat, vagy leállnak a típus gyártásával. Az ilyen illékony platformokra inkább írnak egy egyszerű fordítót, ami szinte makrószerűen legenerálja az IL utasítások aktuális gépi párját. Egy ilyen fordító egy hét alatt megvan, így semmi akadálya annak, hogy a mosógépünkön is fusson a szuper képernyővédőnk, csak legyen rá .NET framework. :)
Az IL kódot elég jól követhetően lehet olvasni, ebben segít az IL Disassembler nevű program, ami a .NET Framework SDK része. A szemléletesség kedvéért álljon itt egy egyszerű VB.NET program, és annak a lefordított IL változata, ahogy az IL Disassembler mutatja:

```
Sub Main()
    Dim i As Integer
    Dim a As Integer = 20
    For i = 1 To 10
        a = a - 2
    Next
    Console.WriteLine(a)
```

```
    End Sub
.method public static void Main() il managed
{
  // Code size        28 (0x1c)
  .maxstack  2
  .locals init ([0] int32 a,
           [1] int32 i)
  IL_0000:  nop
  IL_0001:  ldc.i4.s   20
  IL_0003:  stloc.0
  IL_0004:  ldc.i4.1
  IL_0005:  stloc.1
  IL_0006:  ldloc.0
  IL_0007:  ldc.i4.2
  IL_0008:  sub.ovf
  IL_0009:  stloc.0
  IL_000a:  nop
  IL_000b:  ldloc.1
  IL_000c:  ldc.i4.1
  IL_000d:  add.ovf
  IL_000e:  stloc.1
  IL_000f:  ldloc.1
  IL_0010:  ldc.i4.s   10
  IL_0012:  ble.s      IL_0006
  IL_0014:  ldloc.0
  IL_0015:  call       void
[mscorlib]System.Console::WriteLine(int32)
  IL_001a:  nop
  IL_001b:  ret
} // end of method Module1::Main
```

Jól felismerhetőek az értékadások, illetve a 12-es pozíción a for ciklus végén a feltételes visszaugrás a ciklus elejére. Ugyanezt a programot megírtam C#-ban is. A lefordított IL kód szinte ugyanaz volt. Nem véletlenül írtam, hogy bár a VB kívülről majdnem ugyanaz maradt, belül felnőtt olyanná, mint a többi programnyelv.

Csak egyet nem értek. Mit keres az a nop, a ciklus közepén *(is)*? A C# IL kódjában nincs benne egy darab sem. Esélykiegyenlítés? :)

Nem. A VB fordító csak akkor helyezi el a NOP-okat a kódba, ha debug verziót fordítunk. Ez a nyomkövetés közbeni kód módosítást teszi lehetővé.

Metadata, az összekötő kapocs

Ha .NET, akkor metadata, metadata és metadata. Mit takar ez a fogalom, és miért olyan fontos számunkra?
Egy kicsit nézzünk megint a múltba. Kapunk egy COM komponenst, például egy .DLL-ben. Az a feladatunk, hogy dokumentáció nélkül mondjuk meg róla, hogy milyen osztályok vannak benne, azok milyen interfészeket implementálnak, milyen metódusaik, tulajdonságaik vannak, a metódusok visszatérési típusát és paramétereit. Vagy milyen threading modellű a komponens? És mondjuk ezt futásidőben, VB-ből. Menni fog? Nagyon nehezen, de igen. De van egy feltétele: vagy a DLL-hez kell kapjunk egy Type Library-t, vagy a DLL-be bele kell, hogy legyen fordítva a TL. Emellett nem árt, ha a registry-t is tudjuk olvasni, mert a threading modell ott van leírva. Sőt, ha a kapott komponens be van regisztrálva MTS vagy COM+ alá, akkor még azokat a beállításokat is le kell kérdezzük, a system catalog-ból.
Mondhatná valaki, hogy a Visual Studio Object Browser megmutatja az összes információt, amire kíváncsi vagyok. Hogyne, de honnan is tudja? Az is pontosan arról a három helyről szedi össze az információit, amit az előbb leírtam. Mindhárom információforrás a COM komponens valamilyen jellemzőjét írta le, hogy ezáltal használható legyen a hívó számára. Ezek az információk nem csak kényelmi szolgáltatást nyújtanak a Visual Studio felhasználóinak. A Type Library írja le a belépési pontokat *(vptr)* az osztályok metódusaihoz is. A TL-ban található információk nélkül lehetetlen olyan programot írni, ami hatékonyan meg tud hívni egy komponens egy metódusát *(az IDispath-et végre felejtsük el)*. Azaz eddig is volt leíróinformáció az újrahasznosítható komponensekről, csak több helyről kellett összeszednünk.
Az ilyen jellegű leíró információkat metaadatnak hívjuk. A .NET-ben a metaadatok sokkal részletesebb leírást adnak a komponensekről, mint a COM-ban. Ez szükséges feltétel volt a nyelvek integrációjához. Egy komponens szerkezetét a-tól z-ig fel lehet térképezni a System.Reflection osztályokon keresztül. Milyen előnyök származnak ebből?
Mivel a komponensek a teljes leírásukat magukkal viszik, nem kell regisztrálni őket! Ez nagyon nagy előrelépés, hisz így egy alkalmazás telepítése fájlok másolását jelenti.

Semmi regsvr32.exe!

A hivatkozások feloldása futásidőben történik, így a metódusok hívása előtt biztonsági ellenőrzéseket tud beiktatni a CLR *(lásd korábbi ábra)*. Serialization, Remoting. A .NET gerincei, mind a metaadatoknak köszönhetően működnek. És ami a következő pár évben még nagyon fontos lesz: a meglévő COM komponenseinket nem kell kidobnunk, mert a háttérben a metadata szolgáltatások segítségével könnyedén meghívhatjuk a COM komponenseket a .NET-es programjainkból. Van egy TlbImp.exe nevű kis programocska, amelynek segítségével a COM komponensek Type Library-jét átkonvertálhatjuk .NET metadatává. Ettől kezdve a COM komponensünk funkcionalitását éppúgy elérhetjük, mintha az egy managed .NET assembly *(durván komponens)* lenne.
Azaz a Microsoft nem vágta el az utat a COM felé, sőt kifejezetten könnyű az együttműködés mindkét irányba. Ez a már meglevő hatalmas COM komponenskészlet miatt alapvető szempont volt a .NET tervezésekor.

