tech.net

working with windows

PLUG IN!

III./03.szám
1344 Ft

ISSN 15865185
03
9 771586 518005

RJS

...hely a Nap alatt...

kolokáció

10.00

1132 Victor Hugo u. 18-22.

a hálón:http://ahol.com

mail:info@ahol.com

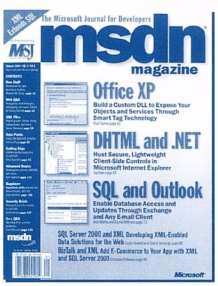
(40)HUNNET

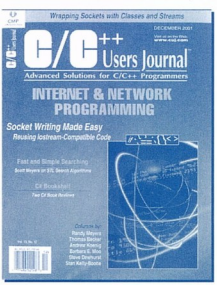
AHOL. MINDENKI TÖBBET KAP

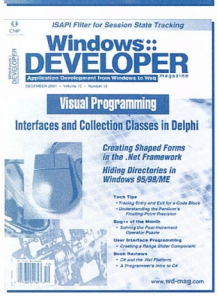
ahol.com

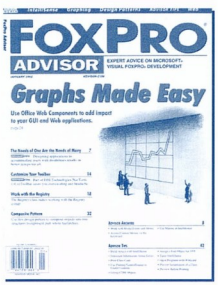
Szeretne előfizetni?

Írjon!









e-mail:

publications@infobyte.hu

Merre vezet a szupersztráda? AC? DC?

tech.net
working with windows

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@technetklub.hu

Kiadja és terjeszti a
NetACADEMIA
A LEGJOBBAKAT TANÍTJUK.

NetAcademia Kft.
Terjesztési, előfizetési információ:
Tel.: 263-2732
terjesztes@netacademia.net
Megjelenik havonta, ára 1.344 Ft
Példányszám: 4.000

NetAcademia © Copyright 2002
Minden jog fenntartva, beleértve *(a részleteket illetően is)* a sokszorosítás, a nyilvános előadás, fordítás jogát. A magazinban közölt cikkeket, képeket és illusztrációkat a kiadó engedélye nélkül közölni, reprodukálni tilos.

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

Hirdetésfelvétel:
BÁRSONYKALAPÁCS
MARKETING
VELVET HAMMER MARKETING
Az ütésből sikert kovácsolunk.
Felelős: **Balogh Zoltán**
Tel.: 489-4665
Fax: 489-4660
info@velvethammer.hu
1027 Budapest, Fő utca 67. V. 1.
Grafikai tervezés, kivitelezés, nyomdai előkészítés:
Bársonykalapács Marketing
Művészeti vezető: **Balogh Zoltán**
Bársonykalapács © Copyright 2002

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

ISSN 1586-5185

Egyenáram

A sajtó, a politika tele van informatikáról szóló hírekkel. A köznép ettől azt hiszi: lemarad, ha kimarad. Ezt a tévhitet erősíti a politika a különböző számítógépvásárlási akciókkal. Lassan boldog-boldogtalan rendelkezni fog otthoni számítógéppel és Internetkapcsolattal. De kérdem én: minek? Egyfelől még mind a mai napig nem tudjuk, **mire** is lesz jó az Internet hétköznapi életünkben *(telefonon pizzát rendelni bájosabban, hagyományos könyvtárban olvasni elmélyültebben lehet stb.)*, sőt, azt sem tudjuk, **milyen eszközzel** fogjuk csinálni azt a valamit, amiről fogalmunk sincs, hogy mi.
Az utca embere tehát vesz egy számítógépet, melyet maximum játszásra használ, majd két év múlva kidobja *(vennél-e bármilyen más játékszert súlyos százezrekért?)*. Ennél nagyobb „károkról" is beszélnek: nemrég olvastam egy felmérést arról, hogy az amerikai nyomtatott sajtó neves lapjai hogyan vészelték át a dotkom-őrületet. Az újságok háromféleképpen közelítették meg a problémát: vagy maguk is beszálltak az Internetre menetelbe *(vagy úgy, hogy a nyomtatott lapot – némi késleltetéssel - egy az egyben kirakták (1. változat), vagy úgy, hogy tartalmi módosításokkal webre szabták(2. változat))*, vagy bele sem kezdtek a „felzárkózásba" *(3. változat)*. Tippeljenek, melyik stratégia hozta a legkisebb veszteséget, mind dollárban, mind előfizetői létszámban? Hát bizony a harmadik. Ennek rengeteg oka van, mindenki tudna mondani legalább hármat.
Wagy wegyük a webkereskedelmet: gyakorlatilag ugyanakkora személyzet és raktár kell hozzá, mint a hagyományos kereskedéshez. Egy látványos különbség van: az áruháznak nem kell csillognia-villognia, mert senki sem tér be hús-vér mivoltában. Ezt a költésmegtakarítást azonban beleölhetjük a magasan képzett informatikus szakembergárdába: a kirakat, és a polcok „átrakodásához" immár nem segédmunkásokat, hanem programozásban jártas szakembereket kell igénybevenni, a biztonsági őr sem lehet akárki, mert az áruházi tolvaj sem hagyományos: a hackerek ellen okos ember kell!

Az 1900-as évek elején már volt villanyvilágítás. A váltóáram „feltalálása" előtt nem lehetett tetszőleges távolságra tenni a fogyasztókat a telepektől, mert a kilométerek „megzabálták az áramot". Ahol villanyvilágítás volt, ott a közelben áramtelepnek is lennie kellett. Izzó tömeggyártás alatt havi háromezer darabot értettek. Amikor felmerült az igény, hogy a közvilágítást is villannyal oldják meg, az akkori korlátok mellett nyilvánvaló volt, hogy a sugárutakat és körutakat nem lehet füzérszerűen kivilágítani, mert egyrész a világ összes villanykörtéje sem lenne elég, másrészt párszáz méterenként telepeket kellett volna letenni. És jött az ötlet: hatalmas villanykörtéket kell készíteni, és fellógatni egy-egy kerület fölé!

Száz év elteltével sem tudunk elszakadni a fenti a sémától: az új technológiát a jelen korlátai mentén próbáljuk különféle célokra használni. Az utókor röhögni fog rajtunk, ez nem vitás.

Váltóáram

Anno a váltóáram pillanatok alatt lesöpörte a föld színéről az egyenáramú energiaszolgáltatást és felhasználást. *(A zseblámpa, a walkman, a digitális fényképezőgépek visszamaradott őskövületek csupán, hisz teleppel működnek. :-)*

Nem vagyok biztos benne, de erősen érzem, hogy a .NET keretrendszer megjelenése hamarosan rendet vág a hagyományos webalkalmazások sorában.

Kell persze alkotó elme is a technológiához. „A webszolgáltatások segítségével akármi megvalósítható!" - mondja a marketing. Azt azonban nem mondja meg, mi az az akármi. Bármi és kész.
Sajnos én sem látom a jövőt, így nem tudnám megmondani, mit takar a bármi. De egy biztos: lassan megszületnek azok a webalkalmazások, amelyek már nem a jelen üzleti folyamatainak webesített változatai, hanem ízig-vérig Internetesek: háló nélkül még csak értelmezni sem lehet majd őket.
Felhasználói felületként pedig akármi szóbajöhet. Akármi alatt olyan eszközöket értek, amit még ki sem találtak.
Valószínűleg most sem Magyarország áll majd a fejlődés élén *(a tízmillió Nobel-díjas országa!)*: figyeljünk a világra! A .NET Akadémia cikksorozat ehhez próbál hozzájárulni.

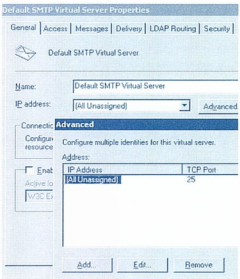
Fóti Marcell
marcellf@netAcademia.net

2002 Március

Exchange 2000

Méltatlanul elfeledett termékre térünk vissza: az Exchange Serverre. Nem a felhasználók feledték el, hanem mi nem törődtünk vele ezidáig megfelelően. Most útjára indítjuk Exchange Server sorozatunkat, mely a telepítéstől kezdve végigvezet a termék használatának rejtelmein.

40. oldal

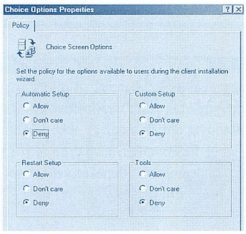


Farkasokkal táncoló

Cluster a gyakorlatban (V. rész)

Ezúttal is alapozómunkát végzünk, folytatjuk az ismerkedést az erőforrástípusokkal. Nem ígérhetek könnyű olvasmányt: minden típusnak megvan a maga bogara, és nem árt ezekkel tisztában lenni, mielőtt használni akarjuk őket. A Microsoft hozzászoktatott minket a varázslók használatához, most viszont bőven akad majd dolgunk „kézzel" is.

5. oldal



RIS

(IV. rész)

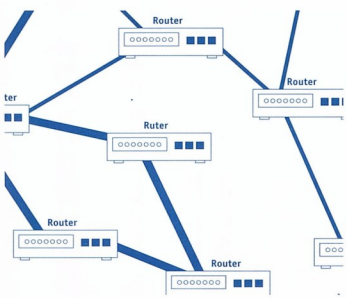
Az elmúlt négy cikkben a RIS-t elég jól megismerhettük minden oldalról, a mostani cikk ráadás. Nem létfontosságú, de azért hasznos dolgokról lesz szó. A cikk első felében a telepítő elindításához használt varázsló, a CIW képernyőiről lesz szó – ismerkedünk velük, majd megpróbáljuk átformálni őket. A másodikban megnézzük több telephelyes környezetben a RIS kiszolgálók elhelyezését, a szinkronizálás lehetőségeit.

8. oldal

Függőségi viszonyok

(I. rész) – At least one service

12. oldal

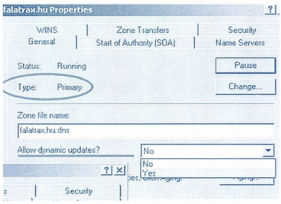


RRAS

Open Shortest Path First

RRAS sorozatunkban utoljára a RIP útválasztási protokollal foglalkoztunk, és megállapítottuk hiányosságait: lassú konvergenciáját, hurokérzékenységét, 15 hopos korlátját – egyszóval butaságát. A RIP ismert gyengeségeit nem a következő verzió kivárásával kell orvosolni, hanem – ha hálózatunk méretével, bonyolultságával a RIP nem tud megbirkózni – áttérhetünk egy sokkal fejlettebb útválasztási protokoll, az OSPF használatára

16. oldal



A windows 2000 és a dinamikus DNS

Mindenki tudja, hogy a Windows 2000 tartományok elsődleges névfeloldó szolgáltatásává *(a WINS-et és egyéb NetBIOS varázslásokat végre leváltva)* a DNS lépett elő. A Windows 2000 munkaállomások a DNS segítségével lépnek be a tartományba, jelentkeznek be a hálózatba, veszik fel egymással a kapcsolatot.

20. oldal

Ki mivel ♥?

Az életmentő LDAP

Januárban indítottuk útjára „**Ki mivel ♥**" rovatunkat, melyben időről időre beszámolunk egy-egy kacifántos, esetleg rejtélyes „ügy" megoldásáról. Ez a rovat nyitott: ha valaki munkája során olyan kintapasztalatra tesz szert, mellyel mások szenvedését enyhítheti, kérem írjon nekünk a cikktipp@netacademia.net címre!

24. oldal

Mikor és miért *felel az internetszolgáltató?*

A 2001. évi CVII. törvény legfontosabb előremutató rendelkezése a szolgáltatói felelősség szabályozása. A kérdés alapvetően természetesen az, hogy a hálóra felkerült tartalomért a szolgáltatást nyújtók mely csoportja, és milyen mértékű felelősséggel tartozik.

26. oldal

ASP Suli

Outlook Wap Acces

Az elmúlt két hónapban megismerkedtünk a WML programozás alapjaival. Most itt az ideje, hogy valami maradandót alkossunk: cikkünkben az Exchange 2000 CDO objektummodellje segítségével megvalósítjuk az Outlook Web Access szuperkompakt változatát: ez lesz az Outlook Wap Access.

28. oldal

.Net akadémia

III. rész

Az előző két részben megtekintettük az osztályok alapvető lakóit, megnéztük, hogyan kezeli a .NET a típusaink által igényelt memóriát. Most továbbhaladunk az objektumorientált jellemzők mentén, és megnézünk néhány fejlettebb típust, amelyek nagy segítséget jelentenek áttekinthető, könnyen bővíthető és karbantartható programok írásában.

32. oldal

XMLgessünk

Átjárás a relációs és az xml világ között

Folytatjuk a relációs adatbázisok és az xml világ közötti átjárást. Tovább boncolgatjuk a DataSet osztály szolgáltatásait, és megszemlélünk egy valódi virtuózt, az XmlDataDocument objektumot.

36. oldal

Aki kíváncsi...

44. oldal

Farkasokkal táncoló
(V. rész) - Cluster a gyakorlatban

Ezúttal is alapozómunkát végzünk, folytatjuk az ismerkedést az erőforrástípusokkal. Nem ígérhetek könnyű olvasmányt: minden típusnak megvan a maga bogara, és nem árt ezekkel tisztában lenni, mielőtt használni akarjuk őket. A Microsoft hozzászoktatott minket a varázslók használatához, most viszont bőven akad majd dolgunk „kézzel" is.

Olyan infrastrukturális szolgáltatásokról szólunk, mint a DTC és az IIS, az SMTP, az NNTP, a WINS és a DHCP.

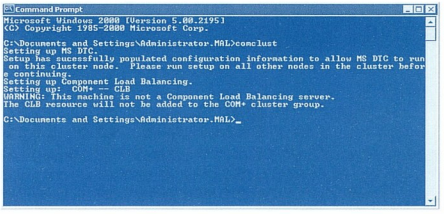
A Distribution Transaction Coordinator

Őszintén bevallom, nem tartozom a fejlesztők csapatába, ezért nem sok figyelmet szenteltem az MSDTC szolgáltatásnak, a fürtszolgáltatással való ismerkedésem során pedig sosem gondoltam volna, hogy ezzel az erőforrással valaha találkozni fogok. Holott igen fontos dologról van szó. A Microsoft meglehetősen sok alkalmazását készítette el úgy, hogy számít az MSDTC meglétére – talán ezért is került be a Windows 2000 operációs rendszerbe. Ilyen alkalmazás az SQL, a BizTalk, az Application Server, a Commerce Server, a Message Queue Server és bizonyos esetekben az IIS is. Olyannyira fontos lehet ez, hogy például az MSDTC-t mindenképp az SQL előtt érdemes fürtösíteni. Akik tehát ilyen alkalmazásokat üzemeltetnek majd fürtkiszolgáló környezetben, azoknak érdemes megismerkedniük az MSDTC erőforrással.

A Distributed Transaction Coordinator talán az egyik legfurább erőforrás. Két érdekes tulajdonsága is van. Minden leírás szerint *(lásd a cikk végén a KB hivatkozásokat)* egyetlen példány elég belőle fürtönként, vagyis úgy viselkedik, mint a quorum lemez vagy a többi clustercsoportbeli erőforrás. A Q294209 szerint a clustercsoportban a helye, én azonban nem osztom ezt a véleményt, mert a saját tapasztalatom az, hogy az MSDTC jó pár más erőforrást is magával „ragad", ami nem kívánatos. A másik érdekes tulajdonsága, szintén minden forrás szerint, hogy nem a szokásos módon és felületről lehet létrehozni. No, nézzük ezt a csodabogarat!

- Először győződjünk meg róla, hogy az MSDTC mindkét állomáson fut. *(Az biztos, hogy létezik ilyen service, mert a Windows 2000 nem is telepíthető e nélkül).*
- Válasszuk ki azt a csoportot, ahol az erőforrást létre akarjuk hozni. Előfeltétel, hogy ebben a csoportban legyen egy lemez, egy hálózati név és egy IP cím erőforrás.
- Hozzuk létre az MSDTC-t az erőforrás varázslóval.
- Tegyük függővé a hálózati névtől.
- Hagyjuk az MSDTC-t offline állapotban.
- Hozzunk létre a csoport fizikai lemezén egy dtclog könyvtárat.
- Az egyik állomás Winnt\system32\dtclog könyvtárából másoljuk át a fájlokat a közös meghajtón lévő mappába.
- Tegyük függővé attól a fizikai lemez erőforrástól, amely az előző könyvtárat tárolja.
- Ha minden rendben van, akkor futtassuk a comclust.exe programot parancssorból az első állomáson, majd a másodikon is.

A comclust egy clb erőforrást is hozzáadna a clusterhez, de ez nem sikerül neki, a figyelmeztetését nem kell komolyan venni. A lényeg, hogy az MSDTC létrejött. *(Lásd 1. ábra)*



Az MSDTC erőforrást létrehoztuk. A COM+ -ra vonatkozó információkat figyelmen kívül hagyhatjuk.

Miért van szükség erre a hókusz-pókuszra? Nos, tapasztalataim alapján azért, mert az erőforrás varázsló és az MSDTC nincs összecsiszolva. A varázsló nem mozgatja át a már meglévő logokat a közös lemezre. Kitalálták tehát a fejlesztők a comclust programot, amely képes egy teljes MSDTC erőforrást létrehozni a következő változásokkal:

- Létrehozza az erőforrást.
- Létrehoz egy közös lemezen egy könyvtárat a tranzakció logoknak.
- Átmásolja a regisztrációs adatbázis megfelelő részét a cluster hive alá.

Csakhogy a comclust az első lehetséges csoportot választja ki, ahol van fizikai lemez, IP cím és hálózati név. Az első csoport általában a cluster erőforráscsoport, ide kerülne be az MSDTC, amely egyáltalán nem kívánatos. Ezért kell „rábeszélni" a segédprogramot, hogy a mi elképzeléseinket kövesse, és az általunk preferált csoportba kerüljön az MSDTC.

Mindenképpen javaslom, hogy a sikeres létrehozás után teszteljük az át- és visszaköltözést. Az MSDTC-től függő újabb erőforrások telepítése után *(SQL, Biztalk stb.)* pedig újra végezzük el a tesztet. Előfordulhat persze, hogy a kiszolgálófürtöt úgy kaptuk meg, hogy a tranzakció koordinátor a cluster csoportban van. Ha szeretnénk más csoportba áthelyezni, van rá mód, a Q243204 cikkben leírtak szerint. Mondanom sem kell, ahogy nem használható az egyszerű erőforrás-varázsló a létrehozáskor, úgy nem lehet egy egérkattintással áthelyezni sem az MSDTC-t.

Nehéz volt? Lehetett volna ennél sokkal bonyolultabb is a létrehozás, akit érdekel a cikk végén talál némi irodalmat. Ha pedig valaki aggódna, hogy minden könyvtár-költöztetés problémát jelent majd, azokat szeretném megnyugtatni: elég, ha továbbolvassa a cikket, a DHCP és a WINS nem szenved ilyen gondoktól.

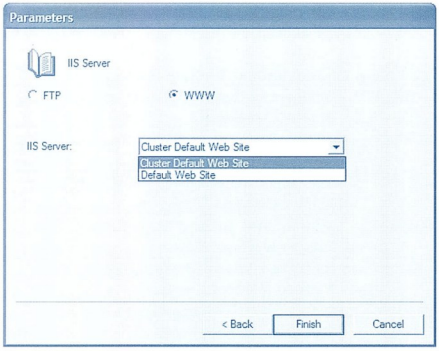
Az Internet Information Server

Most, hogy már minden csínját-bínját ismerjük az MS DTC létrehozásának és életben tartásának, nekiugorhatunk az IIS-nek is. Azért kellett a tranzakció-koordinátorral kezdeni, mert a webhelyünk esetleg tranzakciókat használ. Ilyenkor az IIS erőforrást függővé kell tenni az MSDTC-től, egyébként – természetesen – nem.

Ahogy azt korábban is tettük, győződjünk meg, hogy az IIS mindkét állomáson működik. Semmi nem említi ugyan, de célszerű ugyanazokat a szoftverkomponenseket telepíteni mindkét szerveren, rejtett hibákat lehet így elkerülni.

A továbbiakban feltételezem, hogy már létezik egy erőforráscsoport, amelynek van legalább egy lemezerőforrása, egy IP címe és egy hálózati neve, továbbá, hogy ebben a csoportban definiáltuk az MSDTC erőforrást is.

- Hozzunk létre a lemezerőforráson egy könyvtárat, amely majd a publikálandó anyagainkat tartalmazza. Lehet ez például N:\InetPub\WWWRoot. Helyezzünk el a könyvtárban valamilyen tartalmat, hogy később tesztelni tudjuk a működését.
- Hozzunk létre egy IP cím erőforrást. Ez lesz majd az IIS szerverünk IP címe. Tegyük az IP címet függővé az MSDTC erőforrástól.
- Az egyik állomáson, – célszerűen azon, amelyik a csoportunkat birtokolja – hozzunk létre az IIS MMC beépülő moduljával egy új webhelyet (Web Site). Az IP cím és az alapértelmezett könyvtár legyenek azok, amelyekről az előző két pontban szó volt.
- A cluster administrator segítségével hozzunk létre az erőforráscsoportunkban egy IIS erőforráspéldányt. A szokásos kérdéseken túl csak arra kell válaszolnunk, hogy melyik Webhelyet akarjuk fürtözni, s hogy annak mi a típusa.