Zárszó

Fejlesztői szemmel átnéztük a .NET stratégia alapköveit. A .NET-ről nem 4, de 4000 oldalt is meg lehet tölteni, mégsem jutnánk a végére. Ami viszont világosan látszik, hogy döbbenetesen új szelek fújnak a fejlesztésben, amiről kár lenne lemaradni. És bár még csak Beta1-es korszakában van a .NET, már érdemes vele foglalkozni, mert ősszel már itt is van a végleges fejlesztőeszköz és kódkönyvtár. Izgalmas, új korszak kezdődik!

Soczó Zsolt
MCSE, MCSD, MCDBA
Zsolt.Soczo@netacademia.net

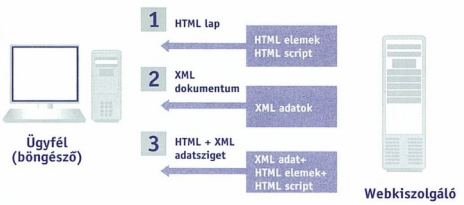
XMLgessünk *(III. rész)*

Bevezetés

Az előző számban átnéztük az XML technológia előnyeit a webfejlesztésben. Megismertük, hogy mekkora lehetőség van az XSL-ben az XML adataink formázásában. A mostani részben megnézzük, hogy az előző részekben lefektetett elméletet hogyan tudjuk átültetni a gyakorlatba, XML adatszigetek, XSLT és DOM felhasználásával.

XML adatszigetek

Milyen tartalmat küldjön a webkiszolgáló a böngészőnek? A HTML formátum nagyon jó a felhasználói felület leírására, az adatok megjelenítésére, sőt, scripteket is bele lehet ágyazni, de elfedi az adataink jelentését, így ügyféloldalon nagyon nehézzé válik azok további kezelése, például, ha helyben rendezni akarunk. Az XML remek az adataink szállítására, de egy weboldalban kellenek scriptek és felhasználói felület is.



Adatforgalom a webkiszolgáló és az ügyfélprogram között

Valahogyan ötvözni kellene a két technológia előnyeit. Ha valamilyen egyszerű, de könnyen kezelhető módon egyszerre tudnánk XML és HTML adatokat is küldeni a böngészőnek, akkor könnyen, gazdag felhasználói felülettel kombinálva tudnánk az adatokat ügyféloldalon manipulálni. Megérkeztünk az XML szigetekhez!

Internet Explorer 5-től kezdődően megjelent egy <xml> nevű HTML elem, amely segítségével XML adatokat ágyazhatunk be HTML lapba. A formátuma nagyon egyszerű, a kívánt XML dokumentumot az <xml> </xml> tagok közé kell rakni, egy azonosítót adva az szigetnek, hogy script-ből el tudjuk érni:

```
<html>
<head>
<title>XML tanfolyamok</title>
</head>
<body>
<xml id="dsoCourses">
<?xml version="1.0" encoding="ISO8859-2"?>
<courses>
<course>
<code>1905</code>
<title>Building XML-Based Web Applications</title>
<trainer name="Soczó Zsolt">
    <email>
        zsolt.soczo@netacademia.net
    </email>
</trainer>
</course>
<course>
<code>1913</code>
<title>Exchanging and transforming data using XML
and XSLT</title>
<trainer name="Soczó Zsolt">
    <email>
        zsolt.soczo@netacademia.NET
    </email>
</trainer>
</course>
</courses>
</xml>
</body>
</html>
```

Mit látunk az adatokból, ha ezt a HTML lapot *(tanf1.html)* **[1]** megnézzük a böngészőben? Semmit. A böngésző magától nem jeleníti meg az xml sziget tartalmát, az a mi dolgunk. Azonban hathatós segítséget kaphatunk tőle. Kétféle módon jelenítheti meg a sziget adatait: vagy automatikusan az adatkötés *(data binding)* felhasználásával, vagy kézzel, scriptből. Nézzük meg őket közelebbről!

Adatkötés (Data Binding)

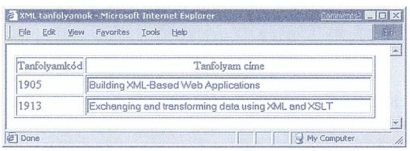
Az Internet Explorer automatikusan képes megjeleníteni az xml sziget adatait különböző HTML elemekben, nekünk csak jelezni kell, hogy mi az adatok forrása. Ez egyrészt a szigetünk megjelölését jelenti, a szigetnek adott id alapján, illetve ezen belül meg kell adnunk annak az xml elemnek a nevét, amelyet meg akarunk jeleníteni. Az alább látható HTML kódot kell csak becsempészni a fenti dokumentum törzsébe, és máris megjelenítettük a tanfolyamok kódját és címét:

```
Tanfolyamkód: <span datasrc="#dsoCourses"
datafld="code" ></span><br/>
Tanfolyam címe: <span datasrc="#dsoCourses"
datafld="title" ></span>
```

A böngészőben ez így néz ki:

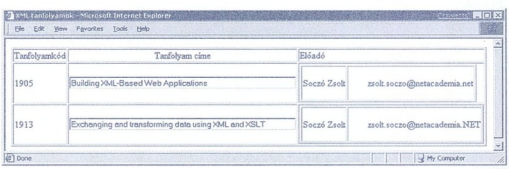
Nem minden HTML elem köthető adatforráshoz, csak néhány. A tipikusak a div, span, label, frame, img, ha nem szerkeszthető módon kellenek az adatok, illetve szinte az összes input mező *(textbox, checkbox, satöbbi)*.
Az előbbi példánk azért több sebből is vérzett. Egyrészt két tanfolyamot is meg kellett volna jeleníteni, amihez valamilyen módon annyiszor meg kellene ismételni a span-eket, ahány course elem van a szigetben. Mit tehetünk? Vagy „kézzel", scriptből ismételjük meg a span-eket a szükséges darabszámmal, vagy megnézzük, van-e lehetőségünk ismétlődő adatok megjelenítésére.
Természetesen van. A jó öreg HTML table elemet kiegészítették az adatkötés képességével. S nem is akármilyennel! Az ő feladata az adatforrásban ismétlődő információk megjelenítése táblázatos formában. Ekkor az adatforrást csak a table elemnek kell megadni, a benne levő, megjelenítést végző elemeknél már csak a megfelelő mezőt kell jelezni. A változatosság kedvéért most a tanfolyam nevét egy input text mezőhöz csatoltam:

```
<table datasrc="#dsoCourses" border="1">
<thead>
<tr>
<td align="center">Tanfolyamkód</td>
<td align="center">Tanfolyam címe</td>
</tr>
</thead>
<tbody>
<tr>
<td><span datafld="code" ></span></td>
<td><input datafld="title" size="60"></input></td>
</tr>
</tbody>
</table>
```



Figyeljük meg, hogy a táblázat csak a tbody-ban található részt ismétli meg, így tudunk fejlécet adni neki.
Mi a helyzet azokkal az adatokkal, amelyek a hierarchia mélyebb szintjén laknak? Ott van mindjárt a <trainer> szekció. Közvetlenül nem írhatom be egy harmadik elembe azt, hogy datafld="trainer/email", mert az adatforrást nem lehet XPath kifejezésekkel lekérdezni *(legalábbis ebben a mezőben nem)*. Azért nem, mert az adatsziget mögött álló adatforrás-objektum *(Data Source Object, DSO)* nemcsak XML, hanem egyéb adatokat is tud kezelni *(szövegfájl, Remote Data Services, ...)*, és azoknak pedig gőzük sincs az XPath-ról.
Ennek ellenére a DSO lehetőséget nyújt hierarchikus adatok megjelenítésére is. Ehhez egymásba ágyazott táblázatokat kell használnunk. A külső megjeleníti a gyökér alatti szint adatait, mint az előbb is. Ezen belül definiálhatunk további táblázatokat, amelyeknek megadjuk ugyanazt az adatforrást mint a szülőtáblázatnak, de megadunk egy mezőt is, azt a mezőt *(illetve XML-ben azt az elemet)*, ahol le akarunk fúrni egy szinttel lejjebb. Ezután a táblázatban a már látott módon megjelenítjük a megfelelő gyermekelemet. Jelenítsük meg a trainer elem gyerekeit, azaz a name és az email értékét!

```
<table datasrc="#dsoCourses" border="1">
<thead>
<tr>
<td align="center">Tanfolyamkód</td>
<td align="center">Tanfolyam címe</td>
<td>Előadó</td>
</tr>
</thead>
<tbody>
<tr>
<td><span datafld="code" ></span></td>
<td><input datafld="title" size="60"></input></td>
<td>
<!-- Itt van a beágyazott tábla -->
<table datasrc="#dsoCourses" datafld="trainer"
border="1">
<tr><td><div datafld="name"></div></td>
<td><div datafld="email"></div></td></tr>
</table>
</td></tr>
</tbody>
</table>
```



Az XML DSO az XML dokumentumból az elemek attribútumait és az elemek értékét síkba kiterítve szolgáltatja, így a mind a name attribútumot, mind az email elem értéket közvetlenül elérhetjük a nevükre hivatkozva. Ha ütközés volna egy attribútum neve és elem neve között, akkor felkiáltójelet kell írni az elem neve elé.

Interaktivitás, ember!

Tegyük fel, hogy sok adat van az XML szigetünkben, amelyeket szeretnénk megszűrni. A hagyományos modellben ilyenkor a szűrési feltételt elküldenénk a webkiszolgálónak, az elvégezné a szűrést, és az eredményt visszaküldenénk az XML szigetben. Természetesen mi olyan megoldást szeretnénk, hogy a szerver háborgatása nélkül, a böngészőben tudjunk egyszerűbb szűréseket vagy sorbarendezést megejteni. Ekkor kell a fejlettebb lehetőségekhez, az XPath és az XSLT vívmányaihoz fordulnunk, vagy használhatjuk az XML DOM-ot is. Mindkét megoldást megmutatom, mert tanulságos.
Tegyük fel, hogy a feladat az, hogy lehessen szűrni a tanfolyamok nevére. Ehhez fel kell rakjunk egy szövegbeviteli mezőt a szűrési feltételhez, és egy gombot, a keresés megkezdéséhez. A gomb megnyomására ki kell éleznünk egy függvény végrehajtását az OnClick eseményre. Itt lesz a szűrést végző kódunk:

```
<script language="JScript">
<!--
function SearchCourse(strCourseFilter)
{
    //Itt lesz a kereső kód.
}
-->
</script>
<input id="txtCourseFilter" />
<input type="button" value="Szimat!"
onclick="SearchCourse(document.all
['txtCourseFilter'].value);" />
```

Keresés DOM-mal

Lássuk hát a feladat megoldását, az XML DOM és XPath felhasználásával!

Az ötlet az, hogy készítünk egy másolatot az eredeti XML szigetről, amely az összes adatot tartalmazza. A másolatot egy másik XML szigetbe rakjuk, az oldal letöltődése után. Amikor a gomb megnyomására keresést kérnek, akkor az eredeti XML szigetben található dokumentumból kitörlünk minden elemet a courses elem alatt. Ezután végigmegyünk a másolat szigetben található course elemeken, és kiválasztjuk közülük azokat, amelyekre teljesül a feltétel. Ezeket beillesztjük a fő XML szigetünk mögött álló dokumentumba, és már kész is a szűrés. Ez nagyon favágó módszer, de legalább megnézzük, mire képes a DOM.

Az XML Document Object Model az a programozási absztrakció, amely segítségével az XML dokumentumot objektumként, programozott módon kezelhetjük. A DOM felolvassa az XML forrást, és az elemekből, attribútumokból felépít egy fát, amelyet programból bejárhatunk, módosíthatunk, és a végeredményt visszamenthetjük az XML dokumentumba. A DOM nem Microsoft találmány, a Word Wide Web konzorcium alkotta meg és kezeli, a Microsoft XML Parser *(MSXML)* ennek egy implementációja. Természetesen minden gyártó, így a Microsoft is kiegészíti egy-két saját funkcióval az általa megvalósított DOM-ot, vagy teljesítmény vagy kényelmi okokból. A kiegészítések is korrektül le vannak dokumentálva az MSDN-ben.