Az erőforrásvarázsló utolsó kérdése az IIS erőforrás létrehozásakor

Ezzel az első állomáson végeztünk - de csak az elsőn! Rá kellene venni a másik állomást, hogy az IIS konfigurációja ugyanaz legyen, mint elsőé. Másképp fogalmazva: a két állomás metabase adatbázisát szinkronizálni kell. IIS 4.0-tól létezik egy segédprogram, amely épp erre hivatott, iissync a neve. Használatának az a feltétele, hogy az anonim felhasználókat megszemélyesítő felhasználói fiókok *(IUSR_kiszolgálónév, IWAM_kiszolgálónév)* azonosak legyenek mindkét állomáson. Célszerű tehát létrehozni egy IUSR_cluster és IWAM_cluster felhasználót, majd „Log on locally" felhasználói jogot adni nekik a fürt állomásaira vonatkozóan. Ezt az alábbi lépéseknek kell követnie:

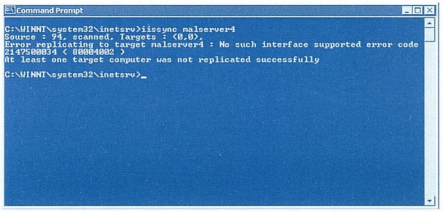
- Az Internet Services Managerben ki kell választani a friss, fürtözött webhelyet, és a „Directory security" tulajdonságlapján módosítani kell a névtelen hozzáférések fiókját az új IUSR_cluster névre.
- A %systemdrive%:\inetpub\adminscripts könyvtárból le kell futtatni az alábbi parancsokat:

```
Cscript adsutil.vbs SET W3SVC/WAMUserName
↳  domain\IWAM_cluster
Cscript adsutil.vbs SET W3SVC/WAMUserPass "jelszo"
```

- Végezetül kiadhatjuk a régen várt parancsot arról az állomásról, amelyik a helyes IIS beállításokat tartalmazza:

```
iissync masikallomasneve
```

Előfordulhat persze, hogy elsőre nem sikerül a szinkronizálás, ahogy azt az ábra is mutatja.



Ez a szinkronizálás még nem sikerült...

A sikertelenség senkit ne keseritsen el. A Q224801 cikk a hibakódokat és a lehetséges okokat fejti vissza. A fenti hiba oka valószínűleg az, hogy a két állomáson nem azonos javítócsomag van. Ha sikeresen lefutott az IISSYNC, akkor már csak azt kell tesztelni, hogy az át- és visszaköltözés problémamentes-e, s máris készen vagyunk: előállt egy hibatűrő IIS szerver.

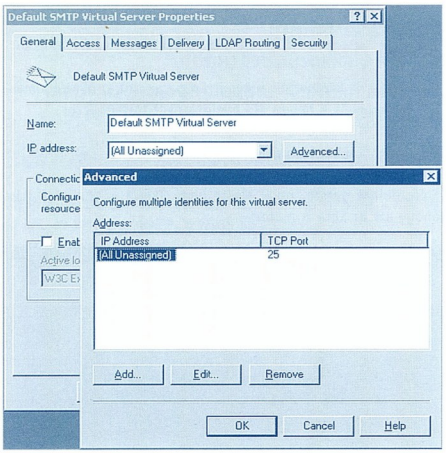
Megkérdezheti a Kedves Olvasó, hogy vajon mi értelme van egyáltalán az IIS ilyetén konfigurálásának, hiszen van sokkal jobb módszer is nagy rendelkezésre állású webfarm létrehozására, mégpedig az NLBS. Nos, a válasz az, hogy a Kedves Olvasónak igaza van, amennyiben az IIS a „cél". Ám nem szabad elfelejteni, hogy az IIS maga lehet „eszköz" is, amelyet egy másik alkalmazás kíván használni. Ha egyenszilárdságot akarunk, és mindkettőt azonos szerveren helyezzük el, akkor ésszerű mindkettőnek azonos rendelkezésre állást biztosítani. Ezért van értelme az IIS erőforrás létrehozásának.

Az SMTP és az NNTP erőforrás

Az Internet Information Serverhez hasonlóan fürtözhető az IIS-hez járó SMTP és NNTP szolgáltatás is, méghozzá az IIS-hez egészen hasonló módon. Ezeknél a szolgáltatásoknál csupán az a fontos, hogy a fürtökre feltett SMTP más porton „figyeljen", mint a virtuális szerveren definiált. A szolgáltatásokhoz előfeltétel az IIS fürtösítése, vagyis mindaz, amit eddig leírtunk. Lássuk, hogyan működik az SMTP fürt a gyakorlatban:

- Győződjünk meg róla, hogy mindkét állomáson telepítve van az SMTP. *(Ha az IIS-nél precízen dolgoztunk, akkor ez nem gond.)*
- Győződjünk meg arról, hogy mindkét állomáson kézi indítású az SMTP szolgáltatás, és LocalSystem fiók alatt fut.
- Indítsuk el az adminisztrációs eszközök közül a Computer Managert, és az Internet Information Services modulban vá-

lasszuk ki az alapértelmezett SMTP virtuális szerver tulajdonságait. Az első lapon látható „Advanced...” gombra kattintva átállíthatjuk az állomás SMTP szerverének alapértelmezett portját. Erre azért van szükség, hogy a node képes legyen fogadni egy olyan virtuális SMTP kiszolgálót, amelyik majd a 25-ös, tehát a normál portot használja.



A fürtkiszolgáló felkészítése nagy rendelkezésre állású SMTP kiszolgáló fogadására

- Ezután – akárcsak az IIS esetében – hozzunk létre egy új virtuális SMTP kiszolgálót, amelyet majd fürtözni fogunk.
- Miután ezzel is végeztünk, a cluster administrator segítségével létrehozhatjuk az SMTP erőforrást, amelyet függővé kell tenni az IIS IP címétől, magától az IIS erőforrástól, és a fizikai lemeztől, amely a mailroot könyvtárat tartalmazza.
- Az utolsó előtti lépés az IISSYNC futtatása, valamint a másik node alapértelmezett SMTP virtuális szervere portjának átállítása.
- Végezetül tesztelni kell, hogy az át- és visszaköltözés problémamentesen működik.

Tekintve, hogy a NNTP szolgáltatás ikertestvére az SMTP-nek, nem válok felületessé, ha azt mondom: csak ugyanúgy, mint az SMTP protokollnál.

A Windows Internet Name Service és a Dynamic Host Configuration Protocol erőforrások

A cikk elején már említettem, hogy a WINS és DHCP erőforrások korántsem olyan problémásak, mint például az MSDTC. Valóságos felüdülés a telepítés az előzőkhöz képest. A clusterszolgáltatáshoz képest a telepítés bármilyen sorrendben történhet, vagyis akár utólag is dönthetünk úgy, hogy WINS-t, vagy DHCP-t fürtözünk. Arra persze ügyelni kell, hogy mindkét állomáson telepítsük a szolgáltatásokat. A varázsló rákérdez a függőségekre *(fizikai lemez, hálózati név, IP cím)*, továbbá definiálnunk kell az adatbázisok helyét, amelyeknek természetesen egy fizikai lemez fürterőforráson kell elhelyezkedniük. Néhány apróságot azért még érdemes megjegyezni.

A DHCP szolgáltatást hitelesítenünk kell, amennyiben Active Directoryt használunk. Figyeljünk, hogy a hitelesítés alatt ne az aktuális állomás, hanem a virtuális kiszolgáló IP címét jegyezze be a DHCP Administrator, különben az átköltözésnél gondok lesznek. A WINS szolgáltatásnál ügyelni kell, hogy a fürtkiszolgáló általában „multihomed” számítógép. A legegyszerűbb, ha a privát hálózaton letiltjuk a NetBios interface-t, ahogy azt a Q193890 is írja. A WINS beállítható úgy, hogy másolatot készítsen a saját adatbázisáról. Az adatbázis megsérülése, vagy elvesztése esetén ez hasznos lehet. Csakhogy a fürtnél a másolatnak is közös lemezen kell elhelyezkednie, amelyet mindkét állomás elér, vagyis a másolat együtt veszne az eredetivel. A Q283290 választ ad a problémára – de erről még szó esik majd a következő cikkekben. A két szolgáltatással kapcsolatban jogosan vetődik fel a kérdés: miért vesződött a Microsoft a fürtözéssel, hisz mindkettőnél létezik jól működő, nem cluster alapú redundáns konfiguráció is? A válasz egyszerű: a két szolgáltatás tranzakcióalapú adatbáziskezelést végez. Íme, itt a példa, hogyan lehet fürtkompatibilis alkalmazást írni. Persze nem csak ez volt az egyetlen érv. A WINS és a DHCP, bár kicsi és láthatatlan szolgáltatások, mégis az IP alapú Windows hálózatok alappillérei. Azok pedig legyenek nagyon megbízhatóak. Nos, a Windows 2000 megjelenésével azok lettek.

Lepenye Tamás, MCSE 2000
lepenyet@mal.hu

Q243204	MSDTC Disaster Recovery Techniques
Q248025	Configure Clustered IIS Virtual Servers on Win2000 Adv Server
Q290624	HOWTO: Configure MSDTC in a Windows 2000 Cluster Environment
Q294209	INF: Rebuilding or Moving MSDTC Used with a Failover Clustered SQL Server
Q249603	Using IISSYNC to Synchronize Clustered Web Sites on Windows 2000
Q248025	Configure Clustered IIS Virtual Servers on Win2000 Adv Server
Q280400	Configuring SMTP Resource on a Windows 2000-Based Server Cluster
Q258693	CLUSTER: IIS Server Instance Resources Offline After NNTP Install
Q283290	Backing Up a WINS Database on a Windows 2000 Cluster
Q193890	Recommend WINS Configuration for Microsoft Cluster Server
Q226796	Using WINS and DHCP with the Windows 2000 Cluster Service

RIS
(V. rész)

Az elmúlt négy cikkben a RIS-t elég jól megismerhettük minden oldalról, a mostani cikk ráadás. Nem létfontosságú, de azért hasznos dolgokról lesz szó. A cikk első felében a telepítő elindításához használt varázsló, a CIW képernyőiről lesz szó – ismerkedünk velük, majd megpróbáljuk átformálni őket. A másodikban megnézzük több telephelyes környezetben a RIS kiszolgálók elhelyezését, a szinkronizálás lehetőségeit.

Ügyféltelepítő varázsló *(Client Installation Wizard – CIW)* képernyők

Ezek a képernyők jelennek meg alapértelmezésben a RIS ügyfelek telepítésének kezdetén. Sajnos a RIS-es ügyféltelepítés ezen részét nem lehet teljesen automatizálni, minimális beavatkozásra itt mindenképp szükség van, de ha egyszer végre elindul a telepítő akkor nyugodtan magára hagyhatjuk a masinát.

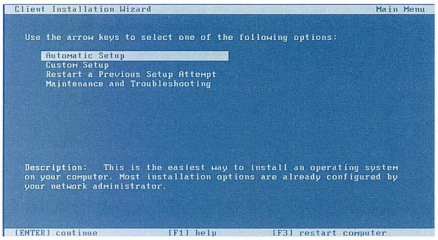
A RIS kiszolgáló CIW képernyői alapértelmezett telepítés esetén a kiszolgáló \RemoteInstall\OSChooser\ alatt levő, nyelvi verziónak megfelelő könyvtárában találhatók, kivéve egyet, mert az rögtön az OSChooser könyvtárban található. Ez nem más mint a welcome.osc, ami mindig a legelső képernyő. Van itt mégegy fájl a multilng.osc, ami nem más mint egy módosított welcome.osc. Erre bizony szükségünk lehet, ha például a RIS kiszolgálónak elállítottuk a területi beállításait. Ha mondjuk egy angol verziójú Windows 2000 kiszolgálón a területi beállítás magyar, akkor a RIS ügyfél telepítésekor egy szép hibaüzenettel elszáll a telepítés mindjárt az első képernyő után, ugyanis a RIS kiszolgáló ilyenkor automatikusan az OSChooser\Hungarian könyvtárban levő CIW képernyőket akarja használni az ügyfél. Ezt a hibát elkerülhetjük, ha a multilng.osc-t átnevezzük welcome.osc-re. Ekkor az első képernyő egy nyelvválasztó képernyő lesz.

Mielőtt belenyúlnánk a képernyőfájlokba, nézzük meg őket sorrendben, ahogy következnek.

A legelső képernyő a welcome.osc, ahogy az előbb mondtam, ez mindenképp szükséges. Innen egy **[Enter]** vezet tovább.

Jön a logon.osc. Gondolom nem nehéz kitalálni, hogy ez egy bejelentkező lépernyő, meg kell adnunk felhasználót, jelszót és tartománynevet a bejelentkezéshez.

Az autentikáció után alapértelmezés szerint a choice.osc jelenik meg, a következő választási lehetőségekkel:



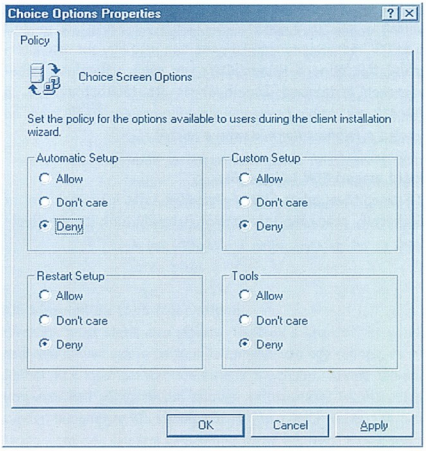
Choice.osc

Automatic Setup – senki nem lepődik meg, ha azt mondom, hogy ezzel rögtön elindíthatjuk a telepítést. Ezt választva az osauto.osc lesz a következő képernyő, ahol a lehetséges telepítőkészletekből választhatunk.

Custom Setup – ez a menüpont a custom.osc-t fogja meghívni. Ezt választva lehetőségünk van arra, hogy a telepítőnek egyedi információkat adjunk át, így például a gépnevet, helyi rendszergazda jelszót, vagy akár képernyő felbontáshoz beállításokat. Ezzel kicsit később még részletesen foglalkozunk.

Restart a Previous Setup Attempt – Egy előzőleg megszakadt telepítést folytathatunk. Mit is jelent ez? A telepítés során a RIS kiszolgálón átmenetileg tárolódik minden ügyfélhez \RemoteInstall\tmp könyvtárban a GUID-nak megfelelő sif kiterjesztésű válaszfájl, ami nem más, mint az adott gép telepítéséhez ténylegesen használt fájl. Ha ezt a menüpontot választjuk, akkor az előzőleg már létrehozott egyedi SIF fájl alapján fogja az ügyfelet telepíteni.

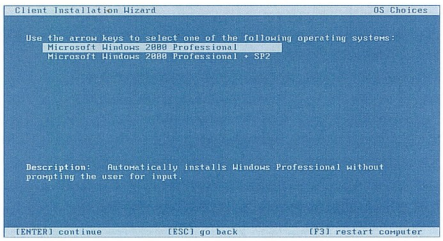
Maintenance and Troubleshooting – Itt elvileg BIOS frissítéseket, illetve egyéb diagnosztikai programokat választhatnánk, ha lennének. Ilyen programokat többnyire OEM gyártóktól lehet beszerezni. Fontos megjegyezni, hogy ezt az egész képernyőt kihagyhatjuk, vagy beállíthatjuk, hogy csak egyes menüpontok legyenek elérhetőek. Ezt a Group Policyben kell módosítani.



Policy\User Configuration\Windows Settings\Remote Installation Services

Ha mindent megtiltunk, mint a fentebb látható képen, akkor nem is jelenik meg a choice.osc, hanem egyből a rendelkezésre álló telepítő készletek listáját láthatjuk – az oschoice.osc képernyőt.

Az choice.osc után a checkguid.osc jön sorban és csendben, látható jel nélkül ellenőrzi az Active Directoryban az ügyfél GUID-ot. Ha nem talál az AD-ban azonos GUID-dal és névvel objektumot, akkor jön az oschoice.osc a választható telepítőkészletek listájával.



oschoice.osc

A felhasználó itt kiválasztja, mit is óhajt telepíteni. Mivel a kiszolgálón levő SIF fájlok NTFS jogosultságaitól függ, mi jelenik itt meg a felhasználónak, nagyon egyszerűen szabályozható a hozzáférés. Amelyik SIF fájlhoz legalább READ joga van az előzőleg bejelentkezett felhasználónak, azok megjelennek itt a listában, a többiek pedig nem.

Miután kiválasztottuk a telepítőkészletet és „Enterelтünk", megjelenik egy figyelmeztető képernyő, a warning.osc, ami arra figyelmeztet, hogy a helyi merevlemez formázása következik. Ha ezen is továbblépünk, még megjelenik egy összegző képernyő, az install.osc amelyen az ügyfél és a RIS kiszolgáló neve, valamint az ügyfél GUID olvasható le. Ha ezen a ponton is továbblépünk, valóban elkezdődik a telepítés, és innen már tényleg automatizált a folyamat.

Van az OSChooser\language könyvtárban még egy sor .osc kiterjesztésű fájl. Minden képernyőhöz van súgó, valamint a hibaüzeneteket tartalmazó képernyők is itt találhatóak *(pl. a 00004e28.osc, amely bejelentkezési hiba esetén jelenik meg, ha helytelen a felhasználói név vagy a jelszó)*.

Csináld magad CIW képernyők

A RIS telepítővel adott CIW képernyőket saját igényeink szerint alakíthatjuk, akár több információt is bevihetünk itt, amellyel úgy tudjuk egyedivé tenni a telepítést, hogy nem kell több telepítőkészletet létrehoznunk. Arra gondolok, hogy például ha nem akarjuk a helyi rendszergazda jelszavát beletenni a SIF fájlba, vagy az adott géphez kötött monitor miatt más képernyőfelbontást szeretnénk beállítani, akkor saját gyártású CIW képernyőkkel bővíthetjük a varázslót a cél elérése érdekében.

A CIW képernyők tulajdonképpen OSCML *(OSChooser Markup Language)* nyelven íródott szöveges állományok .osc kiterjesztéssel. Az OSCML a HTML 2.0 verziójához hasonlít leginkább. Az OSCML-bem használatos jelekre csak annyira térek ki, amennyire a példához szükséges, a Windows 2000 Resource Kit és RIS-ről szóló White Paper is foglalkozik vele részletesen.

Nézzünk inkább konkrét példát. Vegyük azt az egyszerű esetet, amikor minden géphez közvetlenül a telepítés **előtt** szeretnénk beállítani a képernyő tulajdonságait.

Ehhez nem kell mást tenni, mint létrehozni egy - mondjuk - display.osc nevű képernyőt és meg kell változtatni a choice.osc-t. Az alapértelmezett .osc képernyők szerkesztése előtt feltétlenül készítsünk másolatot az eredeti fájlokról, például a *.orig kiterjesztést használva.

Egyszerű szövegszerkesztő, például a Notepad.exe használatával nyissuk meg a Choice.osc fájlt. A következő ábra az alapértelmezett beállításokat mutatja.

```
<OSCML>
<META KEY=F3 ACTION="REBOOT">
<META KEY=F1 HREF="CHOICHLP">
<META SERVER ACTION="DHRESET">
<META SERVER ACTION="FILTER CHOICE">
<TITLE>  Client Installation Wizard                                    Main Menu</TITLE>
<FOOTER>  [ENTER] continue          [F1] help          [F3] restart computer</FOOTER>
<BODY left=5 right=75>
<br>
<br>
Use the arrow keys to select one of the following options:<br>
<P left=8>
<FORM>
<SELECT SIZE=10>
<OPTION VALUE="OSAUTO" TIP="This is the easiest way to install an operating system on your
computer. Most installation options are already configured by your network administrator.">
Automatic Setup
<OPTION VALUE="CUSTOM" TIP="With this option, you can define a unique name for this computer
and specify where the computer account will be created within the directory service.
Select this option if you are setting up this computer for someone else within your company.">
Custom Setup
<OPTION VALUE="RESTART" TIP="A previous remote installation attempt has been detected on this
computer. Select this option to restart a previously started installation.">
Restart a Previous Setup Attempt
<OPTION VALUE="TOOLS" TIP="This option gives you access to tools for keeping your computer
up-to-date and for troubleshooting problems.">
Maintenance and Troubleshooting
</SELECT>
</FORM>
</P>
<BR>
<BOLD>Description:</BOLD>  
<TIPAREA>
</BODY>
</OSCML>
```

Az eredeti choice.osc „hátulról"

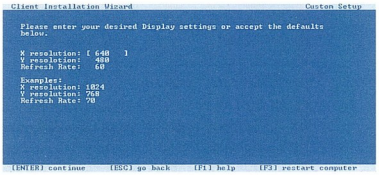
Meg kell változtatni az <OPTION VALUE="OSAUTO" > sort úgy, hogy a megfelelő képernyőre ugorjon, ami az Oschoice.osc. Tehát <OPTION VALUE="OSAUTO" > helyébe az <OPTION VALUE="OSCHOICE" > lép.

Ez lehetővé teszi, hogy kiválasszuk a telepítő készletet, még mielőtt a felhasználónak bármiféle információt meg kellene adnia. Át kell írni a leírást is *(TIP=)*, majd a megjelenítendő részt „Automatic Setup"-ról mondjuk „Windows 2000 Setup"-ra. Az egyedi választást tartalmazó sort távolítsuk el, ez az <OPTION VALUE="CUSTOM" > kezdetű.

A változtatások után így néz ki a fájl:

```
<OSCML>
<META KEY=F3 ACTION="REBOOT">
<META KEY=F1 HREF="CHOICHLP">
<META SERVER ACTION="DNRESET">
<META SERVER ACTION="FILTER CHOICE">
<TITLE>  Client Installation Wizard                                    Main Menu</TITLE>
<FOOTER>  [ENTER] continue          [F1] help          [F3] restart computer</FOOTER>
<BODY left=5 right=75>
<br>
<br>
Use the arrow keys to select one of the following options:<br>
<P left=8>
<FORM>
<SELECT SIZE=10>
<OPTION VALUE="OSCHOICE" TIP="This is the easiest way to install an operating system on your
computer. Most installation options are already configured by your network administrator.">
Windows 2000 Setup
<OPTION VALUE="RESTART" TIP="A previous remote installation attempt has been detected on this
computer. Select this option to restart a previously started installation.">
Restart a Previous Setup Attempt
<OPTION VALUE="TOOLS" TIP="This option gives you access to tools for keeping your computer
up-to-date and for troubleshooting problems.">
Maintenance and Troubleshooting
</SELECT>
</FORM>
</P>
<BR>
<BOLD>Description:</BOLD>  
<TIPAREA>
</BODY>
</OSCML>
```

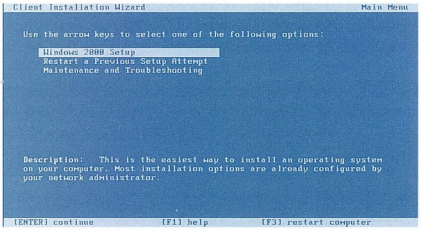
A következő ábrán láthatjuk, hogyan jelenik meg a Display.osc képernyő az telepítéskor.