Az Internet Explorer 5-tel az MSXML 2.0 jött *(1998-ban)*, amely még az akkori w3c javaslatokra épült. Akkor még nem volt XSLT csak XSL, nem volt XPath csak XML Patterns. Azóta eltelt három év, és most az MSXML parser 3.0 az aktuális, éles környezetben is használható verzió. Ezt feltelepítve az Internet Explorer 5 is ennek a funkcionalitását fogja bírni, így lesz XSLT és XPath támogatásunk. A példákban ezekre fogok építeni.

Csapjunk bele! Az XML sziget mögötti XML dokumentumot elérhetjük a sziget XMLDocument tulajdonságán keresztül:

```
var objCourses = dsoCourses.XMLDocument;
```

Ekkor visszakapunk egy referenciát az egész XML dokumentumot reprezentáló DOMDocument objektumra. Ez magába foglal mindent a forrásdokumentumból, az XML bevezetőtől az utolsó megjegyzésig. A lap betöltődése után készítsünk ebből egy másolatot:

```
...
<body onload="CopyXMLToOriginal();">
<xml id="dsoOriginal"></xml>
...
```

```
//Készítünk egy másolatot az eredeti
//dokumentumból, így a szűréseket
//ebből kiindulva tudjuk elvégezni.
function CopyXMLToOriginal()
{
    dsoOriginal.XMLDocument.async = true;
    dsoOriginal.XMLDocument.loadXML(
      dsoCourses.XMLDocument.xml);
}
```

A DOMDocument objektum xml tulajdonsága visszaadja a benne található XML dokumentum forrását, a loadXML metódus pedig betölti és értelmezi a dokumentumot.

Nekünk most csak a szorosan vett adatok kellenek, amelyek a dokumentum szerkezetének megfelelően egy fára vannak felfűzve. Ennek a fának a gyökeréhez a DOMDocument objektum documentElement tulajdonsága vezet el:

```
var objCoursesRoot = objCourses.documentElement;
```

Így a courses elemnél járunk, amely alatt az összes csomópontot ki akarjuk irtani, mert oda fognak menni a megszűrt csomópontok. Az egyszerűség kedvéért nem végigmegyünk a courses összes gyermek csomópontján, és egyesével kitöröljük őket, hanem kicseréljük a feltöltött elemet egy üresre:

```
//Itt az eredeti doksi...
var objOriginalCourses = //IXMLDOMDocument2
  dsoOriginal.XMLDocument;
//És annak a gyökere.
var objOriginalCoursesRoot = //IXMLDOMNode
  objOriginalCourses.documentElement;
//Generálunk egy üres gyökérelemet,
//azaz átmásoljuk az eredeti gyökérelemet
//a gyerek elemek nélkül (parameter: false).
var objNewRoot =
  objOriginalCoursesRoot.cloneNode(false);
```

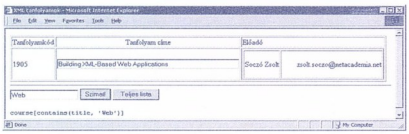
Most el kell végeznünk a szűrést egy jólirányzott XPath kifejezéssel: majd végig kell mennünk a szűrt listán, és felépíteni egy új XML dokumentumot:

```
objOriginalCourses.setProperty(
  "SelectionLanguage", "XPath");
var strPathFilter =
  "course[contains(title, '" +
   strCourseFilter + "')]";
var objFilteredList = //IXMLDomNodeList
  objOriginalCoursesRoot.selectNodes(
  strPathFilter);
var objTempNode; //IXMLDOMNode
objTempNode =
  objFilteredList.nextNode();
while (objTempNode != null)
{
objNewRoot.appendChild(
  objTempNode.cloneNode(true));
objTempNode =
  objFilteredList.nextNode();
}
```

Utolsó lépésként kicseréljük az eredeti gyökeret az újonnan felépített, szűrt gyökérre:

```
objCourses.replaceChild(
  objNewRoot,
  objCoursesRoot);
```

Ennyi. Elsőre talán bonyolult, de az érdeklődőknek mindenképpen ajánlom, hogy menjenek el az **[1]** címre, mert ott a teljes forrás egyben, bőven kommentezve megtalálható. A végeredmény pedig *(némi plusszal, ami a web-es verzióban megtalálható)*:



Keresés XSLT-vel

Ebben az esetben a vezérfonalunk az lesz, hogy készítünk egy sablon XSLT transzformációt, ami szűrni képes a forrás XML dokumentumot. A konkrét kereső kifejezést beleszerkesztjük az XSLT-be, és végrehajtjuk a transzformációt, ami a másolat dokumentumból legenerálja a szűrt dokumentumot.
Az XSLT transzformációt – lévén, hogy ő is egy XML dokumentum – egy XML szigetben helyezzük el:

```
<xml id="dsoXSLTransformation">
<?xml version="1.0" encoding="ISO8859-2"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <courses>
    <xsl:apply-templates select="courses" />
    </courses>
  </xsl:template>
  <xsl:template match="courses">
    <xsl:copy-of select=
    "course[contains(title, keresokifejezes)]">
    </xsl:copy-of>
  </xsl:template>
</xsl:stylesheet>
</xml>
```

Ez a rövidke XSLT transzformáció a courses alól kimásolja az alatta levő összes course elemet, amelyre teljesül, hogy a title elem értéke benne foglaltatik a keresőkifejezésben.
A transzformáció végrehajtása előtt már csak egy dolog hiányzik: a keresőkifejezés helyébe be kellene csempészni a paraméterként kapott keresendő szöveget. Mi sem egyszerűbb, ha van DOM-unk!
Az XSLT dokumentumból kiválasztjuk az xsl:copy-of elemet, és annak a megfelelő módon átírjuk a select attribútumát.

```
function SearchCourse(strCourseFilter)
{
    var strFilter;
    //Összerakjuk a kereső kifejezést.
    strFilter = "course[contains(title, '"
                + strCourseFilter + "')]";
    //Ebben lesz az XSLT.
    var objTransformXML =
        dsoXSLTransformation.XMLDocument;
    //Kiválasztjuk a módosítandó elemet.
    var objCopyNode =
        objTransformXML.selectSingleNode(
        "//xsl:copy-of");
    //Módosítjuk az attribútumot.
    objCopyNode.setAttribute("select", strFilter)
...
```

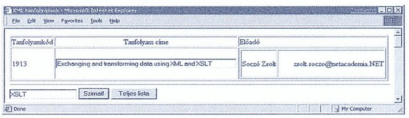
Már csak transzformálnunk kell, és az eredményt visszatölteni az eredeti XML szigetbe:

```
...
    var strXMLTransformed;
    //Itt képződik a szűrt eredmény.
    strXMLTransformed =
    dsoOriginal.XMLDocument.transformNode(
    dsoXSLTransformation.XMLDocumen);
    //Visszatöltjük, hogy megjelenjen az
    //eredeti adatforrásban.
    dsoCourses.XMLDocument.loadXML(
    strXMLTransformed);
}
```

Az utolsó két utasítást helyettesíthetjük eggyel is, a transformNodeToObject metódus felhasználásával:

```
//Alternatív transzformáció.
dsoOriginal.XMLDocument.transformNodeToObject(
  dsoXSLTransformation.XMLDocument,
  dsoCourses.XMLDocument);
```

Lássuk az oldalt működés közben **[1]**:



Azért ez egyszerűbb és átláthatóbb lett, mint a tisztán DOM-os megoldás, mert nem keveredik a DOM-ot kezelő kód és keresés logikája. A kulcsszó: XSLT **[2]**. Érdemes tanulmányozni!

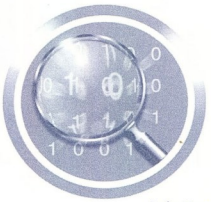
Zárszó

Megtanultunk ügyféloldalon XML adatokat megjeleníteni és transzformálni. De mi lesz az XML-t nem értő böngészőkkel? Nekik már nem is jár XSLT? Dehogynem, csakhogy nekik a kiszolgálón kell elvégezni a transzformációt, így már csak HTML tartalmat kapnak. Nem lesz interaktív, viszont ki tudjuk használni az XSLT-ben rejlő lehetőségeket. Júliusban.

Soczó Zsolt
MCSE, MCSD, MCDBA

A cikkben szereplő URL-ek:
[1]: http://technet.netacademia.net/feladatok/xml
[2]: XSLT rerefencia http://www.w3.org/TR/xslt.html

Cash-e az e-cash?



Lehettem vagy 8 éves, amikor a TV-ben először láttam Marco Polo történetének egy feldolgozását. Ki tudja miért, megragadott az a rész, amikor az Ázsiából hazatérő főhős lelkesen mutogatta a velencei kalmároknak a magával hozott papírpénzt, akik azt erős szkepticizmussal fogadták, és értéktelenségét bizonyítandó, elégették. Történt mindez a XIII. század végén. Hasonló gondolatokat ébresztett bennem nemrég egy ausztrál ismerősöm beszámolója, miszerint valahol nem akarták tőle elfogadni a műanyagból készült ausztrál dollárt. Történt mindez a XXI. század elején. Miért? A pénznek rengeteg fajtája létezett és létezik, a kagylópénztől, mint legősibb árupénztől, a nemesfémpénzen keresztül, napjaink modern pénzhelyettesítő eszközeiig. De miért is fogadjuk el a 20.000 Ft-os papírpénzt? Mert tudjuk, hogy vehetünk rajta sok-sok csokit. A pénz nem több mint társadalmi megállapodás arról, hogy azt mindenki elfogadja fizetőeszközként. Más kritériumoknak is eleget kell azonban tenni. Az angol-magyar üzleti szótár szerint „pénz minden olyan árucikk, amelyet a jog és a hagyomány általánosan elfogad fizetőeszközként, értékmérőként és kincsképzőként."
A pénz egyre kevésbé kézzelfogható, sokan alig tartanak maguknál, és pénzügyeiket, beleértve ebbe az édességboltban való vásárlást is, bankkártyával intézik el. Ez mind szép és jó, de a vásárló szemszögéből felmerülhetnek gondok, hiszen a kártyahasználat nem névtelen, azaz teljes mértékben nyomon követhető. Az internetes kereskedelem jóvoltából a weben is rendelhetek csokit, megadom a kártyaszámom, az árát levonják, majd kiszállítják. Azonban a kisösszegű bankkártyás fizetések nem kifizetődőek sem az eladónak, sem a banknak, s névtelenséget sem biztosít számomra. Hogy nézzen így körül nyugodtan az ember egy webes erotikaszalonban? Ide bizony készpénz kell, *(még ha elektronikus is)*, annak ugyanis nincs szaga, amely nyomravezethetne bárkit. A bankkártya büdös.

Elektronikus készpénz?

A Microsoft egyik kutatója, Yacobi, ezen a kiaknázatlan területen évi 30 milliárd dollárt szimatol. Ez csak töredéke a kibertér teljes forgalmának, mégis szép szelet a tortából, ha valakinek sikerül kihasítania.
Az e-cash több műszaki megoldást is kínál. A kiszolgálóoldali tárca *(wallet)* egy távoli kiszolgálón tárolja az ügyfelek pénzügyi adatait. Földrajzi helyzetét tekintve ez praktikusan az a hely, ahol az ügyfél a leggyakrabban intézi üzleti ügyeit. De lehet a tárca ügyféloldali is, amelyet a benne lévő bizalmas adatokkal egyetemben ott tarthatunk, ahol akarunk, például az otthoni számítógépen, noteszgépen, palmtopon, intelligens kártyán stb. A lényeg az, hogy senkire sem kell bízni a bizalmas adatok kezelését.
Az ügyféloldali tárca lehet érme vagy egyenleg típusú:

- Az egyenleg típusú megoldás olyan bankjegyhez hasonlítható, amelyen az összeget le lehet radírozni, és újat írni rá. Vásárlásnál sosem adom át, hanem a boltossal szép egyetértésben kiradírozzuk a rajta szereplő összeget, és felírjuk a vásárlás utáni maradékot.
- Az „érméket" egy erre jogosult pénzügyi intézmény állítja elő, pontosabban sorozatszámmal és digitális aláírással látják el az értékjegyeket, ami garantálja érvényességüket és egyediségüket. Ily módon különböző címletű „bankókat", váltópénzt állíthat elő a kibocsátó szerv. Az e-érméket a vásárláskor „kivesszük" a pénztárcából, és „betesszük" a boltos kasszájába.