Ezt látja a felhasználó

Az új choice.osc „hátulról"

A következő ábrán azt látjuk, hogy a CIW-t futtató ügyfélnek mi fog megjelenni, ha a fenti új oschoice.osc-t használjuk:



A módosított choice.osc „elölről"

Ha a felhasználó a Windows 2000 Setup menüpontot választja, akkor a következő képernyőn a kiszolgálón tárolt telepítőkészletek listája jelenik meg. Mivel az Oschoice.osc fájl csak a Template alkönyvtárat nézi át SIF állományokat keresve, ezt a képernyőt nem kell módosítani.
Második lépésként hozzunk létre a display.osc-t. A legegyszerűbb az, ha egy meglevő képernyőt módosítunk, mondjuk a custom.osc-t.

```
<OSCML>
<META KEY=F1 HREF="CUSTHELP">
<META KEY=F3 ACTION="REBOOT">
<META KEY=ESC HREF="CHOICE">
<TITLE>  Client Installation Wizard                                    Custom Setup</TITLE>
<FOOTER>  [ENTER] continue     [ESC] go back     [F1] help     [F3] restart computer</FOOTER>
<BODY left=5 right=75>
<BR>
<BR>
Please enter your desired Display settings or accept the defaults below.
<BR>
<BR>
<FORM ACTION="OSCUST">
X resolution: <INPUT NAME="X-res" VALUE=640 SIZE=4 MAXLENGTH=4><BR>
Y resolution: <INPUT NAME="Y-res" VALUE=480 SIZE=4 MAXLENGTH=4><BR>
Refresh Rate: <INPUT NAME="refresh" VALUE=60 SIZE=3 MAXLENGTH=3><BR>
</FORM>
<BR>
<BOLD>Examples:</BOLD><BR>
        X resolution: 1024<BR>
        Y resolution: 768<BR>
        Refresh Rate: 70<BR>
<BR>
</BODY>
</OSCML>
```

display.osc „hátulról"

Ezen a képernyőn lehet testreszabni a képernyő beállításait. A beírható értékek mellett megadhatunk a felhasználónak egy alapértelmezett értéket. Ha csak az ENTER-t nyomjuk meg, a képernyő felbontása 640x480 lesz.
A példában szereplő display.osc a következő változókat továbbítja a BINL szolgáltatásnak:
X-Res = A képernyő vízszintes felbontása
Y-Res = A képernyő függőleges felbontása
Refresh = A képernyő frissítési frekvenciája
Ahhoz, hogy a megadott értékeket figyelembe is vegye a telepítő, módosítanunk kell a RIS kiszolgálón lévő telepítőkészlet*(ek)* Templates alkönyvtáraiban lévő mintá*(ka)*t. A SIF fájlokban a változók % jelek közé írva szerepelnek. A példánál maradva az általunk beadagolt változók a **[Display]** szakaszba kerülnek be:

```
[Display]
ConfigureAtLogon = 0
BitsPerPel = 8
XResolution = %X-res%
YResolution = %Y-res%
VRefresh = %refresh%
AutoConfirm = 1
```

Nincs más hátra, mint az elkészített display.osc-t be kell építeni a meglevő varázslók közé. Ezt legegyszerűbben úgy tehetjük meg, hogy az oschoice.osc-ben a <FORM ACTION="WARNING"> sort módosítjuk <FORM ACTION="DISPLAY"> -re, így a telepítőkészlet kiválasztása után a beépített display.osc fog megjelenni. A fenti csupán egy egyszerű példa volt a változtatásra, azonban felügyelet nélküli válaszfájlokban megadható valamennyi beállítás számára létrehozhatunk képernyőt, amely a BINL közvetítésével a mintaállományba írja azokat. Így automatizálható, illetve finomhangolható a telepítő anélkül, hogy minden lehetséges esethez külön mintát készítenénk. Akit az ismertetett példa felcsigázott, annak tudom ajánlani a Windows 2000 Resource Kitet, valamint a Microsoft webes oldalait, ott megtalálja a BINL szolgáltatás és az Oschooser belső változóinak teljes listáját, ha az OSCML szóra keres.

RIS kiszolgálók több telephelyen

Evezzünk most kicsit más vizekre, tágítsuk a látókört, és gondoljunk arra mi van akkor, ha több telephelye van egy cégnek, és mindenhol RIS kiszolgálókat szeretnénk használni. Ahány telephely annyiszor kell végigcsinálni minden változtatást? Feltétlenül kell minden telephelyre RIS kiszolgáló? Mi van a DHCP-vel és a DNS-sel? Azonnal sok-sok válaszra váró kérdés merül fel. Nézzük csak sorban. Talán a legelső RIS cikkben volt arról szó, hogy a RIS működéséhez milyen infrastruktúra szükséges. A RIS kiszolgálók elhelyezé-

sénél ezt is figyelembe kell venni. Csak emlékeztetőül mondom, 3 dolog feltétlenül kell: **Active Directory, DNS** és **DHCP**.

Az Active Directory és a DNS mint követelmény több telephely esetén sem okoz problémát, mert az alhálózatok közti útválasztók átengedik ezeket a kéréseket, azonban a DHCP egy kicsit más. Kivesézték a legelső cikkben a RIS ügyfél és a DHCP közt folyó párbeszédet is. Most csak annyit, hogy az ügyfél DHCP Discover broadcastot küld DHCP és RIS kiszolgálót keresve. Ezzel nincs probléma akkor, ha minden telephelyen/alhálózaton van egy-egy DHCP kiszolgáló. Az útválasztók alapértelmezésben nem engedik át a broadcast üzeneteket, ezért több telephelyes/alhálózatos környezetben biztosítanunk kell, hogy a DHCP kérések eljussanak a DHCP kiszolgálóhoz. Ehhez vagy az útválasztó beállításait változtatjuk meg, vagy DHCP Relay Agentet kell telepíteni minden telephelyre/alhálózatra.

Meghatározza a RIS kiszolgálók elhelyezését a telephelyek közti összeköttetések kapacitása is. Nyilvánvaló, hogy kis átviteli sebesség mellett senki nem fog távoli telephelyre RIS segítségével operációs rendszert telepíteni, ilyenkor bizony minden olyan telephelyre RIS kiszolgálót kell telepíteni, ahol azt használni is akarják. Ez riasztó lehet sok telephely esetén, hisz elég sok dolgot kell beállítani ahhoz, hogy rendesen működjön a felügyelet nélküli telepítés. Magában a RIS-ben nincs a szinkronizálásra beépített eszköz, rabszolgamunka lenne mindenhol ugyanazt manuálisan beállítgatni. Nem is kell, ha okosan használjuk a Windows 2000 beépített szolgáltatását,a DFS-t - teljes nevén Distributed File System. Ezzel sok munkát és időt spórolhatunk meg.

Egy mondatban összefoglalva: egy RIS kiszolgálót felépítünk az igényeknek megfelelően, majd a könyvtárstruktúrát DFS segítségével lemásoljuk a többi RIS kiszolgálóra. Az első teljes replikáció befejeztével a másolatok egyenrangú felekké válnak, bárhol változtatunk, az a többi kiszolgálóra replikálódik.

Nézzük ehhez mit is kell tenni:

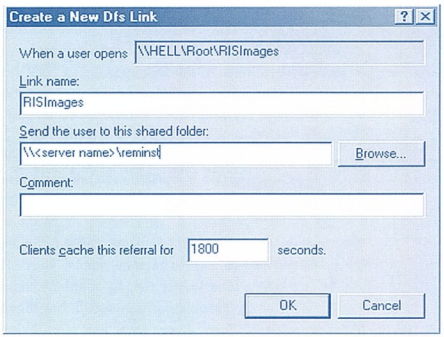
Először is igényeknek megfelelően felépítünk egy RIS kiszolgálót, azaz kialakítjuk a megfelelő könyvtárstruktúrát.

Ezután létre kell hozni egy ún. Domain DFS Root-ot, a következő módon:

1. Indítsuk el a **Distributed File System MMC** snap-int az **Administrative Tools**ból mondjuk a RIS kiszolgálón, vagy bármely tartományban levő gépen
2. Az Action**New DFS Root** varázsló első kérdésénél válasszuk ki, hogy Domain DFS Root-ot szeretnénk létrehozni.
3. Ezután a tartomány neve jelenik meg, amelyben a replikáció működni fog.
4. A következő képernyőn meg kell adni annak az elsődleges kiszolgálónak nevét, amelyik a DFS Rootot tartalmazza. A mi esetünkben itt az imént létrehozott és beállított RIS kiszolgáló nevét kell megadni.
5. Ha ezen is túljutottunk, meg kell adni a Root-hoz egy megosztást, ami egy tetszőleges *(üres)* könyvtár lehet.
6. Ezután még nevet és ismertetőt is fűzhetünk a létrehozott megosztáshoz.
7. Az összefoglaló képernyőn is túljutva elkészül a Domain DFS Root.

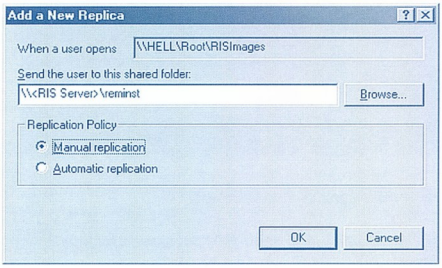
Még csak a beállítások harmadánál járunk, második lépésként létre kell hozni legalább egy DFS Linket is.

Az előzőleg létrehozott DFS rooton jobb klikk – **New Dfs Link** segítségével az alábbi módon létrehozzuk az első RIS kiszolgálónk reminst megosztásához a DFS Linket:



DFS Link a RIS telepítőkönyvtárhoz

Ehhez a DFS Linkhez ezek után Replikákat *(másolatokat)* lehet hozzáadni. A példánál maradva a RISImages Linken jobb klikk – **New Replica** menüpontot választva hozzáadjuk sorban az összes RIS kiszolgáló Reminst megosztásait.



DFS-sel juttatjuk el a telepítőkészleteket a többi RIS kiszolgálóra

Ezek után nincs más dolgunk mint bekapcsolni a replikációt. Ezt úgy tehetjük meg - a példánál maradva-, hogy a RISImages DFS Linken jobb klikk – **Replication Policy** menüt választjuk, ott a legelőször felépített RIS kiszolgálót elsődlegessé tesszük a **Set Master** gombbal, majd a többi kiszolgálót az **Enable** gomb segítségével bekapcsoljuk a replikációba.

Zárszó

A RIS cikksorozat itt véget ér, remélem sok hasznos információval gazdagodtak olvasóim. Az elméleti és a gyakorlati ismertetés mellett igyekeztem minél több, a mindennapi gyakorlatban is jól használható tippet adni amelyek segítségével a RIS minden lehetőségét valóban ki lehet aknázni. Aki úgy érzi nem talált meg valamit a cikkekben a RIS-ről, annak ajánlom figyelmébe a Windows 2000 Resource Kitet, vagy a Microsoft technet-et.

Dorner Csilla
MCSE

Függőségi viszonyok
(I. rész) - At least one service

egy szeg miatt a patkó elveszett,
patkó miatt a ló is elveszett,
a ló miatt a huszár elveszett,
huszár miatt a csata elveszett,
csata miatt az ország elveszett, -
ennyit tesz, lám, egy ici-pici szeg!

Angol gyermekvers

Egyik hajnalban ülök egy kiszolgáló előtt, mely már a nyolcadik kört bootolja, de csak nem akar helyrejönni. Izgatottan várom, kilencedszer is mondja-e, hogy „at least one service failed to start", vagy végre hazamehetünk. Íme az ominózus hibaüzenet, melynek felbukkanásakor minden lehetséges: vagy egy, vagy egyezer szolgáltatás nem indult el:

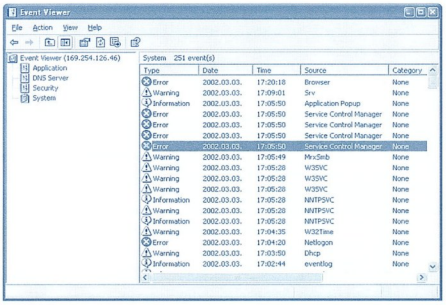
Legalább egy szolgáltatás nem indult el. Egy? Kettő? Vagy több? Ezt az Event Logból olvashatjuk ki.

A Windows belső felépítése a fürt tükrében

A Farkasokkal táncoló... sorozatban igen gyakran esik szó virtuális kiszolgálókról. Emlékeztetőül: ezek azok a „masinák", melyek önálló gépként, saját névvel és IP címmel látszanak a hálózaton, ám valójában nincs mögöttük egy meghatározott vas, amire rábökhetnénk, hogy „íme, ez itt a QQQ nevű kiszolgálónk". Pontosabban nem **egy** meghatározott vas van alattuk, hanem – Advanced Server esetén – **kettő**. A fürt által futtatott virtuális kiszolgálókat magunk építgetjük össze. Mi kell hozzá? Ha adatokat szeretnénk tárolni rajta, kell egy lemez. Ha hálózaton keresztül is elérhetővé kívánjuk tenni, kell TCP/IP, és egy IP cím. Ha a felhasználók kegyeiben szeretnénk járni, a gépnek NetBIOS nevet is adunk. Erre az alapra azután jöhetnek a további szolgáltatások.

Ha az eddig felsorolt három erőforrást *(lemez, IP cím, NetBIOS név)* egyszerre, egy időben indítjuk, a NetBIOS név nem születik meg, mert IP cím híján a névnek sincs értelme – azaz szolgáltatásaink között függőségi viszony fedezhető fel, ami az indítási sorrendjüket is meghatározza: az IP címtől függ a név. Érdekes, hogy míg virtuális kiszolgálóinkat szabadon tákolhatjuk össze *(maximum nem indul el)*, addig fizikai, telepített operációs rendszereinkbe gyárilag belekerül a szolgáltatások helyes indítási sorrendje. A gyakran látott „at least one..." hiba pedig valóban azt jelenti, amit: legalább egy szolgáltatás nem indult el. De hogy pontosan hány, azt a függőségi útvonalak mentén oszladozó szolgáltatástestemek összeszámlálásával tudhatjuk meg.

A fenti ábra egy olyan - Exchange 2000 Servert futtató – tagkiszolgálóról készült, amelyik áramszünet után egyszerre indult a nála lassabb tartományvezérlővel, így az Exchange indítása időben korábbra esett, minthogy a tartományvezérlő önmagára talált volna. Ebből következően a tartományi fiókkal futó Exchange szolgáltatások nem tudtak bejelentkezni. Ebben az esetben az „at least one.." valójában négy, illetve a mellékzöngékkel együtt inkább nyolc:



Ez a rendszerindítás nem valami fényesen sikerült! A tartományvezérlő hiánya miatt Windows 2000 helyett egy félkarú rabló került a kezünk közé.

Alulról felfelé a következő hibák olvashatók az eseménynaplóban:

- A NetLogon szolgáltatás nem talált tartományvezérlőt
- A W32Time nem talált forrást *(tartományvezérlőt)* a rendszeróra szinkronizációjához
- A Service Control Manager nem tudta elindítani az Exchange szolgáltatásait *(System Attendant, Information Store, MTA, POP3, IMAP4)*
- A W3SVC *(Internet Information Server)* nem tudta felvenni az Outlook Web Access könyvtárakat *(Exadmin, Exchange, Public)* az M: meghajtóról

Más hiba nincs. Szerencsére ebben az esetben a „hiba" kijavítása nem áll másból, mint a kiszolgáló újraindításából: mostanra elindult a DC, és mind a tizenvalahány üzenet megszűnik.

A Service Control Manager

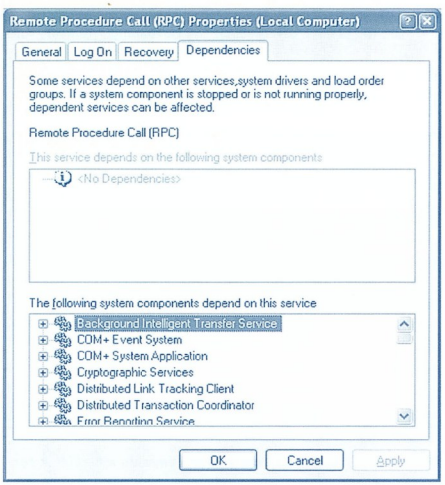
A Windows rendszerindítását a bootolás korai szakaszában az NTLDR, később az NTOSKRNL, majd a Service Control Manager végzi. Ez utóbbi felelős a szolgáltatások megfelelő sorrendű indításáért – mellesleg a SYSTEM32 könyvtárban, a SERVICES.EXE fájlban lakik.

A szolgáltatások indítási sorrendjét a regisztrációs adatbázis tartalmazza – minden szolgáltatás tudja magáról, hogy ő ki/mi után következik.

Vessünk egy pillantást a függőségi viszonyokra, indítsuk el az Administrative Tools/Services eszközt, s jobb gombbal csiklandozzuk meg mondjuk az RPC szolgáltatást, majd kattintsunk a Dependencies *(függőségek)* fülre. A következő ábra alsó részében megfigyelhetjük,hogy mely szolgáltatások indulása hiúsul meg, ha az RPC befuccsol. A Windows Installertől az RRAS-on át a Protected

Storage-ig minden megtalálható itt, kivéve a Server és a Workstation. *(Erről jut eszembe, ha minden igaz, lesz egy bátor jelentkezőnk arra a feladatra, hogy a Windows szolgáltatásait egytől egyig elmagyarázza nekünk, hogy tudjuk, vajon mire való - például - a Distributed Link Tracking, és mit veszítünk/nyerünk azzal, ha leállítjuk.)*

Visszatérve: ez azt jelenti, hogy ha az RPC nem indul el, akkor *(at least...)* mintegy harminc további komponens sem!



Az RPC-től a következő szolgáltatások függnek: szinte mind!

Még szerencse, hogy a párbeszédpanel felső listájának tanúsága szerint az RPC semmitől sem függ, tehát garantáltan mindig elindul. Távolról azonban már nem biztos, hogy hozzá tudunk csatlakozni, ugyanis az RPC kényes a hálózat helyes működésére. Ha az alábbi hibaüzenetek valamelyikével találkozunk, elsősorban hálózati hibára gyanakodjunk!

```
The RPC server is too busy to complete this
operation.
```

és

```
The RPC server is unavailable.
```

Nézzünk egy megtörtént esetet, mi állhat a hibaüzenet hátterében:

1. hálózati hiba miatt meghiúsul az időszinkron a tartományvezérlők között
2. ha az órák 5 percnél jobban szétcsúsznak, a Kerberos beint nekünk
3. ha nem megy a Kerberos, a legkülönbözőbb jelenségek ütik fel fejüket, többek között a rendszergazda...
4. „The RPC server is unavailable" üzeneteket kap, valahányszor rendszergazdai eszközt indít
5. ez odáig fajulhat, hogy leáll az Active Directory replikáció, hisz az *(jobb híján)* RPC alapú

A Service Control Manager működése módosítható, befolyásolható, ha tudjuk, hol tárolódnak a rendszerben a függőséget leíró adatok. Nem máshol, mint a regisztrációs adatbázisban. Mielőtt azonban nekiesnénk, keressünk egy jó példát, vajon mi a csudáért lenne érdemes beavatkozni a szolgáltatások egymásra épülésébe. Nekem van egy ötletem! *(Remélem Redmond is olvassa ezt a cikket, és az ötletből beépített szolgáltatás válik...)*

Last Known Good és System Restore - röviden

Bizonyára sokan ismerik a Windows NT óta meglévő remek szolgáltatást, mellyel szétkonfigurált rendszereinket vissza lehet vinni egy korábbi, talán még működőképes állapotba: ez a Last Known Good indítás, mely ha mást nem is, de a regisztrációs adatbázis CurrentControlSet részét könnyen helyreállíthatóvá teszi. Ha szétparamétereztünk valamit, ettől visszaparamétereződik. *(Emlékeztetőül: bootoláskor szóközt, majd „L" betűt ütünk.)* A Windows XP emellett tartalmaz egy mindent-mentő, úgynevezett System Restore visszaállítási lehetőséget is, mely a Windows 2000-féle Sytem File Checker *(~rendszerfájl-visszamásoló)* szolgáltatás kibővítése tetszőleges visszaállítási pontokkal és csinos varázslófelülettel. *(XP kezdőknek: Accessories->System Tools->System Restore)*. Nomármost ez mind szép és jó, de nekem kevés. A Last Known Good *(na jó, és a Rollback Driver)* tökéletesen megfelelő lenne számomra, ha nem úgy működne, ahogy. Évszázadok óta az a baja, hogy ostobaság rám bízni annak megállapítását, vajon egy adott rendszerindítás hibátlan volt-e, ha egyszer a Service Control Manager pontosan tudja, és jelzi az indítási hibákat *(at least one... ugyebár)*. 1994, azaz Windows NT 3.1 óta azonban változatlanul ugyanúgy dől el, hogy egy beállításturmix helyes-e: vajon sikeresen be tud-e valaki jelentkezni? A hibátlan bejelentkezés pecsételi meg az adott konfiguráció sorsát: tekintsük jónak! A bejelentkezésünk a háttérben egyben minőségellenőrzés is: „Passed" pecsétet kap az adott CurrentControlSet. Könyörgöm, ekkor még **ne** tekintsük jónak! Hisz a sikeres bejelentkezés után még javában indulnak a szolgáltatások, az SQL Server egy, az Exchange három perccel a bejelentkezés után végez, amikorra mi már nemcsak hogy sikeresen bejelentkeztünk, hanem már egy parti WinMine-t is elveszítettünk.