Eddig egyszerű, de mi van a gyarló emberekkel? Lopás, hamisítás és csalás mindig lesz amíg világ a világ, tehát valahogy védekezni kell. Az informatika világában mára megtanultuk tényként elfogadni, hogy százszázalékosan biztonságos rendszer nem létezik, legfeljebb törekedhetünk arra, hogy minél jobban megnehezítsük az ellenfél dolgát. Az összes tranzakció ellenőrzése kivitelezhetetlenül drága, Yacobi módszerével azonban megjósolható, hogy részleges eseménynaplózásnál mekkora az a legnagyobb összeg, ami az egyes tárcatípusokból ellopható. Az egyenleg típusú e-pénznél ez exponenciálisan nő, de az egyedi azonosítóval ellátott érmék feltörésével nem lehet ekkorát kaszálni.
Yacobi kutatótársai, Kuch és England már kísérleteznek egy e-cash rendszer létrehozásával. A következő életszerű eseteket vizsgálják:

- Kisértékű vásárlások az Interneten: például, egy hírmagazin weboldalának látogatója rákattint a kiválasztott cikk aktív hivatkozására, és néhány forint ellenértékért azonnal elolvashatja.
- Anonim vásárlás az Interneten, mivel a vásárlók egyre gyanakvóbbak, és félnek attól, hogy a bankkártyaszámuk illetéktelen kezekbe kerül.
- Vásárlás rendhagyó eszközökkel: az ügyfél a tárcáját tarthatja például egy Windows CE-t futtató palmtopon, vagy egy intelligens telefonon. Fizethetünk úgy, hogy a palmtop és az árusító automata infravörös jelekkel rendezi le a dolgot, de úgy is, hogy a telefonba pötyögünk be egy titkos kódot a jegyrendeléshez, miközben épp a moziba tartunk.

Az üzleti élet átalakulása új pénzformákat kíván, amelyek rendelkeznek a pénz összes ismérvével, de felhasználásuk az azokat éltre hívó gazdaság területére korlátozódik.
Az e-cash a maga nemében forradalmi, és ennek megfelelően nagy ellenállást válthat ki. Yacobi szerint a múlt hibáinak nagy tanulsága az, hogy a fontos pénzügyi intézményeket kell megnyerni az ügynek. Mert zseniális gondolat ide vagy oda, az e-cash pénzzé csak a mögötte álló társadalmi bizalom révén válhat, vagyis ha egytől egyig mindenki elfogadja. Ilyen meggyőző erőt azonban jelenleg csak a kormányzati felügyelettel és garanciával rendelkező bankok képviselnek. Bízzunk hát együtt!

Zacco

Kérdezz, felelek!

K: *Hogyan lehetne időlegesen letiltani a System File Protection (SFP) szolgáltatást? Egyetlen fájlt szeretnék lecserélni a SYSTEM32 könyvtárban, de mindig visszajön az eredeti.*
V: Kezdjük azzal, hogy a SFP pontosan az ilyen jellegű fájlcsereberék ellen véd. Többszörös védelem van a Windows 2000-ben a kritikus fájlok lecserélésének megakadályozására, ezek közül a SFP az egyik legdrasztikusabb, de ez érthető, hisz egy hibás eszközmeghajtó *(kernelmódban fut!)* teljesen felboríthatja az operációs rendszert *(kék halál)*. Másik védelmi vonalat képez az eszközmeghajtók digitális aláírása. Az ilyen meghajtók egészen biztosan átestek a Microsoft legszigorúbb tesztjein is *(Kivéve, ha azzal a két tanúsítvánnyal van ellátva, melyet a Verisign oly ostobán vadidegeneknek adott ki a Microsoft helyett – de ez egy másik történet.)* A rendszerből kiszűrhetők a digitális aláírással el nem látott eszközmeghajtók SIGVERIF.EXE futtatásával.

Digitális aláírások ellenőrzése...

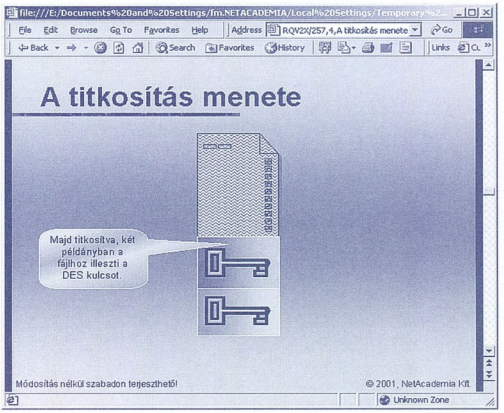
De hogy a kérdésre is választ adjunk: az egyik megoldás lehetne a SFC /cancel parancs használata, ha működne. Azonban van megoldás, mégpedig a registry buherálásával:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\
  CurrentVersion\Winlogon
  SFCDisable REG_DWORD
  0: SFC működik. Ha baj van, kiabál
  1: SFC kikapcs. Következő indításkor rákérdez,
     hogy működjön-e
  2: SCF kikapcs a következő indításra. Magától
     visszakapcsolódik
  4: SFC működik. Nem kiabál
```

Forrás: NetAcademia Windows 2000 lista

K: *Az NT4 SP6-ot szeretném újratolni, de nem megy fel a gépre, mert azon fent van a High Encryption Pack.*
V: Az SP6-ból van külön High Encryption kiadás is. Ennek hiányában kézihajtánnyal juthatunk célba.
1) Bontsuk ki a nem Hi-Sec service packot /x paraméterrel!
2) Keressük meg az update alkönyvtárban az update.inf fájlt!
3) A [CheckSecurity.System32.files] szekcióban a Schannel.dll, Security.dll, és Ntlmssps.dll bejegyzések elé tegyünk pontosvesszőt!
4) Az update.exe-vel indítható a telepítés
Forrás: NetAcademia Windows 2000 lista

K: *Odalett az operációs rendszerem. Ez még nem lenne baj, de újratelepítettem, és az új példánnyal nem tudom elolvasni az EFS-sel titkosított fájlokat. Mennyire reménytelen a helyzet?*
V: Az attól függ. Az EFS titkosítás egy régi, már nem létező felhasználó publikus kulcsával készült ugyan, de minden EFS fájl visszaállítható a titkosításkori Recovery Agent *(RA)* privát kulcsával. Tehát a visszaállíthatóság annak függvénye, hogy fellelhető-e az RA privát kulcsa.
A kulcsokról lásd bővebben az **[1]** címről letölthető PPT mellékletet, mely részletesen elmagyarázza a titkosítás folyamatát.

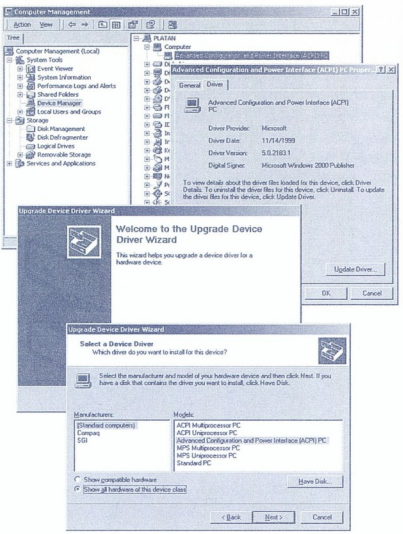


Az [1] címen található animált efs.ppt egy mozzanata

Az RA személye alapértelmezésben a helyi Administrator, illetve tartományi tagság esetén a tartománybéli Administrator! A különbség lényeges: míg a helyi Administrator privát kulcsa valószínűleg megvan a merevlemezen *(hacsak nem formáztuk le a kötetet, de arról volt szó, hogy megvannak az adatok, tehát nem formáztuk le)*, addig a tartományi rendszergazda privát kulcsa tuti, hogy nincs meg lokálisan, hisz csak a publikus kulcs kell a DRF kitöltéséhez – a privát kulcs valahol a rendszergazda nadrágzsebében lapul.
Tehát a visszaállítás „kulcsa", hogy meglegyen az egyik privát kulcs. Érdemes az Administrator jelszavára hajtani, hisz az az ÖSSZES titkosított fájlt el tudja olvasni – lévén ő a RA. A helyi rendszergazda kulcsa megszerezhető, ha megvan még a profilja. Abban csücsül ugyanis a privát kulcs –

ha csak ki nem exportálta valaki flopira. Már csak azt kell elérni, hogy a régi Administrator profilját az új Admin „vegye magára". Ezt könnyedén elintézhetjük egy harmadik felhasználói fiók igénybevételével, amely oldalról „betolja" az új Admin alá a régi profilt. És – elvileg – kész! Nyílnak a doksik!
Forrás: NetAcademia Windows 2000 lista

K: *Windows 2000 Professional alá újabb processzor került, de nem veszi észre. Ugyanúgy kell átállítani, mint NT4 esetén (ResKit UptoMP.EXE)?*
V: Nem. Maga az eljárás hasonló, és a cél is ugyanaz, vagyis a HAL kerül lecserélésre, de ez ma már nem tiltott/tűrt funkció, hanem a beépített Device Manager használható. Az alábbi ábrán a kattintgatásokra előbújó bűvészek és varázslók igazi arca látható:



Uni to Multiprocessor

Értelemszerűen az egyprocesszoros ACPI driver helyett az ACPI Multiprocessor PC választandó, míg ha valakinek Standard PC az eredeti meghajtója, azt MPS Multiprocessor PC-re kell cserélni. A keresztbe cserélés halálos!
Forrás: NetAcademia Windows 2000 lista

K: *Szeretném összenyomni az SQL Server amúgy üres tranzakciónapló fájlját, de hiába shrinkelem, nem megy össze.*
V: Bizony nem! Az SQL Server a tranzakciónaplót körkörösen használja. Ha az a kevés adat, ami éppen benne van, pont a logfájl végén található, akkor mindaddig nem megy össze a fájl, amíg ki nem kergetjük onnan az aktív részt. Ezt némi aktivitás magától megteszi, de mi magunk is írkálhatunk és rollbackelhetünk dummy tranzakciókat, hogy túlforduljon a log. Abban a pillanatban, amint az utolsó aktív tranzakció is kifut a fájl végéről, a log a kívánt méretre rogyik össze.

Az, hogy egy adott pillanatban a log melyik része aktív, a