Kedves Bill!

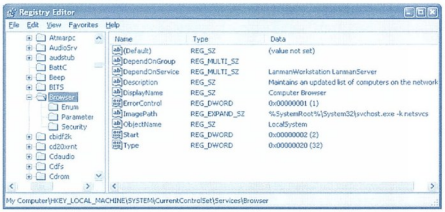
Javaslatom a következő: legyen a Windowsban egy olyan szolgáltatás, ami a többi után, azoktól függően indulna el, és ez állítsa át a Last Known Good jelzést a beállításokon. Ezzel a megoldással ugyanis egy csapásra megoldódik a meo-dilemma. Ha „at least one service failed to start", akkor ez utóbbi sem indul el, hisz függ a többitől. S ha nem indul el, nem is pecsételi le feleslegesen (sőt hibásan) a CurrentControlSetet.

Üdvözlettel
Fóti Marcell

Ezek után pillantsunk bele a regisztrációs adatbázisba.

A szolgáltatások beállításai

Szemeljük ki áldozatként a jó öreg Browse szolgáltatást. Tudják, ez tehető felelőssé a Network Neighborhoodban látottakért. Az ő beállításai így néznek ki:

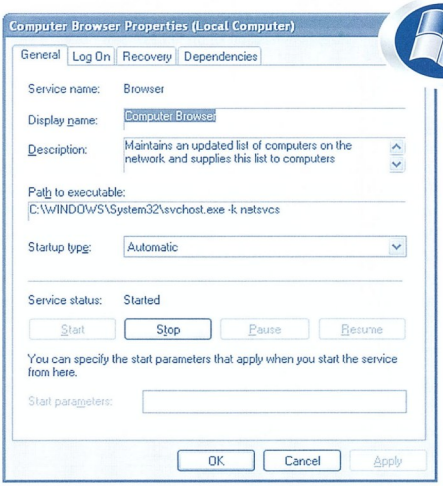


Egy kiragadott Windows szolgáltatás beállításai

A fenti ábráról minket a DependOnGroup és DependOnService sorok érdekelnek. Ez írja le, hogy az adott szolgáltatás milyen egyéb komponensektől függ. A Browse indulása például mind a LanmanWorkstation, mind pedig a LanmanServer futásától függ. *(Az is megérne egy cikket, hogy mitől Lanman, mikor lassacskán a NetBIOS-tól tökéletesen megszabadulunk)*. Ilyen egyszerű. Ha Bill megfogadja a tanácsomat, akkor a Windows .NET 2005-ben találunk majd egy LastKnownGood nevű szolgáltatást *(nomeg egy varázslót, hogy ne regedittel kelljen beállítani a működését)*. Az egyes szolgáltatások beállításai alkönyvtárakban *(alkulcsokban)* találhatók. A Browser például a Parameters kulcs alól olvassa ki, hogy egyáltalán szükség van-e rá *(MaintainServerList érték)*, a LanmanServer pedig saját Shares kulcsába dugja el a megosztott könyvtárak adatait *(könyvtárnév, megosztásjogok stb)*.
A Knowledge Base 62 találatot jelez, ha rákeresünk a DependOnService szóra. Mit érdemes átállítani? A Browser szolgáltatás függőségét nem, jottányit sem javul tőle, azonban meglepő módon a Netlogon bizonyos esetekben módosítandó! Ha egy tartományvezérlő egyben saját DNS Servere is, akkor a szolgáltatásokat érdemes úgy felfűzni, hogy a Netlogon akkor induljon, amikor a DNS már garantáltan fut. Ehhez a Netlogon szolgáltatás DependOnService értékébe a meglévő LanmanWorkstationon kívül fel kell venni a DNS szolgáltatást is.

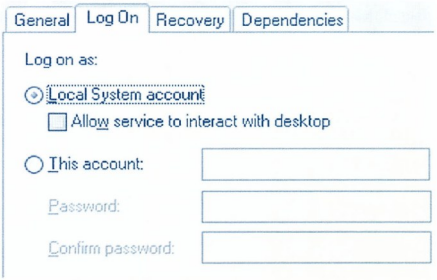
A Services eszköz

Maradva a Browse szolgáltatásnál, nézzük meg, mit nyújt az Administrative Tools->Services. Először is meg kell találnunk benne a Browsert, ami csak másodikra sikerül, mert itt Computer Browser a neve. *(A fenti registry-ábrán ez a DisplayName sorban olvasható.)* Hatalmas újdonság, hogy nemcsak egyesével lehet újraindítani a szolgáltatásokat, hanem - a függőségek figyelembe vételevel – csoportosan is. Erre való a jobbklatty, Restart opció. De most lássuk a Browser tulajdonságlapját!



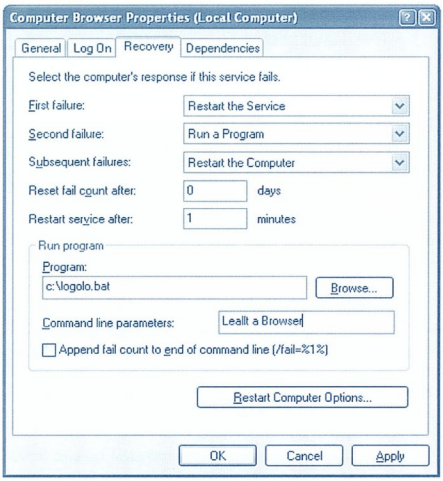
Szolgáltatásaink beállításai egytől egyig a regisztrációs adatbázisban vannak. Amit ezen a panelen nem lehet átírni – azt átvéshetjük Regedittel!

A legalsó mezőbe olyan indítási paraméterek írhatók, amiket az adott program parancssorból elfogadna. Két élő példa: az SQL Server különböző üzemmódjait *(factory default, minimal stb.)* valamint az Exchange Key Management Server indítási kulcsát „szokás" ide beírni. Ez az érték sajnos nem kerül ki automatikusan a registrybe. Ha tartósan használni szeretnénk, írjuk az ImagePath végére!
A következő fülön a szolgáltatás bejelentkezési neve és jelszava állítható be *(melyet a Service Control Manager felír egy cetlire, hogy mindig kéznél legyen neki, valahányszor indítania kell az adott programot)*. Két beállítási lehetőség van:



A szolgáltatás is „ember", ezért kell neki név és jelszó!

A Local System jelentése: a szolgáltatás a Windows része, nem használ külön felhasználói fiókot. A Windows korábbi változataiban *(NT4 és elődei)* a Local System egyet jelentett azzal, hogy az adott szolgáltatás nevesített külső kapcsolatot nem kezdeményezhetett *(a tévhit ellenére anonymoust viszont igen)*. Ma már ez a korlátozás is a múlté, ugyanis a Windows 2000 és utódai ilyenkor a Computer Accounttal próbálkoznak a hálózaton *(emlékezzünk: MASINA$)*. Gyakorlati példával: az MSSQLServer esetében csak akkor kell átállítani a futtató felhasználót, ha beindítjuk az SQL Mail szolgáltatást, mert ebben az esetben hozzá kell férnie egy MAPI postafiókhoz, ami MASINA$-nak általában nem jár. És végül a Recovery fül:



Ha a Computer Browser leáll, próbáljuk újraindítani. Ha ismét leáll, induljon el egy batchfájl, harmadszorra pedig induljon újra a gép

Kritikus szolgáltatások esetén *(például DNS)* megfontolandó a Windows beépített helyrehozási lehetőségének beállítása. Minden szolgáltatásnak három élete van. Első meghalása után – például – megpróbálhatjuk automatikusan újraindítani. Ha *(a fenti beállítások esetén)* 24 órán belül megint elpukkan, futtathatunk egy programot. Vigyázat, a Service Control Manager által indított programok felhasználói felületet nem kapnak! Komolyabb esetben megfontolandó egy VBScript futtatása, mely parancssorból átveszi a hiba körülményeit, és akár emailben, akár net send paranccsal értesít valakit. Ha a második életét is elpazarolta a szolgáltatás, akkor jöhet a végső megoldás: a gép újraindítása.

A beállítások tesztelésére a Support Tools eszközkészlet KILL.EXE segédparancsát javaslom, bár pont a Browsert nehéz így leállítani, hiszen a SvcHost.EXE része. SQL Server, Lotus Notes és hasonló, utólag telepített alkalmazások azonban szabadon megdönthetők a segítségével. Így megismerhetjük a rendszer reakcióidejét *(1 másodpercen belül lefut a Recovery fülön beállított cselekmény)*, valamint a tesztelt szolgáltatás túlélési képességét is. A KILL egy csodafegyver, használják bátran! Előtte még véletlenül se mentsék adataikat!

Fóti Marcell
marcellf@netacademia.net

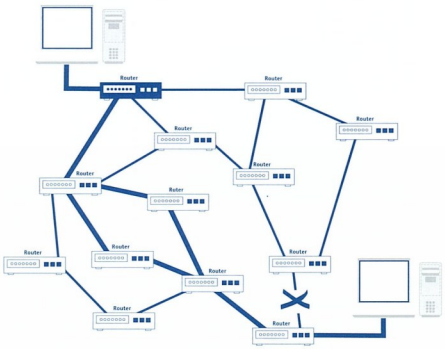
RRAS:
Open Shortest Path First

RRAS sorozatunkban utoljára a RIP útválasztási protokollal foglalkoztunk, és megállapítottuk hiányosságait: lassú konvergenciáját, hurokérzékenységét, 15 hopos korlátját – egyszóval butaságát. A RIP ismert gyengeségeit nem a következő verzió kivárásával kell orvosolni, hanem – ha hálózatunk méretével, bonyolultságával a RIP nem tud megbirkózni – áttérhetünk egy sokkal fejlettebb útválasztási protokoll, az OSPF használatára

Az OSPF legnagyobb újdonsága, hogy – ellentétben a RIP-pal –útválasztóink nem komplett útvonaltáblákat küldenek át a szomszédos routerre, hanem csak egy icipici táblázatot, melyben az általuk ismert hálózati útvonalak adatai szerepelnek: irány, ár, és állapot.

Open Shortest Path First

Az **[1]** címen található RFC tanúsága szerint az OSPF útvonalválasztók valójában útvonal **tervezők**, amit úgy kell értenünk, hogy minden egyes router önmagában, intelligensen számítja ki egy-egy IP csomag leszállításának optimális útvonalát. A protokoll elnevezésének tükörfordításával: a legrövidebb útvonalat nyitja meg az áthaladó csomagok előtt. Az OSPF bonyolult hálózatok lekezelésére is képes. Nemhogy nem borul ki hurkokkal terhelt hálózatokban, hanem ott érzi igazán elemében magát. Vessünk egy pillantást az alábbi ábrára:



Hurkokkal tarkított vállalati hálózat

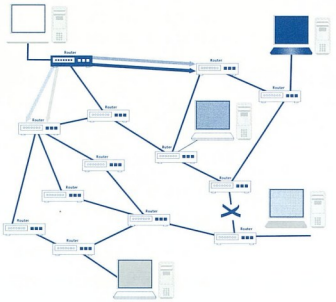
Ilyen hálózatban a RIP valószínűleg pillanatok alatt végezne önmagával, pedig ez a hálózat megbízhatóbb csomagtovábbítást nyújt egyszerűbb társainál, mert ha egy útvonal kiesik, szinte biztosan akad más út, melyen a két, sarokba állított számítógép kommunikálni tud egymással. Ehhez természetesen az kell, hogy vonalszakadás esetén hamar átszámítódjanak a közbenső routerek útvonaltáblái.

A bal felső gép szeretné megpingelni a jobb alsót. A hálózat jobb alsó részében a kereszttel jelölt helyen szakadás van. Mi egyetlen szempillantással fel tudjuk mérni, hogy a fekete útválasztónak a három lehetséges hálózati útvonal közül csak a vastag vonallal jelzett úton szabad elindítania az alsó gép felé a csomagot: a többi út 3-4 router múlva „eldugul". Igen ám, de az útválasztók nem felülről szemlélik a hálózatot, hanem abban benne állnak. Madártávlatból könnyű megtalálni az Eiffel toronyhoz vezető utat, de Párizs szűk utcáinak egyikén toporogva a feladat korántsem ilyen egyszerű. Mi kell a célpont megtalálásához? Térkép!

A Link State Table

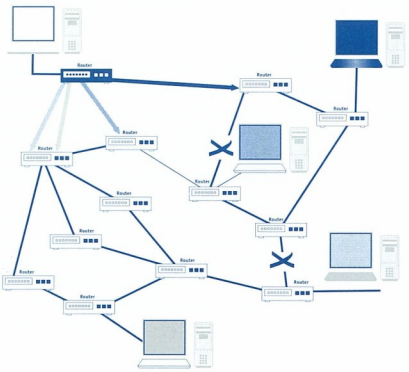
Térkép sajnos nincs. Felesleges lenne térképet készíteni olyan városban, ahol egyik pillanatról a másikra új utcák születnek, más utcákat eltorlaszolnak, megváltozik az utca neve, vagy a végpontja. Ehelyett van egy csapatnyi turistánk, akik a város kereszteződéseiben állva figyelik az általuk látható utakat, és – ha változás áll be – közlik közvetlen szomszédaikkal a hírt: teljes szélességében felbontották a Rákóczi utat, vagy fizetőssé vált a Lánchíd használata. Az egyes útválasztók az általuk kezelt hálózatokról, a vonalak állapotáról, áráról stb. táblázatot vezetnek, melynek neve Link State Table. Ezt a - lokális helyzetképet tükröző – táblázatocskát küldik át szomszédaiknak, akik természetesen továbbítják a távolabbi útválasztóknak is. Idővel mindegyik útválasztóhoz eljut a többiek által közölt állapottáblázat. És ekkor kezdődik a valódi munka: a beérkezett jelentések alapján minden egyes router elkészíti **saját** térképváltozatát, melyben nem fognak szerepelni sem a lehetetlen, sem a drága útvonalak. Az előző ábrán a fekete vonal valójában nem más, mint a fekete útválasztó által generált térkép. Egy-egy útválasztó csak a saját hatáskörébe tartozó útvonalak használatáról dönt, de a döntést a többiek által beküldött Link State Table alapján teszi!

Közelítsük az ábrát egy kicsit jobban a valósághoz! Színes, iMAC-like gépeket vásárolunk, hupikéket, ezüstszürkét és bíbort *(bár ez utóbbi szín a lapban használt két festék tetszőleges arányú kevergetésével sem lesz más, mint szürke)*. A bal felső munkaállomás változatlan marad, színpompás gépeinket viszont a hálózat különböző pontjaira helyezzük. A fekete útválasztót megvizsgálva kiderül, hogy a beérkezett Link State Tablek *(LST)* alapján mindegyik célpont felé tud térképet rajzolni.



Az OSPF útválasztó útvonaltáblája különböző célcímeket tartalmaz

Ha megszakítjuk az egyik vonalat, a távoli útválasztók már küldik is vadonatúj Link State táblájukat, és fekete routerünk hamarosan új térképet készít.



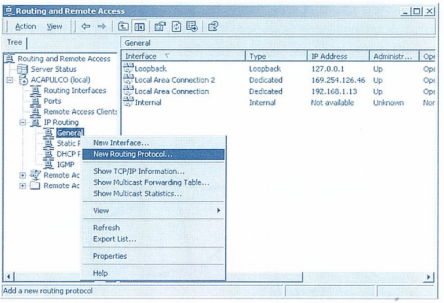
Az OSPF útválasztók gyorsan alkalmazkodnak a megváltozott hálózati felépítéshez

OSPF fogalommagyarázat

Ennyi „matematika" talán elég is lesz *(Dijkstra bácsi forog a sírjában)*, mivel nem tartom valószínűnek, hogy közülünk bárki is belefogna OSPF implementáció készítésébe. Azonban korántsem vagyunk készen. Hátra van még az OSPF környezet ismertetése, valamint az RRAS megfelelő beállítása.

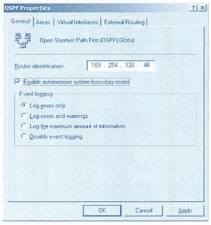
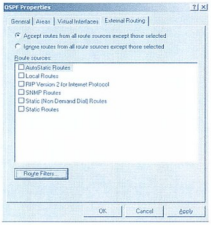
- **Autonomous System *(AS)***. Az egymással Link State Tablet cserélő OSPF útválasztók egy közös Autonomous System részei, közös autentikációval, közös állapotinformációval rendelkeznek. Egy hálózaton elvileg lehet egynél több AS is, ennek azonban nem sok értelmét látom.
- **Area**. Nagy hálózatokban érdemes lehet bizonyos zárt leágazások útválasztóit csoportokba foglalni. A csoporton belül szabadon áramlik a Link State, kifelé viszont a csoport tagjai egyetlen útválasztónak fognak látszani. Az area nagyon hasonló az Active Directory-féle telephelyekhez: kis zárt világ, mely a külvilággal általunk definiált módon kommunikál. Ha a Microsoft nevezte volna el, ma azt mondanánk rá: site. Az areakat IP cím szerű, 4 tagból álló számokkal azonosítjuk, melyeknek valójában semmi közük semmihez: a rendszergazda hasraütéses módszerrel választhat area azonosítókat. Ha nem definiálunk külön területeket, akkor útválasztóink az alapértelmezett, 0.0.0.0 azonosítót viselő area tagjai lesznek. Egészen pontosan nem az útválasztók tagjai egy-egy areanak, hanem az általuk kezelt vonalak. Ezzel elérhető, hogy ha egy router mondjuk három különböző areaba eső hálózatot kezel, akkor tulajdonképpen mindháromnak része.
- **Range**. Egyetlen area egynél több IP tartományt lefedhet. Ezt az areahoz rendelhető IP alhálózatokkal adhatjuk meg.
- **Stub**. Stub, vagy végállomás areanak azokat a területeket nevezzük, melyek útválasztói zárt közösséget alkotnak, külső útvonalakat nem tanulnak meg.
- **Boundary router**. Az előző ellentéte. A Boundary, vagy határútválasztó annyi útinformációt gyűjt, amennyit csak tud: statikus bejegyzéseket, RIP-et, SNMP-től tanul stb.
- **Hello protokoll**. OSPF útválasztóink automatikusan keresik meg társaikat. Címzésmódját tekintve Multicast, azaz a periodikusan kibocsátott hellózás nem zavarja a hálózat összes gépét. Alapértelmezésben tíz másodpercenként történik. Szintén alapértelmezés, hogy ha egy útválasztó négyszer egymás után nem válaszol a Hellora, akkor hibásnak kell tekinteni.

Lássuk ezek után, hogyan kell összehozni egy OSPF megoldást a Windows 2000 beépített RRAS szolgáltatásával! Először is telepítenünk kell az OSPF protokollt, ami a szokásos módon, az IP Routing alatti General fülön jobbklattyintva, a New Routing Protocol-lal végezhető el:



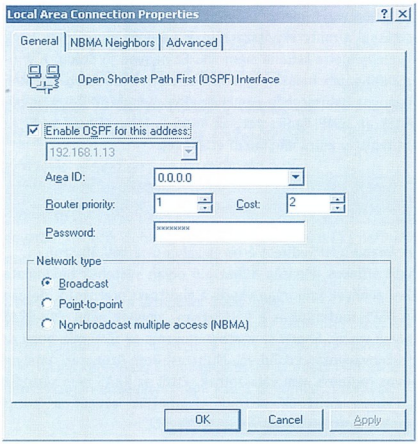
Az OSPF telepítése

Telepítés után azonnal kezelhetjük az areakat, átállíthatjuk útválasztónkat Boundary routernek, és kiválaszthatjuk, hogy a határvidéken túlról milyen útinformációt gyűjtsön be. Ha az alábbi ábraparon az OSPF protokoll tulajdonságlapjának első fülén beállítjuk, hogy határvidék-útválasztó legyen, a negyedik fülön kiválaszthatjuk, hogy milyen forrásokból tanuljon:

Határvidék routereken megválaszthatjuk, mit emeljenek be a mi hálózatunkra a külvilágból

Az útválasztó protokoll mindaddig nem indul be, amíg meg nem mondjuk neki, hogy mely kártyákon Hellózhat. Ezt az OSPF-en jobbkattintva tehetjük meg *(New Interface...)*.
Vegyünk fel tehát egy hálózati kártyát, hogy azon át OSPF routereink megkezdhessék aktív hellózásukat:



Hálózati kártya felvétele az OSPF hatásköre alá. Ezen a kártyán keresztül tanulni, és tanítani fog

Hozzá kell rendelnünk egy előre létrehozott areahoz, be kell állítani, hogy mekkora eséllyel legyen ő a fő-fő Hellózó *(designated router priority)*, valamint hogy a tőle kimenő útinformációkon mennyivel növelje meg a metric *(cost)* értéket. A jelszóra még visszatérünk, a hálózat típusa ethernet esetén nyilván broadcast. Amint ezen a kártyán elindul a Hello, a rutinos rendszergazda nyomban Network Monitort ragad, és megnézi, hogyan is fest az üzenet a hálózaton. Sorokra tördelve és megritkítva nézzünk meg egy Hello csomagot!

```
Hello ACAPULCO 224.0.0.5 IP
```

Az ACAPULCO nevű gép küldte ezt a csomagot a 224.0.0.5-ös Multicast (OSPF) címre. Ethernetszinten a célcím így fest:

```
ETHERNET: Destination address : 01005E000005
ETHERNET: .......1 = Group address
ETHERNET: ......0. = Universally administered
↳ address
```

Tehát csoportcím, mégpedig a szabványban rögzítettek egyike *(universally administered)*. IP szinten az érdekesebb adatok:

```
IP: Time to Live = 1 (0x1)
```

Látszik, nem arra tervezték, hogy több hopon átmenjen! Ez egy router-router csomag.

```
IP: Protocol = 0pen Shortest Path First IGP
↳ (0x059)
```

IP-be ágyazva OSPF utazik. *(Mi másra számítottunk?)* És most jön maga az OSPF rész:

```
OSPF: Message = Hello
    OSPF: Version = 2 (0x2)
    OSPF: OSPF Packet Type = Hello
    OSPF: Packet Length = 52 (0x34)
    OSPF: Source Router ID = 169.254.126.46
    OSPF: Area ID = 0.0.0.0
    OSPF: Authentication Type = Simple Password
    OSPF: Authentication = 0x3837363534333231
```

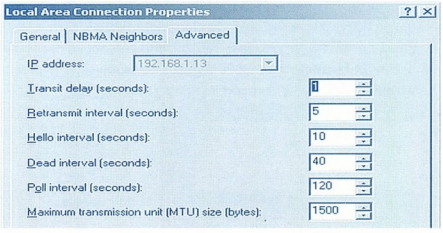
Itt álljunk meg egy pillanatra. Vissza tudjuk fejteni a jelszót *(authentication)*? Még nem?

```
    OSPF: Netmask = 255.255.255.0
    OSPF: Hello Interval = 10 (0xA) seconds
    OSPF: Router Priority = 1 (0x1)
    OSPF: Dead Interval = 40 (0x28) seconds
    OSPF: Designated Router = 192.168.1.13
    OSPF: Backup Designated Router = 192.168.1.14
    OSPF: Neighbor = 10.10.10.10
    OSPF: Neighbor = 30.30.30.30
```

A Neighbor mezőkben olvashatjuk, hogy ez az útválasztó milyen egyéb útválasztókat talált ezen hálózati szegmensen. A hexa panelről ennyit érdemes megfigyelni:

```
12345678
```

Ez bizony a jelszó. Enyhén clear text. Adapterszinten érdemes még megtekinteni a különböző időzítési paramétereket:



Az OSPF időzítésének alapértelmezett értékei. Hello 10 másodpercenként stb.