```
USE adatbázis
Go
dbcc loginfo
go
```

paranccsal nézhető meg. *(Ahol a status=2, az a log darabka aktív.)*

Ha az adatbázisnak csak egy adat- és egy logfájlja van, akkor a legegyszerűbb megoldás:
1) sp_detach_db 'dbname'
2) átnevezzük a logfájlt
3) sp_attach_single_file_db @dbname = 'dbname',
   @physname = 'adatfájl (path-szal együtt)'
4) ekkor az SQL Server készít egy új logot
5) ha minden OK, a régi, átnevezett logfájl törölhető
Forrás: NetAcademia SQL Server 2000 lista

K: *A Command Prompt felokosítására van valamilyen módszer? Állandóan a Command Promptban állok a PING, IPCONFIG és hasonló parancsok miatt, és jó lenne, ha a standard felnyíl mellett egyéb kényelmi funkciók is elérhetők lennének, mint a jó öreg DOS-ban...*
V: A Windows 2000 parancssorát a CMD.EXE adja. Ennek mintegy ezer kapcsolója közül *(érdemes egyszer megnézni a CMD /? outputját)* a /F:on egy nagyon hasznos kényelmi szolgáltatást kapcsol be: a parancsok a fájlok és könyvtárak első néhány betűje alapján egyetlen billentyűleütéssel kiegészíthetők!
- fájlnevek kiegészítése: CTRL+F
- könyvtárnevek kiegészítése: CTRL+D

Forrás: NetAcademia Windows 2000 lista

K: *Ha minden patch fent van a gépen, és minden jog halál szigorú, kell-e félnem a hackerektől?*
V: Természetesen! A NetAcademia Security listán hetente jelennek meg írások újabb hackertrükkökről. Hacking is easy – különösen egy magára hagyott gép ellen. De lássunk konkrét példát: az elmúlt héten kapott valaki az FTPRoot könyvtárába messze tájakról egy ASP lapot. A tüzetesebb vizsgálat kiderítette, hogy ebben egy olyan VBScript lapul, ami kilistázza a lemezegységeket. Ámde az FTPRoot nem futtat ASP-ot, hacsak:
- véletlenül és felelőtlenül nem állítjuk azonos könyvtárra a web és az FTP szolgáltatásokat, és ráadásul engedélyezzük a Scriptek futását, vagy
- mégsincs fenn minden patch, így Hacker Henry a fájlt tovább tudja másolni az IIS alá a közismert *(és régesrég befoltozott)* CMD trükk segítségével...

Forrás: NetAcademia Security lista

A cikkben szereplő URL-ek:
[1] http://technet.netacademia.net/feladatok/ppt

A „**tech.net magazin Brainstorm**" a Dupla KV rovathoz hasonló, ám a személyes kérdésfelvetést és vitát is lehetővé tevő rendezvény, melynek célja:

- **az elsőre talán ismeretlen technológiák élő bemutatása**
- **a cikkekhez kapcsolódó kódok megírása/kipróbálása**
- **a terjedelmi okokból kimaradt információk átadása**

E magazinnal együtt a rendezvényre érvényes belépőjegyet minden előfizetőnkhöz eljuttattuk.
További információk a belépőjegyen olvashatók.

Várjuk Önöket a NetAcademia Mesterkurzusokon

NETACADEMIA
A legjobbakat tanítjuk.

(Bár belépőjegyet adtunk, ez a tény önmagában nem biztosítja helyét a rendezvényen. A jegy célja előfizetőink elsődlegességének biztosítása, de a korlátozott résztvevői létszám (100 fő) miatt a regisztráció kötelező. Jelentkezzen, amíg nem késő!)

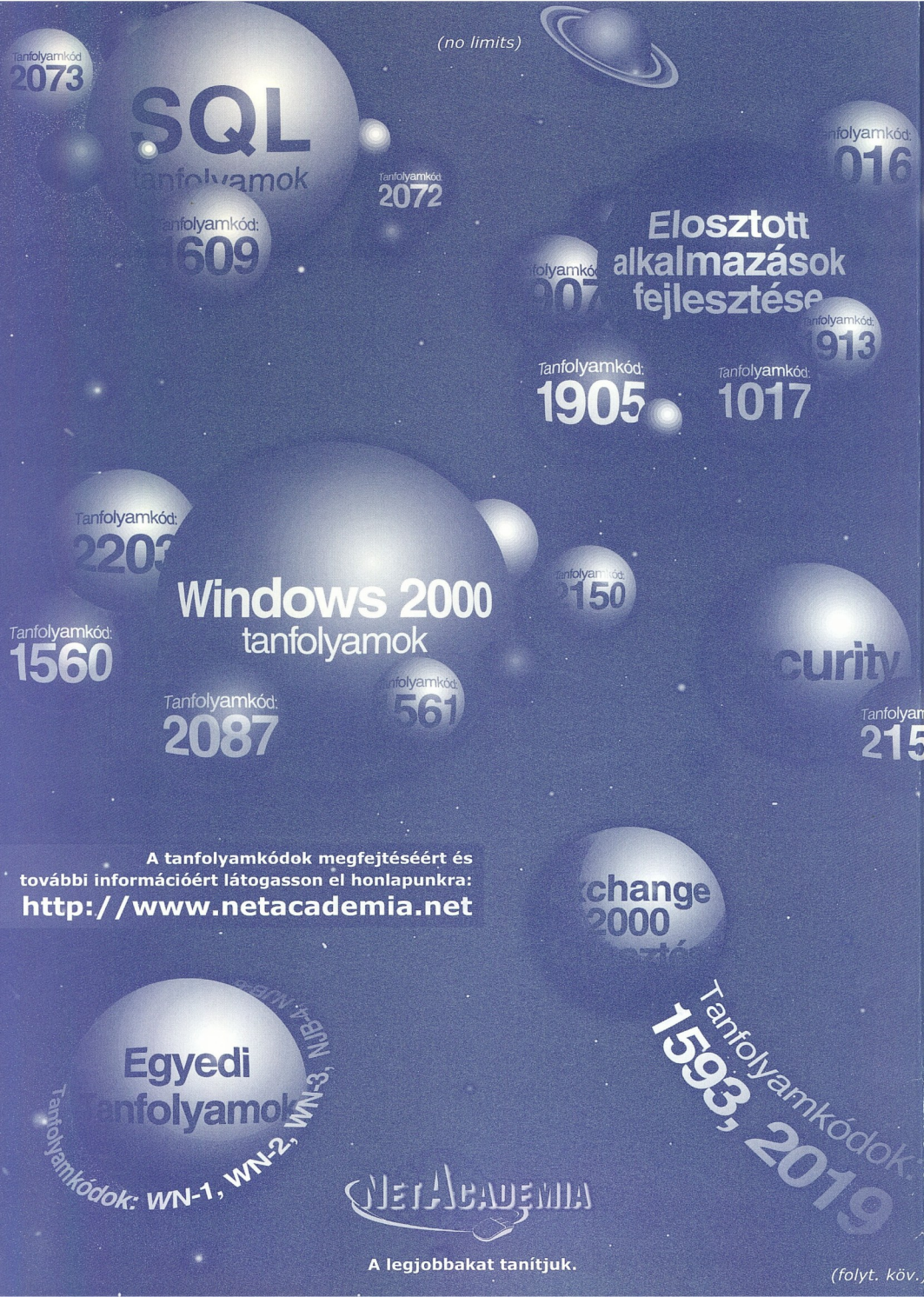
http://technet.netacademia.net/brainstorm

HunNet RL512W

512 kbit/sec-os szimmetrikus forgalomfüggetlen csatlakozás az internetre havi 35.000 Ft-ért

.hu Hungarian Networks

Tel: 06-40-hunnet (486-638)


(no limits)
Tanfolyamkód: 2073
SQL tanfolyamok
Tanfolyamkód: 2072
Tanfolyamkód: 609
Tanfolyamkód: 016
Elosztott alkalmazások fejlesztése
Tanfolyamkód: 913
Tanfolyamkód: 1905
Tanfolyamkód: 1017
Tanfolyamkód: 220
Windows 2000 tanfolyamok
Tanfolyamkód: 150
Tanfolyamkód: 1560
Tanfolyamkód: 561
Tanfolyamkód: 2087
curity
Tanfolyam 215
A tanfolyamkódok megfejtéséért és további információért látogasson el honlapunkra:
http://www.netacademia.net
change 2000
Egyedi tanfolyamok
Tanfolyamkódok: WN-1, WN-2, WN-3, NJB-4, NJB-5
Tanfolyamkódok: 1593, 2019
NETACADEMIA
A legjobbakat tanítjuk.
(folyt. köv.)