Ezekről annyit érdemes tudni, hogy ha bármelyik útválasztón átállítjuk, akkor a többin is után kell húzni!
Miután a kártyákat felvettük, már zajlik a Link State Table cse-

rebere. Le is ellenőrizhetjük az RRAS konzollal, vajon látják-e egymást útválasztóink. Az OSPF protokoll jobbgombos menüjében ugyanis a következő értékesebb lehetőségek rejlenek:

- Show Link State Database. Itt megtekinthetjük, mit is tanult útválasztónk a többiektől. Vigyázat! Itt nem konkrét útvonalbejegyzéseket látunk, hanem csak egy tájékoztató táblázatot! Ennek minden gépen közösnek kell lennie – de az ez alapján generált útvonaltáblák már egyediek! *(Lásd később.)*

ACAPULCO - OSPF Link State Database

Area ID	Type	Link state ID	Advertising rou...	Age	Sequence
0.0.0.0	Stub	10.0.0.0	10.10.10.10	431	0x80000000
0.0.0.0	Router	10.10.10.10	10.10.10.10	772	0x80000005
0.0.0.0	Router	30.30.30.30	30.30.30.30	881	0x80000006
0.0.0.0	Router	169.254.126.46	169.254.126.46	431	0x80000008
0.0.0.0	Router	169.254.177.213	169.254.177.213	3.4...	0x80000002
0.0.0.0	Network	192.168.1.13	169.254.126.46	431	0x8000000C
0.0.0.0	AS External	30.0.0.0	30.30.30.30	750	0x80000002
0.0.0.0	AS External	45.45.45.0	30.30.30.30	750	0x80000002

A Link State Table közös minden, azonos areaba tartozó routeren. Ez a recept. Ebből főzik az egyedi útvonaltáblákat

- Show Neighbors. Itt tekinthetjük meg, hogy az OSPF Multicast címre kibocsátott intenzív hellózás milyen eredményre vezetett. Találkoztak-e útválasztóink?

ACAPULCO - OSPF Neighbors

Neighbor	Neighbor ID	Type	State	Priority	State-changes ...	Retransmit-qu
192.168.1.14	30.30.30.30	Dynamic	Full	1	0	0
192.168.1.1	10.10.10.10	Dynamic	Full	1	0	0

ACAPULCO szomszédai

Nem tudom, megfigyelték-e, de ezidáig az OSPF telepítése is éppolyan könnyedén zajlott, mint a RIP-é. Egyszerű hálózatban valóban ennyi, de oda a RIP is elég. Ugyanakkor az eddig említett eszközkészlettel *(NetMon, Show Neighbors, Show Link State Database)* felfegyverkezve összetettebb hálózatokon sem kell tartanunk attól, hogy az OSPF kicsúszik az ellenőrzésünk alól. Sőt! Telepítsd, és felejtsd el!

A kész útvonaltábla

Miután gépeink közös nevezőre kerültek a Link State Table kapcsán, akkor jön a munka oroszlánrésze: egyedi, az ő nézőpontjukból optimális útvonaltáblát kell generálniuk. Ebbe a folyamatba se beavatkozni, se bekukkantani nem tudunk, csak a végeredmény a mienk. A szokásos route print paranccsal megvizsgálhatjuk, mi született az OSPF áldásos tevékenysége nyomán. Nem valami látványos, de azért megmutatom:

```
C:\>route print
===========================================================================
Interface List
0x1 ........................... MS TCP Loopback interface
0x1000003 ...00 30 1b 0c 0e 5a ...... NDIS 5.0 driver
0x1000004 ...00 40 f4 2d f7 08 ...... Realtek RTL8029(AS) Ethernet Adapt
===========================================================================
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          10.0.0.0        255.0.0.0      192.168.1.1    192.168.1.13     4
          30.0.0.0        255.0.0.0      192.168.1.14   192.168.1.13     1
         45.45.45.0  255.255.255.0      192.168.1.14   192.168.1.13     3
        127.0.0.0        255.0.0.0        127.0.0.1      127.0.0.1     1
      169.254.0.0      255.255.0.0  169.254.126.46 169.254.126.46     1
  169.254.126.46  255.255.255.255        127.0.0.1      127.0.0.1     1
  169.254.255.255  255.255.255.255  169.254.126.46 169.254.126.46     1
      192.168.1.0    255.255.255.0      192.168.1.13   192.168.1.13     1
     192.168.1.13  255.255.255.255        127.0.0.1      127.0.0.1     1
    192.168.1.255  255.255.255.255      192.168.1.13   192.168.1.13     1
        224.0.0.0        224.0.0.0  169.254.126.46 169.254.126.46     1
        224.0.0.0        224.0.0.0      192.168.1.13   192.168.1.13     1
  255.255.255.255  255.255.255.255  169.254.126.46 169.254.126.46     1
Persistent Routes:
  None
C:\>_
```

A route print kimenete. Az utolsó, a Metric oszlop alapján gyaníthatjuk, hogy melyek a tanult és dinamikusan generált bejegyzések

Gondok?

Hogy a problémákról is szó essen: az ICMP Redirect csomagot arra találták ki, hogy rossz helyen kopogtató munkaállomásoknak a fejébe verje, merre van az arra. Dupla utas *(triangle)* útválasztásnál a routerek visszaszólnak a munkaállomásnak, hogy legközelebb merre kellene menniük. Ez nagyon jó dolog. A Windows 2000 gépek hálásan fogadják az ICMP Redirectet. Az RRAS is. Bizonyos hálózatokon nemkívánatos, hogy a routerek is befogadják az ICMP Redirectet: az RRAS is lebeszélhető erről az alábbi registry kulcs megfelelő értékével:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
  Services\Tcpip\Parameters
```

Itt az EnableICMPRedirect értékét value 0-ra kell állítani.

További érdekes jelenség állhat elő olyan switchek használata esetén, amelyek szigorúan veszik a mulitcast protokollt, de van olyan OSPF routerünk is a hálózatban, amelyik nem. *(Az RRAS komolyan veszi)*. Ebben az esetben ugyanis az történik, hogy az RRAS szabványosan kiküld egy Multicast Join üzenetet, amelyre bizonyos routerek nem válaszolnak. Ettől az RRAS nem sértődik meg, egyes switchek viszont elzárhatják ezeket a butus routereket a rendes, szabványos RRAS elől.

Fóti Marcell
marcellf@netacademia.net

A cikkben szereplő URL-ek:
[1] OSPF Version 2 http://www.ietf.org/rfc/rfc2328.txt

A Windows 2000 és a dinamikus DNS

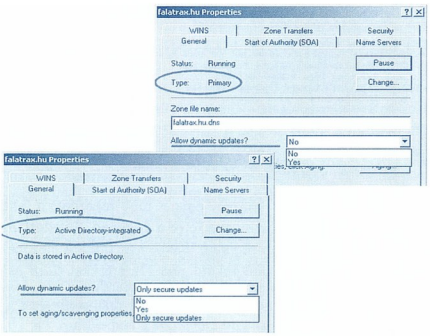
Mindenki tudja, hogy a Windows 2000 tartományok elsődleges névfeloldó szolgáltatásává *(a WINS-et és egyéb NetBIOS varázslásokat végre leváltva)* a DNS lépett elő. A Windows 2000 munkaállomások a DNS segítségével lépnek be a tartományba, jelentkeznek be a hálózatba, veszik fel egymással a kapcsolatot.

A tartomány DNS zónája ennek megfelelően viszonylag összetett és bonyolult, ráadásul a tartalma – főleg dinamikus IP címmel ellátott munkaállomások használata esetén – gyakran frissül. Az „ős" DNS-implementációk adatbázisát csakis a zónáért felelős rendszergazda módosíthatta, frissíthette igény szerint. Egy napi használatban lévő Windows 2000 tartomány DNS zónájának karbantartása azonban automatizálásért kiált. A Windows 2000 munkaállomások és kiszolgálók egyaránt ismerik és használják az RFC 2136-ban **[1]** definiált dinamikus DNS *(DDNS)* protokollt.

az „Only secure updates" (azaz: „csak biztonságos frissítések") lehetőség csak az Active Directoryba integrált zónák esetén jelenik meg

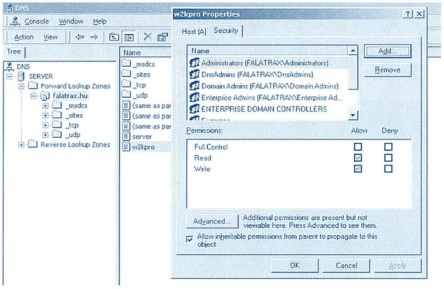
DDNS kiszolgáló a Windows 2000-ben

Kezdjük a kiszolgálóoldallal: a Windows 2000 DNS szolgáltatása képes fogadni és feldolgozni a DDNS ügyfelektől érkező névregisztrációs kéréseket. Ehhez mindössze az adott DNS zóna tulajdonságlapján engedélyeznünk kell a dinamikus frissítést.

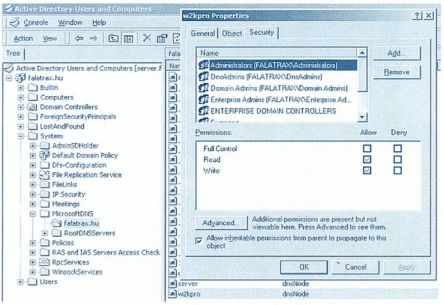


A DNS zónák dinamikus frissítésének engedélyezése. A biztonságos frissítés csak AD-integrált zóna esetén választható ki

A kiszolgáló felkészítése a dinamikus üzemmódra mindössze ennyiben merül ki. Ami feltűnhet, hogy az „Only secure updates" *(azaz: „csak biztonságos frissítések")* lehetőség csak az Active Directoryba integrált zónák esetén jelenik meg. Ennek az az oka, hogy a hagyományos, standard elsődleges *(Primary)* és másodlagos *(Secondary)* zónák „adatbázisai" egy-egy szövegfájlban tárolódnak, az AD-integrált zóna adatai pedig a címtárban található objektumok képében jelennek meg. Míg a címtárobjektumokra *(akár egyenként, azaz rekordonként is)* lehet hozzáférési jogosultságokat definiálni, egy szövegfájl soraira már nehézkesen. Ez látható abból is, hogy a nem AD-integrált zónák rekordjainak tulajdonságlapjáról hiányzik a „Security" oldal. A címtárintegrált zónánk objektumait egyébként megtaláljuk az Active Directory Users and Computers eszközben is, ha a View menüben kiválasztjuk az Advanced Features opciót.



Az AD-integrált zónában található rekordok hozzáférési jogosultságait definiáló oldal a DNS Manager-ben...



... illetve ugyanez az adat a rekordot megtestesítő dnsNode címtárobjektum tulajdonságlapján

A hozzáférési jogosultságok alapértelmezését különösebb ok nélkül nem érdemes módosítani, de ha szükség van rá, bátran megtehetjük. A biztonságos adatfrissítéshez használt DDNS protokollt és eljárást később ismertetjük.

Dinamikus DNS-frissítés

Térjünk át ügyféloldalra. *(Figyelem! DDNS regisztrációt nemcsak a munkaállomások, hanem a kiszolgálók is végeznek, ezért a „DDNS ügyfél" kifejezést a későbbiekben egyaránt érthetjük mind a Professional, mind a Server operációs rendszerekre.)*

A Windows 2000 munkaállomások és kiszolgálók címregisztrációját a DHCP Client Service *(DHCP ügyfél)* végzi – függetlenül attól, hogy statikus, vagy dinamikus *(DHCP kiszolgálótól kapott)* IP címet használ a számítógép!

A címregisztráció alapértelmezésben minden 24 órányi folyamatos működés után, illetve az alábbi események bekövetkeztekor zajlik le:

- Megváltozik a TCP/IP beállítás, a számítógép új IP címet kezd el használni, vagy megválik egy régitől
- Lejár a DHCP-től kapott IP cím bérleti ideje, illetve a munkaállomás frissíti a kapott címet
- Plug-and-Play esemény után

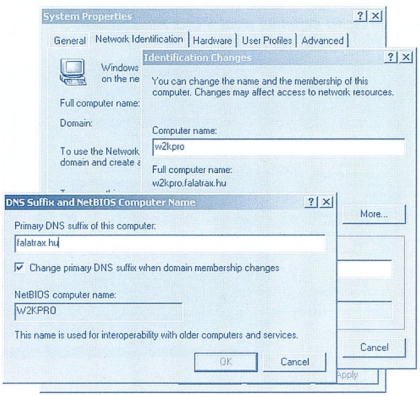
A regisztrációt két módon „kézzel" is kiválthatjuk. Az első az ipconfig parancs új kapcsolója:

```
ipconfig /registerdns
```

... a másik pedig a NetLogon szolgáltatás újraindítása *(az ugyanis minden indulásakor elvégzi a címek regisztrációját)*:

```
net stop netlogon
net start netlogon
```

A Windows 2000 nem csak akkor próbálkozik meg a regisztrációjával, amikor tartományba léptetjük. Ha a rendszer tulajdonságlapján a számítógép neve mellett kitöltjük az elsődleges DNS utótag értékét *(Primary DNS suffix)*, a regisztráció működni fog az így képzett számítógépnévvel, tartományon kívül is. *(Ha ez a mező nincs kitöltve, nincs mit bejegyezni – ergo a regisztráció is elmarad.)*



A számítógép teljes nevét a számítógépnév és az elsődleges DNS utótag összevonásával generáljuk – tartományon kívül is!

Példánkban a számítógép teljes neve tehát: „w2kpro.falatrax.hu". Az elsődleges utótag a gép tartományba léptetésekor megváltozhat.

A dinamikus regisztráció protokollja

Kicsit kénytelenek vagyunk elmélyedni a DDNS protokoll szabványába *(RFC 2136,* **[1]***)*. A **[2]** címről letölthető néhány, Network Monitorral elfogott regisztrációs forgalom. Az alábbiakban leírtak jól követhetők a nem tartományi tag Windows 2000 Professional által generált forgalmon, amit a pro_workgroup_start_dnsdyn.cap fájlban találunk. A DDNS frissítés igazodik a megszokott DNS protokollhoz, némileg kibővítve azt. Hálózati forgalom szempontjából elég azt tudnunk, hogy a DDNS forgalom az UDP 53-as, vagy szükség esetén a TCP 53-as porton zajlik. Maga a dinamikus DNS csomag két legfontosabb része az úgynevezett előfeltételek és a frissítendő adatok. A prerekvizíciós rész olyan előfeltételeket tartalmaz, amelyeknek teljesülniük kell ahhoz, hogy a kívánt rekord bejegyzésre kerülhessen. Ilyen feltétel lehet például, hogy egy adott nevű A vagy CNAME rekord már legyen a zónában, vagy pedig éppen ellenkezőleg: ne létezzen. A prerekvizítumok teljes listáját a **[3]** címen foglaltuk össze, de természetesen az RFC-ben is utána lehet nézni.

```
⊟DNS: Prerequisite: w2kpro.falatrax.hu. of type Canonical name on class
  ⊟DNS: Resource Record: w2kpro.falatrax.hu. of type Canonical name on cl
     DNS: Resource Name: w2kpro.falatrax.hu.
     DNS: Resource Type = Canonical name for alias
     DNS: Resource Class = 0x00FE
     DNS: Time To Live = 0 (0x0)
     DNS: Resource Data Length = 0 (0x0)
  ⊟DNS: Resource Record: w2kpro.falatrax.hu. of type Host Addr on class
     DNS: Resource Name: w2kpro.falatrax.hu.
     DNS: Resource Type = Host Address
     DNS: Resource Class = 0x00FE
     DNS: Time To Live = 0 (0x0)
     DNS: Resource Data Length = 0 (0x0)
```

Prerekvizítumok listája egy regisztrációhoz

A fenti példában például két feltételt látunk:

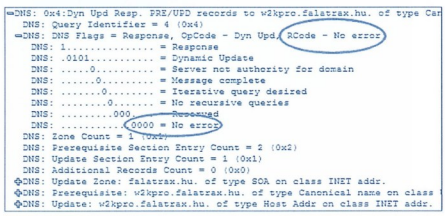
- w2kpro.falatrax.hu nevű, CNAME típusú rekord nem létezhet *(RRset Does Not Exist)*
- w2kpro.falatrax.hu nevű A típusú rekord nem létezhet *(RRset Does Not Exist)*

Azaz, a bejegyezni kívánt cím se alias, se közvetlen hivatkozás formájában ne szerepeljen a zónában – érthető feltétel egy új cím regisztrációjához. Ha a feltételek rendben vannak, a kiszolgáló bejegyzi a kérés Update mezőjében elküldött adatokat:

```
⊟DNS: Update: w2kpro.falatrax.hu. of type Host Addr on class INET addr.
   DNS: Resource Name: w2kpro.falatrax.hu.
   DNS: Resource Type = Host Address
   DNS: Resource Class = Internet address class
   DNS: Time To Live = 1200 (0x4B0)
   DNS: Resource Data Length = 4 (0x4)
   DNS: IP address = 192.168.77.111
```

Regisztrációs adatok az Update mezőben

Az Update mező is tartalmazhat egynél több rekordot; a fenti például egy új bejegyzést hozna létre a zónában. Emellett rekordok törlésére is lehetőség van: ilyenkor az Update mezőben megjelenik a törölni kívánt rekord, 0-s TTL értékkel *(lásd például az IP cím változtatáskor lezajlott forgalmat bemutató pro_domain_ipchange_dnsdyn.cap fájlt)*. Ha a feltételek nem teljesülnek, a regisztráció nem megy végbe, mi pedig a kiszolgáló által visszaküldött válaszból értesülünk a tényről. Az alábbi ábra egy sikeres műveletet mutat:



A bejegyzés sikerességéről a kiszolgáló válaszában az RCode mezőben tájékoztat minket.

A hibakódok teljes listája az RFC-ben található, de a Network Monitor is visszafejti nekünk, amit látni szeretnénk. Érdekesség, hogy „tapogatózási" célból olyan DDNS kérést is küldhetünk, ami Update mezőt nem, csak előfeltételeket tartalmaz. Ilyenkor a kiszolgáló válaszából lehet következtetni a zóna tartalmára *(vagy akár arra, hogy a kiszolgáló támogatja-e a DDNS kommunikációt)*. Ennyit az általános regisztrációs forgalomról, most pedig következzen a Windows 2000-féle, gyakorlati megvalósítás.

Dinamikus DNS bejegyzés a' la Windows 2000

Amikor a Windows 2000 dinamikus DNS bejegyzésre készül, az alábbiakat teszi *(ha van lehetőségünk, Network Monitorral követhetjük a leírtakat a pro_workgroup_start_dnsdyn.cap fájlban; a sorok elején látható számok a frame-ek sorszámát jelölik)*:

- #6: A Windows a beállított DNS kiszolgálótól lekérdezi a saját nevéhez *(w2kpro.falatrax.hu)* tartozó SOA rekordot *(ami egyben a szülőtartomány SOA rekordja is)*. Erre azért van szükség, mert a dinamikus regisztrációs kéréseket a zóna hiteles *(authoritative)* példányát tartalmazó kiszolgálóhoz kell küldeni.
- #7: a helyi DNS kiszolgálótól visszaérkező válaszban *(a SOA rekordban)* többek között megtalálható a zóna elsődleges névkiszolgálójának neve és IP címe egyaránt *(server.falatrax.hu, 192.168.77.2)*.
- #8: a kliens egy frissítés nélküli *(!)* DDNS kérést küld a kiszolgálónak, amiben az alábbi két feltétel szerepel:
  1. Ne létezzen w2kpro.falatrax.hu alias *(CNAME)* bejegyzés
  2. Létezzen w2kpro.falatrax.hu host *(A)* bejegyzés, mégpedig a 192.168.77.111 IP címmel *(ez a kliens jelenlegi IP címe)*

Gondoljunk csak ebbe bele! Ha létezik a számítógép nevével megegyező alias *(CNAME)* a zónában, az eleve kellemetlen *(akkor nem hozhatunk létre A rekordot a nevére, előtte a meglévő aliast törölni kell)*, tehát az első kérdés jogos. A második kérdés pedig arra ad választ, hogy a név, mint host *(A)* rekord regisztrálva van-e, és ha igen, akkor az IP címe megfelelő-e. Ha a kiszolgáló most pozitív választ ad, nincs szükség regisztrációra, hiszen pontosan a megfelelő adatok vannak a zónában. Ha a második feltétel nem megfelelő, akkor belekezdhetünk az adatok frissítésébe. Végül pedig, ha a kiszolgáló abszolút elutasító választ ad („not supported"), a zóna dinamikus frissítéséről is le kell mondanunk.

- #9: esetünkben a kiszolgáló hibaüzenete jelzi, hogy a bejegyzés még nem létezik, ezért következhet a dinamikus regisztráció
- #12: A következő DDNS kérés már előfeltételeket és Update adatokat is tartalmaz. A feltételek – az előző válaszhoz igazodva – az alábbiak:
  1. Ne létezzen a w2kpro.falatrax.hu alias *(CNAME)* rekord
  2. Ne létezzen a w2kpro.falatrax.hu host *(A)* rekord

Az Update szakaszban pedig megjelenik a bejegyzendő név és IP cím.

- #13: a kiszolgáló válaszában jelzi a sikeres regisztrációt

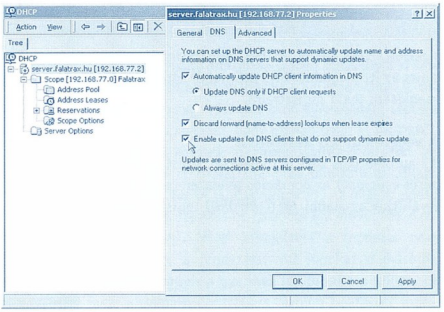
Ha az első kérdésre adott válaszban az derül ki, hogy a cím már létezik, csak helytelen IP címmel, akkor a frissítő csomagban a regisztráció mellett találnánk egy, a régi rekordot törlő bejegyzést is *(emlékszünk, 0-s TTL értékkel)*; valamint a prerekvizítumok is ennek megfelelően alakulnának. Az újonnan bejegyzésre kerülő rekordok TTL értéke *(mint láthatjuk)* 1200, azaz húsz perc. Ezt az alapértelmezést megváltoztathatjuk, ha a registryben módosítjuk az alábbi *(REG_DWORD)* értéket:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
Serices\Tcpip\Parameters\DefaultRegistrationTTL
```

A Windows 2000 ezután megpróbálkozik a reverse DNS cím *(PTR rekord)* bejegyzésével is, ami a mi minta-alhálózatunkon a reverse zóna hiányában nem sikerül.

A Windows 2000 DHCP Server

Statikus IP címet használva a Windows 2000 tehát mind a saját A, mind a reverse *(PTR)* rekordját automatikusan regisztrálja. Ha az IP címét dinamikusan, *(Windows 2000)* DHCP kiszolgálótól kapja, akkor a DHCP címkiosztási folyamat során megállapodnak abban, hogy melyik adatot ki fogja bejegyezni. Az alapértelmezés ilyenkor az, hogy az A rekordot a kliens, a PTR rekordot a szerver regisztrálja. Ha a kiszolgáló válaszában nem jelzi, hogy képes lenne a regisztrációra *(például, mert régebbi DHCP kiszolgálóval állunk szemben)*, akkor a teljes regisztrációt elvégzi a kliens. A Windows 2000 DHCP Server szolgáltatása azonban képes a teljes regisztrációs folyamatot is magára vállalni, így a dinamikus DNS áldásos szolgáltatásait akár a Windows 95/98/Me/NT számítógépek is élvezhetik, ha az IP címüket egy Windows 2000 DHCP kiszolgálótól kapják.



A DHCP kiszolgáló DNS-beállításai

A DHCP kiszolgáló tulajdonságlapjának DNS oldalán az alábbi beállítási lehetőségeket találjuk:

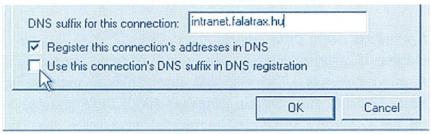
- Automatically update DHCP client information in DNS – a DHCP ügyféladatok automatikus frissítése a DNS-ben, mégpedig:
- Update DNS only if DHCP client requests – frissítés csak akkor, ha a DHCP kliens ezt kéri *(lásd fenn)*
- Always update DNS – frissítés minden esetben
- Discard forward *(name-to-address)* lookups when lease expires – ha ezt engedélyezzük, a DHCP kiszolgáló a bérleti idő lejártakor a DNS-ből törli az adott IP címet *(akkor is, ha azt nem ő regisztrálta be!)*
- Enable updates for DNS clients that do not support dynamic update – frissítés engedélyezése a DDNS-t nem ismerő kliensek nevében

RAS-ügyfelek IP-cím regisztrációja

A Windows 2000 RAS-ügyfelek betárcsázáskor regisztrálják mind az A, mind a PTR rekordokat. A vonal bontásakor megkísérlik a regisztrációk törlését is *(!)*, bár, ha a kérésre négy másodpercen belül nem érkezik válasz, bontják a vonalat *(és a tényről bejegyzés készül az Eseménynaplóba is)*. Az ügyfél kijelentkezésekor egyébként az RRAS kiszolgáló szolgáltatás is megpróbálkozik a PTR rekord deregisztrációjával.

Több hálókártya, több DNS kiszolgáló, egyedi DNS-utótagok

Ha a számítógépben több hálókártya található, a Windows minden hálózati csatoló elsődleges IP címét jegyzi be. Ha ezt nem szeretnénk, a regisztrálni nem kívánt hálózati csatoló TCP/IP protokolljának Advanced tulajdonságlapján, a DNS oldalon kapcsoljuk ki a „Register this connection's addresses in DNS" opciót.



További beállítások a hálózati csatolók TCP/IP protokolljánál

Ha a csatolókon különböző DNS kiszolgálókat definiáltunk *(például a belső hálókártyára a belső; az Internet felé néző hálókártyán pedig az Internet-szolgáltató DNS kiszolgálóját)*, akkor a Windows mindegyik DNS kiszolgáló felé csak az adott „irányból" elérhető IP címeket fogja regisztrálni. Ha valamelyik hálózati csatolón egyedi DNS utótagot definiálunk, és engedélyezzük az ábrán kurzorral jelölt „Use this connection's DNS suffix in DNS registration" opciót, akkor a Windows az itt megadott DNS utótagból képzett számítógépnevet is megkísérli majd regisztrálni.

Ha a számítógépben több hálókártya található, a Windows minden hálózati csatoló elsődleges IP címét jegyzi be.

A végére egy kis desszert...

Az automatikus DNS regisztrációt *(az A és PTR rekordok bejegyzését egyaránt)* az operációs rendszer szintjén letilthatjuk, ha 1-re állítjuk az alábbi *(REG_DWORD)* registry-értéket:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
Services\Tcpip\Parameters\DisableDynamicUpdate = 1
```

Ha csak a PTR rekordok regisztrációját szeretnénk letiltani, az érték neve legyen „DisableReverseAddressRegistration". Ha a fenti értéket nem a globális TCP/IP paraméterek között, hanem egy hálózati csatoló kulcsa alatt hozzuk létre, akkor a dinamikus regisztrációt csak az adott csatolóra tiltjuk le, például:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
Services\Tcpip\Parameters\Interfaces\
{FD3ABAAB-C5FB-4E74-ABEE-9BCF351B534E}\
DisableDynamicUpdate = 1
```

További információkat a Q246804 tudásbázis-cikkben találunk **[4]**. A Windows 2000 Server-en Service Pack 1 előtt az SRV rekordok a fenti *(és akár kártyánkénti)* beállítástól függetlenül is bejegyzésre kerültek; SP1 óta a dinamikus regisztráció tiltása érvényes az SRV rekordokra is **[5]**.

A Windows 2000 Server és Advanced Server üzemeltetői talán észrevették már, és nyilván nagyon csodálkoztak, hogy hiába tiltották egy-egy hálózati csatoló IP címének regisztrációját a kártya tulajdonságlapján, a Windows a következő alkalommal hajlamos volt a „tiltott" címet is bejegyezni a DNS-be. Ennek egyik oka lehet, ha a kiszolgálón fut a DNS szolgáltatás, ez ugyanis minden IP címet, amin keresztül elérhető, bejegyez az adatbázisba, függetlenül a hálózati csatolón található beállítástól. Ha ezt szeretnénk elkerülni, hozzuk létre az alábbi registry-értéket *(típusa: REG_SZ)*:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
Services\DNS\Parameters\PublishAddresses
```

Ide, szóközzel elválasztva felvehetjük a regisztrálni kívánt IP címeket – ha az értéket üresen hagyjuk, a DNS Server szolgáltatás a kiszolgáló összes IP címét regisztrálni fogja **[6]**.

folytatjuk...

Fülöp Miklós
mick@netacademia.net

A cikkben szereplő URL-ek:

[1] http://www.ietf.org/rfc/rfc2136.txt
[2] http://technet.netacademia.net/download/ddns/
[3] http://technet.netacademia.net/download/ddns/ddns-prereq.txt
[4] http://support.microsoft.com/default.aspx?scid=kb;en-us;Q246804
[5] http://support.microsoft.com/default.aspx?scid=kb;en-us;Q280439
[6] http://support.microsoft.com/default.aspx?scid=kb;en-us;Q275554

Ki mivel ♥?

Az életmentő LDAP

Januárban indítottuk útjára „**Ki mivel ♥**" rovatunkat, melyben időről időre beszámolunk egy-egy kacifántos, esetleg rejtélyes „ügy" megoldásáról. Ez a rovat nyitott: ha valaki munkája során olyan kíntapasztalatra tesz szert, mellyel mások szenvedését enyhítheti, kérem írjon nekünk a cikktipp@netacademia.net címre!

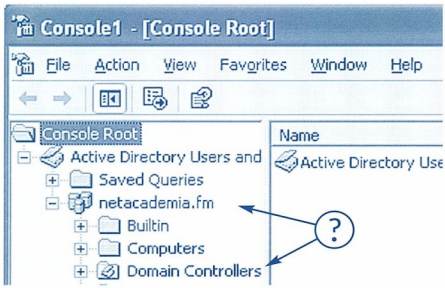
„A csatamezőn szerzett sebesülés még akkor sem szégyen, ha a hátunkba kapjuk, mert mire katonatörténet kerekedik belőle, a kínos részletek a feledés homályába merülnek..."

(Svejk)

I. felvonás

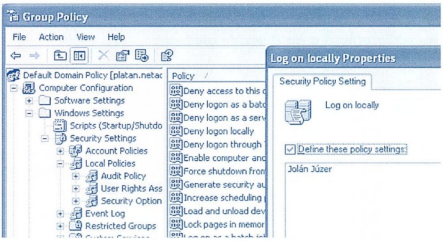
Történt egyszer, hogy X vállalatnál felmerült az igény, hogy egy gyalogos, mezei felhasználó bejelentkezési jogot kapjon a tartományvezérlőkön. Mint az közismert, Active Directory esetén a rendszerszintű jogosultságokat *(Log On Locally, Change the System Time stb.)* csoportos házirenddel *(group policy)* lehet kiosztani. A házirendet ez esetben mindig arra a szervezeti egységre kell illeszteni, amelyikben az érintett számítógép fiókja található. Ha a tartományvezérlő jogviszonyait kívánjuk módosítani, a Domain Controllers szervezeti egység házirendjét illik átállítani.

Illik, de nem muszáj! A házirendbeállítások öröklődése miatt ugyanis nemcsak az adott tárolón, hanem a felette lévő szinteken végrehajtott módosítások is érvényre jutnak. Az alábbi ábrán tehát mind a Domain Controllers, mind pedig a tartomány gyökere alkalmas célpont a Log On Locally jog kiosztására. Melyiket válasszam?



Hová tegyük a csoportos házirendet? Ezt mindig a feladat határozza meg.

- Ha magasabb ponton adagolom a jogot, messzebbre jut - gondolta a rendszergazda. Ebben tökéletesen igaza van. Kitt-katt, és a tartomány gyökerébe az alább látható módon felvette Józer Jolánt a helyi bejelentkezésre jogosult felhasználók *(egyébként üres)* listájába. Ám hiába állította be megfelelően a kívánatos jogosultságot, az csak nem akart érvényre jutni.



Helyi bejelentkezési jogot adunk Józer Jolánnak a tartomány gyökerén. Hogy mi lesz ebből...!

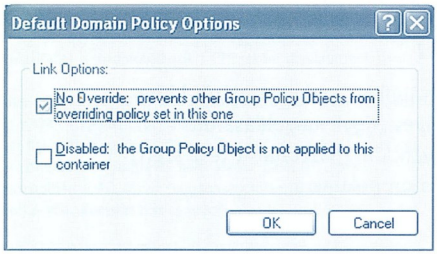
Várt-várt, majd az alábbi házirendfrissítési cselhez folyamodott:

```
secedit /refreshpolicy machine_policy
```

A fenti parancsnak a tudomány mai állása szerint soron kívül ki kellett volna értékelnie a házirend változásait - ám a jogváltozás továbbra sem lépett érvényre. Hősünk ekkor elkeseredett lépésre szánta magát: elővett egy könyvet, hogy utánanézzen a „hibának". A szakirodalom szerint kétféle házirendbeállítás létezik: az egyik típus összeadódik a hierarchia más pontjain található testvérkéivel *(ilyen például a szoftvertelepítés)*, a másik pedig felülírja szülei ellentétes utasításait *(például így működnek a regisztrymódosító parancsok)*. Mivel hibajelzést a rendszer nem adott, kézenfekvő, hogy a rendszerszintű jogokat a második csoportba kell sorolni, vagyis a gyermek felülbírálja a szülőt. Ez valóban így igaz: a Domain Controllers tárolón beállított jogok miatt a fentről jövő jogáldás nem érhette el célját. Ebből a kutyaszorító helyzetből két módon is ki lehet evickélni:

- vagy megalkuszunk, és áttesszük az erősebb *(tehát a gyermek-)* házirendbe a kívánatos módosítást
- vagy beavatkozunk a házirendöröklődés szokásos menetébe, és megemeljük a szülőnek beadagolt szabály prioritását

Hősünk nem egy megalkuvó típus, így a második megoldás mellett döntött. No Override! Senki ne eressze el a füle mellett az én parancsomat! Ez a házirendbeállításoknál itt található *(Group Pilicy, Options gomb)*:



A magasabb szinten elhelyezett házirend felülírhatósága megszüntethető a No Override pipával

A fenti policyfrissítési parancs után Józer Jolán végre be tud jelentkezni a tartományvezérlőkre. Csak éppen senki más nem!

II. felvonás

Helyzet: az egyetlen „normális" felhasználó a tartományban Józer Jolán!

Hősünk hamarosan észreveszi a malőrt, és szokásához híven kapkod fűhöz, fához, szalad a vargához... Miután ezzel végzett, elkezd gondolkodni: vissza kellene venni a policyról a No Override jelzést. Sajnos ahhoz Jolánnak nincs megfelelő jogosultsága. Sebaj, a Run As szolgáltatást pont az ilyen helyzetekre találták ki nemde? Nosza, Jolán bejelentkezéséből futtassuk az Active Directory Users and Computers eszközt! Futás helyett azonban ezt kapjuk:

Ha már a Minesweeper elindításához sincs elegendő joga az Administratornak, akkor komoly baj van!

Ez baj. Komoly baj. Más eszköz nem ismert a házirend módosítására, ADSI Edittel meg azért mégse. Milyen „szerencse", hogy az sem indul! A policy közvetlen visszaállítása tehát - eszköz híján - kiesett a megoldási lehetőségek közül. Vagy esetleg egy másik gépen bejelentkezve, távolról futtatva? És ekkor jön a sokk: **a vállalat összes gépéről kitiltódott mindenki!** *(Nyilván. Hisz a tartomány gyökerétől erőszakkal végigöröklötetett Log On Locally beállítás a munkaállomásokra is lecsorgott!)* Most mi lesz?

III. felvonás

Úgy látszik, nincs más megoldás: meg kellene „hekkelni" a tartományt, hogy egyetlen felhasználónk, Józer Jolán rendszergazdai jogkörbe kerüljön, és képes legyen visszavenni a házirend erősségéből. Azonban az Active Directoryt nem olyan puha anyagból gyúrták, hogy csak úgy mindenféle nyomásnak engedjen. Elevating privileges? Ilyen biztonsági lyuk nincs benne.

Használjuk fel azt a tényt, hogy ismerjük a rendszergazda nevét és jelszavát. Igaz ugyan, hogy egyetlen gépen sem tud bejelentkezni, de távolról, más tartományból, vagy más operációs rendszerről a név+jelszó egyezés miatt mégiscsak jutna valamire. Más tartomány nincs kéznél, s telepíteni is időigényes lenne. Más operációs rendszer sincs, de van valami, amivel az idegenség látszatát lehetne kelteni, s így be lehetne csapni a Windowst: dehogyis akar helyileg bejelentkezni az Administrator! Csakis távolról! Az LDAP protokoll, még pontosabban a címtár exportálása-importálása *(LDIFDE segítségével)* segít nekünk a cselezésben. Tavaly májusi számunkban olvasható az LDIF fájlok használatának részletes leírása, itt most csak az elvet és a megoldást közlöm.

Célunk: a kiütött rendszergazda segítségével „helyzetbe hozni" Jolán felhasználói fiókját, hogy engedelmeskedjenek neki a rendszergazdai eszközök. Ehhez egy olyan LDIF fájlt készítünk (modosit.txt), mely Jolánt beemeli a Domain Admins csoportba:

```
DN: CN=Domain Admins,CN=Users,DC=falatrax,DC=hu
changetype: modify
add: member
member: CN=Juzer Jolan,CN=Users,DC=falatrax,DC=hu
-
```

Ezt a fájlt az alábbi paranccsal importálhatjuk be az Active Directoryba *(ahol -i az import üzemmódot, -f a beemelendő fájlt, -b pedig a felhasználó azonosítási adatait állítja be)*:

```
LDIFDE -i -f modosit.txt -b Administrator falatrax
*
```

Maga az import tehát az eredeti Administrator erősségével fut. Az LDIFDE szerencsére nem kér interaktív bejelentkezést az operációs rendszertől, így akadálytalanul beemeli a címtárba ravasz módon megfogalmazott importfájlunkat. Jolán Domain Adminná vált. Gyors kijelentkezés-bejelentkezés után *(hogy a csoporttagság-változás érvényre jusson)*, máris rendet lehet tenni a szétesett címtárban. Ezúttal: Happy End.

Fóti Marcell
MCSE

Mikor és miért felel az Internet szolgáltató?

A 2001. évi CVII. törvény legfontosabb előremutató rendelkezése a szolgáltatói felelősség szabályozása. A kérdés alapvetően természetesen az, hogy a hálóra felkerült tartalomért a szolgáltatást nyújtók mely csoportja, és milyen mértékű felelősséggel tartozik.

A jogalkotó mindenekelőtt kimondja, hogy a szolgáltató *(ezalatt bármely típusú szolgáltatás nyújtóját értve)* a polgári jog általános szabályai szerint felel az információs társadalommal összefüggő szolgáltatás során továbbított, tárolt vagy hozzáférhetővé tett, jogszabályba ütköző tartalmú információval okozott jogsérelemért, illetve kárért. Ezt követően kerül sor a felelősség alóli mentesülés eseteinek meghatározására.

A polgári jog általános szabályai szerinti felelősség megállapításához szükséges, hogy az adott magatartás jogellenes és felróható legyen. A jogellenesség ténylegesen azt jelenti, hogy a magatartás *(vagy mulasztás)* valamely jogszabályba ütközzön, jogszabály által biztosított vagy védett jogot sértsen, vagy kötelezettségnek ne tegyen eleget. A felróhatóság viszont a Polgári Törvénykönyvben úgy kerül meghatározásra, hogy a károkozó számára mentesülést jelenthet, ha magatartása nem volt felróható. Vagyis alapvetően feltételezi a jogalkotó, hogy a jogellenes magatartás egyben felróható is, de lehetőséget ad a károkozó számára, hogy bizonyítsa, hogy az adott helyzetben úgy járt el, ahogy az az adott helyzetben általában elvárható volt. A bizonyítás kötelezettsége tehát a szolgáltatóé, és nem azé, aki kártérítési igényt kíván érvényesíteni. Kérdés – és itt kapott korábban nagy teret a bírói gyakorlat – hogy egy Internet szolgáltató esetében, ha mondjuk több tízezer oldal tárhelyet biztosít, mi az elvárható magatartás a jogellenes tartalmú információk közzétételének megakadályozására? Elvárható-e, hogy naponta, vagy annál gyakrabban ellenőrizze a több tízezer oldalon, hogy vajon felkerült-e jogsértő tartalom oda vagy sem? Egy vitafórum működtetése esetében milyen mértékű moderálás várható el? Az ilyen, és ehhez hasonló kérdésekre részben választ ad a törvény, amikor a mentesülés eseteit határozza meg.

A szolgáltatónak egy adott jogsértő tartalomért való felelősség alóli mentesüléshez egyszerre két kérdésnek kell megfelelnie: tartózkodjon a törvényben konkrétan meghatározott formájú, aktív közreműködést megvalósító magatartástól, továbbá tegyen eleget a notice and take down eljárás megindítása esetén az ebből eredő kötelezettségeinek.

A továbbiakban a törvény valamennyi szolgáltató vonatkozásában kimondja, hogy nem köteles előzetesen és rendszeresen ellenőrizni az általa csak továbbított, tárolt, hozzáférhetővé tett információ tartalmát, továbbá nem köteles olyan tényeket vagy körülményeket keresni, amelyek jogellenes tevékenység folytatására utalnak. Ezt követően kerül sor az egyes szolgáltatásokhoz fűződő magatartások meghatározására.

A távközlő hálózaton történő adattovábbítás, vagy a távközlő hálózathoz való hozzáférés biztosítása *(egyszerű adatátvitel)* esetén a szolgáltató akkor nem felel, ha nem maga kezdeményezi az információ továbbítását, nem maga választja meg a címzettet, nem maga választja ki az információt, illetve azt nem változtatja meg. Az egyszerű adatátvitel fogalmába tartozónak érti a törvény az információ közbenső és átmeneti jellegű automatikus tárolását is, amennyiben ennek célja kizárólag az információtovábbítás lebonyolítása, és nem tart hosszabb ideig, mint ami a továbbításhoz szükséges.

Az a szolgáltató, aki az információt a távközlő hálózaton továbbítja és ezzel alapvetően a mások által kezdeményezett információtovábbítást teszi hatékonyabbá, az információ közbenső és automatikus tárolása esetén akkor nem felel, ha

- nem változtatja meg az információt, az információhoz való hozzáférést a megfelelő feltételek szerint biztosítja,
- a közbenső tárolóban az információ frissítése megfelel a széles körűen elismert, és alkalmazott információfrissítési gyakorlatnak,
- a közbenső tárolás nem zavarja meg az információ felhasználásával kapcsolatos adatok kinyerésére szolgáló, széles körűen elismert és alkalmazott technológia jogszerű használatát, és
- a szolgáltató haladéktalanul eltávolítja az általa tárolt információt vagy nem biztosítja az ahhoz való hozzáférést, amint tudomást szerez arról, hogy az információt az adatátvitel eredeti kiindulási pontján a hálózatról eltávolították, vagy az ahhoz való hozzáférés biztosítását megszüntették, illetve hogy a bíróság vagy más hatóság az eltávolítást vagy a hozzáférés megtiltását rendelte el.

A tárhelyet biztosító szolgáltató akkor mentesül a kártérítési felelősség alól, ha nincs tudomása arról, hogy a tárolt információ jogellenes, vagy bárkinek jogát, jogos érdekét sérti, illetve, amint erről tudomást szerez, haladéktalanul intézkedik az információ eltávolításáról.

Ez utóbbi rendelkezés némi aggodalomra adhat okot, hiszen nyitott kérdés, hogy a tudomást szerzés mivel merül ki? Tudomást szerez-e a szolgáltató, ha valaki vélt vagy valós jogsértésről tájékoztatja? Jogosult-e a szolgáltató ez esetben haladéktalanul eltávolítani az információt, vagy meg kell várnia, amíg a jogellenességet valamilyen bírói vagy más határozat jogerősen kimondja?

Az említett aggályt leszámítva a szolgáltatók mentesülési esetei logikusan, egymásra épülve kerültek meghatározásra, és elfogadható a valamennyi közös alapelvként szolgáló gondolat, hogy a szolgáltató nem felel azért, amiről nincs tudomása. Bármily egyértelműnek tűnik is ma már ez a meghatározás, korábban számos olyan bírói döntés volt mind az amerikai, mind az európai jogesetek körében, amely ezzel ellentétesen, megállapította a „vétlen" szolgáltató felelősségét is azért a tartalomért, amelyről nyilvánvalóan nem lehetett tudomása. Így ítélt el a svájci bíróság egy Internet szolgáltatással, tárhely biztosítással foglalkozó egyetemistát, aki a több tízezer oldal közül nem

szűrte ki és nem távolította el az ott elhelyezett jogellenes, esetünkben történetesen éppen pornográf tartalmú információt.

A törvény alapjául szolgáló Irányelve az Európai Uniónak nem rendelkezik a keresőszolgáltatást működtető Internet szolgáltatók felelősségéről. A magyar jogalkotók azonban erre is kiterjesztették a normatív jogalkotást, így a Yahoo eset tanulságait leszűrve a szolgáltatót mentesíti a törvény a kereséssel fellelhető információk esetleges jogellenessége miatti felelősség alól, ha a szolgáltatónak nincs tudomása az információ jogellenességéről, vagy nincs tudomása olyan tényről vagy körülményről, amely valószínűsítené *(!)*, hogy az információval kapcsolatos magatartás jogellenes vagy arról, hogy az információ bárkinek jogát vagy jogos érdekét sérti. Itt ismét a régi dilemma köszönt ránk vissza: vajon ha a keresőszolgáltatást fenntartónak van tudomása olyan tényről, amely szerint esetleg az adott információ jogellenessége valószínűsíthető, akkor már felelősségre vonható azért, mert elősegíti az információ megtalálását? Ha jól belegondolunk, olyan mértékű gondosságot követel meg ez a szabály a szolgáltatótól, amely ellentétes a polgári jogban használatos „általában elvárható" magatartás mércéjével. A jogellenesség valószínűsége olyan megfoghatatlan, annyira az ügyben majd eljáró bíró személyiségétől és más körülményektől függő mérlegelésnek tág teret engedő meghatározás, amelynek nem igen lenne helye egy állami normában. A Yahoo esetére emlékezve, ahol a francia bíróság arra kötelezte az amerikai székhelyű Yahoo-t, hogy az általa működtetett keresőszolgáltatást állítsa át úgy, hogy francia felhasználóktól kezdeményezett keresések esetére ne tegye elérhetővé az Amerikában amúgy nem illegális, náci relikviákat hirdető web oldalakat, megállapítható, hogy a magyar AltaVizsla és más keresőprogramok fenntartói is hasonló, de még kártérítés megfizetésére is kötelező határozatokra számíthatnak ezen szakasz alapján. A bizonytalan megfogalmazású jogszabályok végső értelmezését mindig a kialakuló bírói gyakorlat adja majd meg, de mint ügyvéd állítom, hogy nyilván nem a helyes jogszabályi szövegezésre utal, amikor a tanácsért hozzám forduló szolgáltatónak adott esetben a kezemet széttárva nem tudom megmondani, hogy milyen döntésre számíthat majd az adott ügyben.

A szolgáltatók tehát a törvényi logika szerint általában nem felelnek azokért a jogsértő tartalmú információkért, amelyekhez való hozzáférésben valamilyen módon közreműködtek, de amelyek jogsértő mivoltáról nem volt tudomásuk. Ez a megoldás rendben is lenne, ha nem toldaná meg a jogalkotó azokkal a további feltételekkel a mentesülés esetét, amelyeknek a szolgáltató valójában már csak akkor tudna ténylegesen megfelelni, ha vállalná az előminősítést, vagyis mintegy előzetes bíróságként eljárva vállalná annak felelősségét, hogy eldöntse, hogy a hozzá érkezett panasszal érintett információ jogsértő-e vagy sem, vagy van- e olyan tény vagy körülmény, amely valószínűsítené, a jogellenességet. Ez a megoldás ténylegesen arra fog vezetni, hogy a szolgáltató félve az esetleges kártérítési felelősségtől, minden további nélkül, automatikusan el fogja távolítani az információt, vagy megakadályozza az ahhoz való hozzáférést, ha bármilyen arra utaló jelet kap, hogy az esetlegesen jogsértő tartalmú. Így az eredetileg tervezett, széles körű notice and take down eljárás helyett egy rosszabb, a tartalom szolgáltatójának jogorvoslati lehetőséget nem adó megoldás született.

A notice and take down eljárás lényege ugyanis az, hogy az ismeretlen tartalomszolgáltatót kilétének felfedésére kényszeríti, ha az ragaszkodik a jogsértő tartalom elérhetőségéhez. Vagyis ha valaki azt észleli, hogy valamely Interneten található információ jogsértő, és az információt elhelyezőt nem tudja azonosítani, úgy az azonosítható Internet szolgáltatót felhívja a jogsértő tartalom eltávolítására vagy a hozzáférés megakadályozására. E felhívásnak a szolgáltató – a tartalom szolgáltatójának egyidejű értesítése mellett – köteles eleget tenni. A tartalom szolgáltatója azonban a saját azonosításához szükséges adatok közlésével kérheti az információ visszahelyezését, és az ahhoz való hozzáférés helyreállítását. Az eljárás lényege tehát az, hogy a panaszos számára azonosíthatóvá, és így perelhetővé tegye a tartalom szolgáltatóját. Az a tartalomszolgáltató tehát, aki hajlandó vállalni a felelősséget az általa szolgáltatott tartalomért, kilétének felfedésével elérheti annak visszahelyezését illetve a hozzáférés helyreállítását.

Az eljárás nem túlságosan terjedt el, leginkább az önszabályozó kódexekben találkozhatunk vele. Az elektronikus kereskedelemről szóló magyar törvény tervezete azonban eredetileg erre alapozta a jogsértés elleni hatékony védekezést. Aggályos volt azonban, hogy ha bárkinek bármilyen jogsértés esetére nyitva hagyjuk az eljárás megindítására a lehetőséget, úgy számos olyan eset lesz, ahol ezzel a joggal visszaélve ténylegesen a véleménynyilvánítás szabadságát korlátozzák, vagy éppen fontos üzleti akciót akadályoz meg a versenytárs. Ezért a szakmai lobbi nyomására a törvénybe már szűkebb körre vonatkoztatva, a szerzői jog megsértésének orvoslására épült be a notice and take down eljárás. Így akinek szerzői jogi védelem alatt álló jogát sért egy adott információ, kérheti annak eltávolítását. Az információt elhelyező tartalomszolgáltatónak pedig 8 nap áll rendelkezésére a kifogás előterjesztésére. Ha kellőképpen azonosította magát, úgy a panaszosnak további 10 nap áll rendelkezésére ahhoz, hogy keresetet nyújtson be a tartalomszolgáltatóval szemben. Ez esetben az információhoz való hozzáférést ismét megszünteti az Internet szolgáltató a bíróság döntéséig.

Az eljárás így tehát csak a szerzői jogok fokozott védelmét garantálja egy olyan környezetben, ahol a szerzői jog megsértése könnyebben és büntethetetlenül történhet, mint a nyomtatott vagy hagyományos média világában. Gyakorlatilag azonban a szolgáltatói felelősség ismertetésénél bemutatott más, a jogsértő információ eltávolítására, illetve a hozzáférés megakadályozásra vonatkozó kötelezettségeket tekintve megállapíthatjuk, hogy az eltávolítási kötelezettség minden esetben terheli a szolgáltatót, ha annak akár csak gyanúja merül is fel, hogy valamely adott tartalom jogsértő lehet. Ezekben az esetekben azonban, a tartalomszolgáltatónak még csak a gyors jogorvoslatra sem nyílik lehetősége.

Noha a törvény hatálya deklaráltan csak a véleménynyilvánítás szabadsága által védett terület határáig terjed, e rendelkezések arra utalnak, hogy a jogalkotó e határon átlépve bizony a véleménynyilvánítás szabadságát is bizonyos mértékig korlátozta.

ASP Suli:
Outlook Wap Access

Az elmúlt két hónapban megismerkedtünk a WML programozás alapjaival. Most itt az ideje, hogy valami maradandót alkossunk: cikkünkben az Exchange 2000 CDO objektummodellje segítségével megvalósítjuk az Outlook Web Access szuperkompakt változatát: ez lesz az Outlook Wap Access.

A CDO for Exchange 2000

A CDO for Exchange 2000 objektummodell *(CDOEX)* az általunk már korábban megismert CDO objektummodell-család legfiatalabb tagja. A CDOEX az Exchange 2000 telepítésével kerül a számítógépre, működése során az Exchange 2000 és az Active Directory környezetet igényli *(az előbbit ráadásul lokálisan)*. Éppen ezért tartsuk fejben, hogy az **[1]** címről letölthető példaprogramok *(a példaként bemutatott fájlok neveit zárójelben mindig jelezzük)* csak az Exchange kiszolgálóként üzemelő számítógépen működnek majd megfelelően. A CDOEX lefedi a teljes Exchange-funkcionalitást, azaz segítségével szinte bármit megtehetünk, amit az Exchange-ben meg lehet tenni *(levelet írni-olvasni, naptárat, névjegykártyákat kezelni, találkozót szervezni, stb.)* – mi itt szigorúan a levelek olvasásában szorítkozunk a CDOEX segítségére. Aki ennél többet akar, az a CDOEX teljes referenciáját megtalálja a **[2]** címen.

A kiszolgáló előkészítése

Mielőtt a komolyabb munkába belekezdenénk, hozzunk létre a webkiszolgálónkon egy virtuális mappát, ahova majd az alkalmazásunk kerül. Ne felejtsük el véghezvinni az alábbi beállításokat *(teljeskörű leírásuk a tech.net magazin 2002/02. számában található meg)*:

- Definiáljuk a WML tartalomtípusokat *(.wml, .wbmp)*
- Kapcsoljuk ki a webalkalmazásban a munkamenetek *(Session)* kezelését, úgysem lesz rá szükség
- Engedélyezzük a nyíltjelszavas *(Basic)* felhasználóazonosítást.

Igen, Kedves Olvasó! Utóbbi beállítás természetesen nem más, mint egy kövér kis biztonsági rés. A titkosított wapos kommunikáció nem jelen cikkünk témája, ezért egyelőre a hagyományos felhasználóazonosítást választjuk. Éles környezetben el kellene gondolkodnunk azon, hogy kinek és mihez engedélyezzük a hozzáférést, hiszen az így elfogott hálózati forgalomból a felhasználó jelszava könnyen visszafejthető. Az itt bemutatott példaalkalmazás nem csak ezen a ponton sántít, biztosan sok helyen egyszerűbben, vagy főleg elegánsabban is meg lehetne oldani dolgokat – ezt már az Olvasó fantáziájára bízom.

A lényeg a körítés

A webalkalmazás beállításait, a tartalomtípusok helyes működését kipróbálhatjuk, ha létrehozzuk az OWA bejelentkező képernyőjét képező oldalt. Ehhez korábban már – nem kis munkával ;-) – létrehoztunk egy csinos kis logót *(owa.wbmp)* is. Az oldalban *(default.asp)* definiált kód pedig a következőképpen fest:

```
<%
If InStr(Request.ServerVariables("HTTP_ACCEPT"),
↳ "wml") = 0 Then
   Response.Redirect "http://mail.falatrax.hu/
↳     exchange/"
End If

Response.ContentType = "text/vnd.wap.wml"
%>
```

(1)

A kód elején – biztos, ami biztos – megnézzük, hogy a böngésző képes-e a WML tartalomtípus fogadására *(azaz, a HTTP_ACCEPT fejlécek tartalmazzák-e a „wml" szót)*. Ha nem, akkor valószínűleg nem wapos, hanem hagyományos böngészővel van dolgunk, amit egyszerű átirányítással továbbküldünk a hagyományos Exchange Outlook Web Access felé *(1)*. Ellenkező esetben kezdődhet a wapos feldolgozás. Legelső dolgunk legyen mindig a wml tartalomtípus beállítása:


```
Response.ContentType = "text/vnd.wap.wml"
```

Az oldal ezután a logót, majd egy hivatkozást tartalmaz a postaládához vezető oldalra. Ezt a hivatkozást a könnyebb kezelhetőség érdekében hagyományos link *(<a>)* és menüparancs *(<do>/<go>)* formában is definiáltuk, bár erre nem lenne feltétlenül szükség.

Meg kell jegyeznünk, hogy a wapos programozás aranyszabályaiban valahol az elsőközött szerepel, hogy az idő, a sávszélesség *(és a pénztárcánk)* kímélése érdekében tartózkodjunk a hasonló, cél nélküli, gyakorlatilag értelmetlen üdvözlő képernyőktől. Demonstrációs célra viszont kitűnően megfelel, és aki ezt az oldalt nem szeretné látni, hivatkozhat majd közvetlenül a postaláda címére is.

A Person objektum és a postaláda elérési útja

A *tools.asp* fájlban néhány olyan függvény található, amit több helyről is használni fogunk, és felesleges lett volna többször megírni. Ilyen többek között a felhasználó postaládája helyének keresése, vagy maga a postaláda megnyitása is. Az első függvény mindjárt az előbbi feladatot végzi el:

```
USERDOMAIN = "@falatrax.hu"

Function GetMailboxURL( sUser )
```

```
    On Error Resume Next

    Set oPerson = Server.CreateObject
    ("CDO.Person")
    strURL = "mailto:" & sUser & USERDOMAIN

    oPerson.DataSource.Open strURL
    GetMailboxURL =
    oPerson.GetInterface("IMailbox").Inbox

    Set oPerson = Nothing

    If Err.Number <> 0 Then
       GetMailboxURL = ""
    End If
End Function
```

A Person *(CDO.Person)* objektum egy Active Directory-beli felhasználót személyesít meg. A felhasználó adatait a Person objektum IDataSource interfésze *(amit a .DataSource jellemző segítségével érünk el)* segítségével tölthetjük be, ehhez az Open() metódusnak át kell adnunk az objektumra mutató URL-t. Esetünkben ehhez tökéletesen megfelel a felhasználó nevéből képzett e-mail-cím *(a USERDOMAIN változó persze a valós e-mail domaint tartalmazza)*. A lényeg, a bejövő postafiókra mutató URL pedig a Person objektum IMailbox interfésze segítségével kérdezhető le. Ehhez először magát az interfészt kell előcsalogatnunk az objektum GetInterface() metódusa segítségével. Maga az IMailbox interfész egy felhasználó komplett postaládáját jelképezi, bejövő, kimenő mappákkal, naptárral, egyebekkel együtt. A beérkezett üzenetek mappájának elérési útját az interfész .Inbox jellemzője tartalmazza. Az elérési út egyébként valahogy így fest:

```
file://./backofficestorage/falatrax.hu
/MBX/mick/Inbox
```

Ennek az elérési útnak a birtokában már könnyű hozzáférni a postaláda adott mappájában található levelekhez.

A postaláda-Recordset

A CDOEX szorosan összenőtt az ADO objektummodellel (is). Ha például egy mappa leveleit szeretnénk böngészni, az Exchange adatbázist elérhetjük a megszokott módon, ADO-ból, Recordseteken keresztül, az ADO minden előnyét kihasználva. Ez annak köszönhető, hogy az Exchange 2000-hez készült egy OLE DB provider, aminek segítségével az adattár közvetlenül elérhető. A bejövő üzenetek megnyitásának eredménye tehát egy ADO Recordset lesz *(az ADO adatbáziskezelésről többek között novemberi számunkban írtunk, de az objektummodell referenciája elérhető a* **[3]** *címen)*. A következő megemésztendő feladat tehát a Recordset megnyitása, amit ugyancsak a *tools.asp* fájlban találunk, mert több helyen is szükség lesz rá.

```
Function GetMailbox(sInboxURL)
    Set oRC = Server.CreateObject("ADODB.Record")
    oRC.Open sInboxURL

    Set oRS =
    Server.CreateObject("ADODB.Recordset")
    oRS.CursorType = 3 ' static
    oRS.CursorLocation = 3 ' client
    oRS.LockType = 1 ' read only
    ' sSQL = lásd alább

    oRS.Open sSQL, oRC.ActiveConnection, 3, 1, 1
    ' static, read only, text
    oRS.PageSize = 5
    Set GetMailbox = oRS
End Function
```

Legelőször is, egy rekordként meg kell nyitnunk magát a postaládát, hogy majd a rekord „adatbázis-kapcsolatát" felhasználhassuk a Recordset megnyitásához. A példaprogramból a könnyebb értelmezhetőség kedvéért kivágtuk a lekérdezéshez szükséges SQL stringet:

```
SELECT
  "urn:schemas:httpmail:sendername" As SenderName,
  "urn:schemas:httpmail:fromemail" As FromEmail,
  "urn:schemas:httpmail:from" As From,
  "urn:schemas:httpmail:to" As To,
  "urn:schemas:httpmail:textdescription" As
  MsgText,
  "urn:schemas:httpmail:date" As Received,
  "urn:schemas:httpmail:subject" As Subject,
  "http://schemas.microsoft.com/exchange/mid" As
  MID

FROM
  scope ('shallow traversal of """
  & sInboxURL & """')

WHERE
  "DAV:isfolder"=false AND "DAV:ishidden"=false"

ORDER BY
  "urn:schemas:httpmail:datereceived" DESC
```

A lekérdezésben kiválasztjuk a nekünk szükséges mezőket *(a teljes lista a referenciában megtalálható)*, ezek:

- sendername, from: A feladó teljes neve
- fromemail: a feladó e-mail címe
- to: a címzett neve
- textdescription: a levél törzsének szöveges tartalma *(tökéletes: automatikus konverzió HTML levélről, nem zavarnak a csatolt fájlok, stb.!)*
- date: a levél feladásának dátuma
- subject: a levél témája
- MID: a levél Exchange adatbázison belüli egyedi azonosítója *(MessageID, ennek segítségével hivatkozhatunk később az üzenetre)*.

A FROM részben jelezzük, hogy a postafiók *(fentebb látott)* URL-je és az azalatti mappákban keresnénk, a WHERE kitételben pedig azt, hogy a mappák illetve a rejtett *(törölt, rendszer, stb.)* üzenetek ne kerüljenek be a válaszba. Végül, az egész listát a levelek beérkezési időpontja szerinti csökkenő sorrendbe rendezzük. A Recordset megnyitása után még beállítjuk a lapméretet 5-re *(ennyi üzenetet kezelünk majd oldalanként)*.

A bejövő üzenetek listája

Ennyi előkészítés után lássuk az *inbox.asp* kódját, ami a bejövő üzenetek lapozható listáját varázsolja elénk. Mindenekelőtt jelezzük, hogy a *tools.asp* kódját is szeretnénk elérni az oldalból:

```
<!-- #include file="tools.asp" -->
```

A postaláda kódja így folytatódik:

```
If Request.ServerVariables("AUTH_USER") = "" Then
    Response.Status = "401 Unauthorized"
    Response.End
End If

Response.ContentType = "text/vnd.wap.wml"
Response.Expires = -1
Response.AddHeader "Cache-Control",
↳ "no-cache, must-revalidate"
Response.AddHeader "Pragma", "no-cache"
```

Mindenekelőtt, ellenőrizzük, hogy a felhasználó bejelentkezett-e. Ha nem *(azaz a megadott felhasználónév üres)*, akkor mindössze egy „401 Unauthorized" HTTP hibaüzenetet adunk vissza, és rögtön be is fejezzük a működést. Ennek hatására a böngésző bekéri majd a felhasználónevet és jelszót, majd újra próbálkozik.
Ezután beállítjuk a wml tartalomtípust, majd jelezzük, hogy az oldal ne kerüljön sehol gyorsítótárba *(-1-es lejárati idő, Cache-Control, Pragma fejlécek)*. Ez a néhány sor a webalkalmazásunk minden fájljának elején megtalálható. A következő lényegesebb részlet a fájlban:

```
Set oRS = GetMailbox(sInboxURL) ' ld. tools.asp

oRS.Filter = ""

If oRS.RecordCount = 0 Then
  Response.Write "<p align='center'>Nincs új
  ↳ üzenet."
  Response.Write "<br/>"
Else
  If Request("pg") <> "" Then
    oRS.AbsolutePage = CInt(Request("pg"))
  End If
  Response.Write "<p align='center'><b>" &
  ↳ RS.RecordCount & " üzenet. </b>"
  Response.Write oRS.AbsolutePage & "/" &
  ↳ oRS.PageCount & " old.</p>"
...
```

Miután a *tools.asp*-ben már bemutatott GetMailbox*()* segítségével hozzájutottunk az üzeneteket tartalmazó Recordset-hez, a Recordset .RecordCount jellemzőjét kiolvasva kiderül, hogy van-e levél. Ha ez az érték 0, akkor a postaládában nincs üzenet. Ha nem, következhet az üzenetek listázása. A korlátozott méret miatt oldalanként csak néhány üzenet listázására van mód, ezért lapozni kell. Korábban már bemutattuk, hogy az ADO Recordset objektum hogyan támogatja ezt. Az .AbsolutePage jellemző az aktuális oldal sorszámát jelzi *(az oldalak méretét már a tools.asp-ben beállítottuk)*. Ha az .AbsolutePage értékét felülírjuk *(mint itt is, ha a kérésben szerepelt a pg változó)*, akkor a Recordset kurzor is az aktuális oldal első rekordjára ugrik.

A beérkezett üzenetek közötti navigációt az ADO Recordset lapkezelése segíti. Ha szükség van rá, menüparancsokat (második kép), és hagyományos linkeket (harmadik kép) is definiálunk a lapozáshoz

A szöveg normalizálása

Ne felejtsük el, hogy az oldalak maximális mérete erősen köti a kezünket, ezért a feladók, címzettek, témák hosszát ellenőriznünk kell, mielőtt a telefonra küldjük. Ezért a kódban számos helyen látható, hogy a Left*()* függvény segítségével itt-ott elvágjuk a szövegeket. A másik nagyon fontos tényező, hogy bizonyos karaktereket konvertálnunk kell a WML-beli azonosítóra. A Server.HTMLEncode*()* metódus erre elvileg megfelelő lenne, de az például a tipikusan WAP-specifikus $ jelet nem alakítja $$-ra – egyszerűbb a saját megoldás. Ehhez a *tools.asp* fájlban definiált Normalize*()* függvényt hívogatjuk bőszen szerte a webalkalmazásunkban.

Az üzenet olvasása

A következő feladat az üzenet teljes megjelenítése *(read.asp)*. Ehhez rendelkezésünkre áll az üzenet azonosítója:

```
sMID = Request("mid")
```

A postaláda-Recordset szokásos módon történő megnyitása után az adott azonosítójú üzenetre szűrjük:

```
On Error Resume Next
oRS.Filter = "MID=" & sMID

If Err.Number <> 0 Then
  oRS.Filter = ""
  Response.Redirect("inbox.asp")
End If
On Error Goto 0
```

A hibakezelés lényege az, hogy ha a szűrés során bármilyen hiba keletkezne *(pl. nincs ilyen rekord)*, akkor visszaugrunk a postaládához. Ezután a szöveg oldalakra tördelt megjelenítése következik *(hasonlóan, mint a postaládánál, csak most nem segít nekünk a Recordset – mindent kézzel kell csinálni)*.

A hosszabb üzeneteket is lapozni kell. A módszer ugyanaz, mint a postaládánál (menüparancsok, linkek), csak most nincsenek Recordset-oldalak

Üzenet törlése

Az üzenet olvasása során definiálunk egy menüparancsot, ami az üzenet törlését kezdeményezi. Ez a menüparancs a *del.asp* oldalra ugrik, és átadja neki az MID-t. Az üzenet törlése során egy írható Recordset-ben megnyitjuk a postaládát, azt szűrjük a törölni kívánt áldozat üzenet azonosítójára, majd töröljük az aktív rekordot *(del.asp)*:

```
Set oRec = Server.CreateObject("ADODB.Record")
oRec.Open sInboxURL

Set oRSD = Server.CreateObject("ADODB.Recordset")
oRSD.CursorType = 1
oRSD.CursorLocation = 2
oRSD.LockType = 2

sSQL = "SELECT ""http://schemas.microsoft.com/
↳ exchange/mid"" As MID " _
  & " FROM scope ('shallow traversal of """ &
↳ sInboxURL & """')" _
  & " WHERE ""DAV:isfolder"" = false AND
↳ ""DAV:ishidden"" = false" ' _

oRSD.Open sSQL, oRec.ActiveConnection, 1 , 2 , 1
↳ ' cursor, lock, text

oRSD.Filter = "MID=" & sMID

oRSD.Delete
oRSD.Update
```

Az Exchange teljes OLE DB integrációjának köszönhetően a rekord törlése egyben a levél törlését is jelenti.

Új üzenet *(és válasz egy üzenetre)*

A két dolog mindössze annyiban különbözik, hogy az utóbbi esetben átveszünk egy MID-t, így ki tudjuk másolni az eredeti üzenet feladóját *(ő lesz az új címzett)*, illetve témáját. Az eredeti üzenet szövegét azonban érthető okokból nem idézzük a levélbe. Az üzenet írását mindkét esetben a *new.asp* végzi, számunkra azonban sokkal érdekesebb az üzenet elküldését végző *send.asp* kódja. Mindenekelőtt, a *send.asp* nem tartalmazza a felhasználóazonosítás kikényszerítését. Ennek az az oka, hogy a levél írása során a telefon gyakran bontja a kapcsolatot *(időtúllépés miatt)*, és az újbóli kapcsolatfelvétel miatt elveszik a felhasználónév-jelszó páros. Ilyenkor újra be kell jelentkezni, ennek hatására viszont sok telefonon elveszik az addig bepötyögött adat... kellemetlen. Maga a levél elküldése az általunk már korábbról ismert CDONTS.NewMail objektum segítségével történik:

```
sSub = Request("sub")
sBody = Request("msg")
sPg = Request("pg")

sSub = ConvertToCharset( sSub, "us-ascii" )
sBody = ConvertToCharset( sBody, "iso-8859-2" )

Set oNewMail =
↳ Server.CreateObject("CDONTS.NewMail")

oNewMail.From = Request("from")
oNewMail.To = Request("to")
oNewMail.Subject = sSub
oNewMail.Body = sBody & vbCRLF & vbCRLF &
↳ "[Sent by Outlook Wap Access]"
oNewMail.Send
```

A ConvertToCharset*()* metódus a tools.asp fájlban található, és feladata, hogy *(az ADO Stream objektum segítségével)* a kapott szöveget a megadott kódtáblára konvertálja. Ezután a CDO.Message objektum segítségével a levelet – ha éppen ismerjük a felhasználó nevét – elmentjük az elküldött üzenetek mappájába.

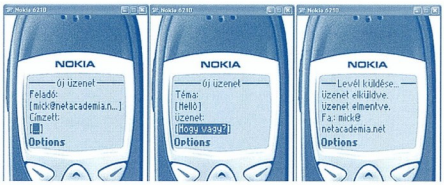
```
sEmail = sUser & USERDOMAIN
sSentURL = GetSentItemsURL(sUser)

Set oMsg = Server.CreateObject("CDO.Message")
oMsg.From = Request("from")
oMsg.To = Request("to")
oMsg.Subject = sSub
oMsg.TextBody = sBody

oMsg.DataSource.SaveToContainer( sSentURL )
```

A GetSentItemsURL*()* függvényt *(ami az elküldött üzenetek mappájának URL-jét adja vissza)* a tools.asp fájlban találjuk, és a bejövő üzenetek mappájának lekérdezésétől egyetlen sorban különbözik: most az IMailbox interfész SentItems metódusát adjuk vissza:

```
GetSentItemsURL =
↳ oPerson.GetInterface("IMailbox").SentItems
```



Levélküldés az Outlook Wap Accesel

További teendők

Jó néhány dolog persze megoldatlan maradt így is. Rögtön felmerülhet például, hogy valahogy kiküszöböljük a felhasználóazonosítás hiányában bárki által kihasználható levélküldözgető oldal *(send.asp)* elérését. Gondolkodhatunk azon, hogy szűrőket építsünk be a kevésbé fontos levelek elrejtésére *(az SQL lekérdezésbe nagyszerűen belekódolhatjuk a szűrő kifejezéseket)*. Nyitott probléma a biztonságos felhasználóazonosítás, a kódtáblák teljes körű és helyes kezelése... akit tehát érdekel a téma, íme egy jó kiindulási alap. Jó szórakozást!

Fülöp Miklós
mick@netacademia.net

A cikkben szereplő URL-ek:
[1] http://technet.netacademia.net/download/asp/owa/
[2] http://msdn.microsoft.com/library/en-us/wss/wss/_cdoex_cdo_for_exchange_2000_server.asp
[3] http://msdn.microsoft.com/library/en-us/ado270/htm/mdmscadoapireference.asp

.Net akadémia
(III. rész)

Az előző két részben megtekintettük az osztályok alapvető lakóit, megnéztük, hogyan kezeli a .NET a típusaink által igényelt memóriát.
Most továbbhaladunk az objektumorientált jellemzők mentén, és megnézünk néhány fejlettebb típust, amelyek nagy segítséget jelentenek áttekinthető, könnyen bővíthető és karbantartható programok írásában.

Interfészek

A mai objektumorientált rendszerekben az objektumok közötti csatolás, függőség csökkentésére az egyik legelterjedtebben alkalmazott eljárás az interfészek használata. Habár a VB6 nem volt objektumorientált nyelv, a COM komponensek használata és írása során - még ha nem is tudtunk róla - állandóan COM interfészeket definiáltunk és alkalmaztunk, de ezt a VB ügyesen elrejtette előlünk. Aki azonban akart, még VB6-ban is különválaszthatta a COM interfész definiálást és az implementációját *(ld. Implements kulcsszó)*.
A .NET Framework Class Library *(FCL)* rengeteg helyen használja ki az interfészek adta előnyöket könnyen kiegészíthető, kényelmesen használható alapkönyvtár felépítésére.
Mi takar hát az interfész fogalom az OOP világban? Az interfész egy formai és működésbeli szerződés, amelyhez minden leszármazott osztálynak kötnie kell magát. Közelebbről: az interfész definiálja mit kell csinálni, milyen metódusokat kell tudni, a leszármazott osztály pedig megvalósítja, implementálja a metódusokat, azaz tudja hogyan kell azt megvalósítani.
Gyakori például, hogy egy-egy funkcionalitáshalmazt sokféle, nagyon különböző feladatra készített osztálytól is elvárunk. A funkcionalitás konkrét megvalósítása minden osztály esetén más és más lenne, csak az azt megvalósító metódusoktól elvárt működés lenne azonos. Ezt meg lehetne oldani úgy, hogy egy-egy absztrakt alaposztályban deklaráljuk a szükséges metódusokat, majd a konkrét osztályokat leszármaztatjuk ebből, így kötelezően implementálnunk kell az alaposztály absztrakt metódusait *(mert absztraktok voltak, azaz nincs a metódusoknak törzse)*. Viszont erre a célra nem a legszerencsésebb megoldás az öröklődés, különösen akkor, ha többféle, különböző jellegű funkcionalitáshalmazt is meg kell valósítani. Ebben az esetben többszörös öröklődést kellene használni, ami számos gubanc forrása szokott lenni, éppen ezért például a .NET-ből és a JAVA-ból száműzték is.
Az absztrakt alaposztályból öröklést általában akkor szoktuk bevetni, ha a szülőosztály és a leszármazott között is-a *(a dog **is an** animal...)* kapcsolat van. Például a Kutya az egy Állat, így mindent tud, amit egy állat általában tud, azért a minden állatra közös funkciót bele lehet rakni egy alaposztályba, és ebből leszármaztatni a kutyát *(macskát, satöbbi)*. Ebben az esetben általában kihasználjuk a tagváltozók öröklődését is, hiszen az állatoknak sok olyan közös jellemzője van, amit elég az Állat osztályban *(egy helyen)* definiálni, és az összes leszármazott állat örökölni fogja azt:

```
public abstract class Allat {
  abstract public void Eszik(string kaja);
  public string Szin;  //nem abstract!
}
```

```
public class Kutya : Allat {
  public override void Eszik(string kaja) {
    Console.WriteLine("Nyam-nyam {0}-t eszek.",
                      kaja);
  }
  public string Ugat(int hanyszor) {
    string u="";
    for(int i=0;i<hanyszor;i++) u+="Vau ";
    return u;
  }
  public void Pitizik() {
    Console.WriteLine("Kolbászt, plíz!!!");
  }
}
//Teszt:
Kutya k = new Kutya();
k.Szin = "barna";
k.Eszik("répa");
k.Pitizik();
Console.WriteLine(k.Ugat(3));
Console.ReadLine();
//Kimenet:
Nyam-nyam répa-t eszek.
Kolbászt, plíz!!!
Vau Vau Vau
```

Vannak olyan állatok, amelyek képesek harcolni. Ezt a funkciót nem rakhatjuk bele az Állat alaposztályba, mert nem minden példányra jellemző a harcikedv. Azt sem mondhatjuk, hogy valamelyik leszármazott szinten vezetjük be, mert annyira egyéni ez a jellemző. Egyéni, de sokféle állat *(objektum)* tudhatja.
Mivel a különböző funkcionalitáshalmazok ráerőltetésére megfeneklünk az örökléssel *(vagy túlságosan bonyolulttá válik a modellünk)*, itt az ideje, hogy belépjenek az interfészek.

A többszörös öröklődést száműzték a .NET-ből

Formailag egy interfész deklarációja majdnem ugyanaz, mint egy osztályé *(class)*, csak nincsenek benne hozzáférést szabályzó kulcsszavak *(private, protected, stb.)*, mert minden tag automatikusan nyilvános, és nincs megírva a tagok törzse, mert azt majd egy, az interfészt implementáló osztály fogja leírni.
Lássuk hogyan lehet definiálni egy IHarcos interfészt, amelyet majd minden harcias osztály *(Kutya, Macska, bármi más nem állat osztály is)* implementálhat:

```
interface IHarcos {
  void Csip();
  void Karmol();
}
```

Konvenció szerint az interfészek nevét nagy I-vel kezdjük. Öröklessünk le egy Pitbullt a Kutya osztályból, és tegyük őt harcossá, azaz támogassa, valósítsa meg az IHarcos interfészt:

```
public class Pitbull : Kutya, IHarcos {
  public void Csip() {
    Console.WriteLine("Csípek mint állat, mert
↳ egy dög Pitbull vagyok!!!");
  }
  public void Karmol() {
    Console.WriteLine("Karmolok");
  }
}
```

A Pitbull : Kutya azt jelöli ki, hogy a Pitbull osztály örökli a Kutya osztály összes jellemzőjét *(metódus, tagváltozó, stb.)*. Ez a klasszikus öröklődés. Az IHarcos azt jelöli, hogy a Pitbull osztály implementálja az IHarcos interfészt, azaz megvalósítja az összes benne definiált metódust. A kettő nagyon különbözik, de sajnos a C# szintaksza a kettőt összemossa, emiatt különösen jó, ha tartjuk magunkat az I kezdőbetűs interfész nevekhez. Ez azért van, mert a C++ formátumához közelivé akarták tenni a nyelvet. Ebből a szempontból szerintem jobb a JAVA-s formátum: extends kulcsszó az öröklésre, implements az implementációra.
Az interfészekben kijelölt funkcióknak természetszerűleg nem lehet törzse, azt majd az implementáló osztályok írják meg.
Az interfészek öröklődhetnek egymásból, így az implementáló osztályoknak meg kell valósítani az összes interfész *(papa, nagyapa, stb.)* műveleteit is.
Egy interfész teljesen absztrakt, így tagváltozókat nem tartalmazhat, csak „metódusjellegű" tagokat: metódusokat, property definíciókat, eventeket és indexer definíciókat. *(Az indexer nagyvonalakban az a speciális elérő metódus, amivel az indexelést kijelölő [] –jelek működését lehet leírni.)*
Egy interfészt megvalósító osztálynak az interfész összes tagját meg kell valósítania. Ezt azért követelik meg a nyelvek, mert a hívók látva, hogy mi megvalósítunk egy interfészt, bátran belehívnak bármelyik implementált metódusba, így ha nem szolgáltatnánk mindegyikhez implementációt, az nagy hiba lenne.
Ha van egy macskánk, ami szintén harcos, akkor azt így definiálhatjuk:

```
class HarciMacska : Allat, IHarcos
  void IHarcos.Csip() {
    Console.WriteLine("Csípek, mert {0}
    harcicica vagyok.", Nev);
  }
  void IHarcos.Karmol() { ... }
  public override void Eszik(string kaja) { ... }
  public string Nev = "Kormos";
}
```

Látjuk, hogy az implementáció során ki lehet írni az interfész nevét is, nem csak a megvalósított metódusokét. Ezt hívjuk explicit interfész implementációnak. Ez egy nagyon furcsa jószág, mert ebben az esetben a megvalósított interfésztagok nem látszanak a megvalósító osztályra mutató referencián keresztül, mintha nem is lennének részei a megvalósító osztálynak.
A Pitbull osztálynál ezzel nem volt gond:

```
Pitbull p = new Pitbull();
p.Csip();
Csípek mint állat, mert egy dög Pitbull vagyok!!!
```

Azaz a megvalósított metódus úgy érhető el, mintha mindig is az osztály része lett volna.
Ezzel szemben a HarciMacskánál ezzel megbukunk:

```
HarciMacska m = new HarciMacska();
m.Csip(); //hiba, nincs Csip() metódus
//'HarciMacska' does not contain a definition
↳ for 'Csip'
```

A Csip metódus eléréséhez a HaziMacska osztályreferenciát át kell kasztolunk IHarcos-ra, és azon keresztül már elérhető a kívánt metódus:

```
IHarcos h = (IHarcos) m;
h.Csip();
```

Egy kicsit skizofrén az explicit implementált tagok láthatósága: az osztályreferencián keresztül nem láthatóak, így ebben ez értelemben egyéniek *(private)*, másfelől viszont az őket deklaráló interfészen keresztül kitűnően láthatóak, így ebben az értelemben publikusak.
Mikor kell ilyen kórtani esetekhez nyúlnunk? Két esetben. Ha van olyan funkcionalitás, amit szeretnénk néhány osztályunkon implementálni, de nem szeretnénk, hogy a külvilág, az osztályunkat felhasználók közvetlenül elérjék őket, akkor érdemes bevetni az explicit interfészimplementációt. Másrészt ha több interfészt implementálunk egy osztályban, akkor névütközések is lehetnek a tagok között, azaz lehet, hogy több szülő interfész is ugyanolyan nevű tagokat deklarál. Ilyenkor a kiírt interfésznév miatt egyértelmű az implementáció során, hogy melyik tagot implementáljuk, és a hívók is kénytelenek explicit kasztal jelezni a szándékukat.

Interfészalkalmazások a FCL-ban

A .NET osztálykönyvtár nagyon sok helyen alkalmazza az interfészeket. Mivel az érték szerinti típusokat definiáló struct is implementálhat interfészeket, ezért első ránézésre elég meglepő, de a még az olyan primitív típusok, mint az int is támogatnak jó pár interfészt. Például a c# int megfelelője a System.Int32, amelynek definíciója:

```
public struct Int32 : IComparable, IFormattable,
IConvertible
```

Azaz három interfészt implementál. Az IComparable-t azon osztályok vagy struktúrák valósítják meg, amelyeken lehet értelmezni sorrendet, például számokon, dátumokon, stringeken. Mivel minden osztályon vagy struktúrán másként kell értelmezni a kisebbség-nagyobbság fogalmát, ezért minden egyes típusnak külön-külön meg kell valósítani ezt az interfészt. Ez az interfész egyetlenegy metódust kér az implementálóktól:

```
int CompareTo(object obj);
```

Azaz vár egy objektumot, amelynek a típusa tipikusan ugyanaz, mint az implementáló osztályé, és kisebb, mint egyet ad vissza,

ha a paraméterként átadott típus kisebb, mint a példány értéke, 0-t, ha egyenlő és pozitív számot, ha nagyobb.
Például két integer esetén:

```
int a = 5, b=8;
Console.WriteLine(a.CompareTo(b));
-1
```

Látható, hogy nem explicit implementálja az int az IComparable-t, mert akkor nem tudtuk volna elérni a CompateTo-t, csak így:

```
IComparable i = (IComparable) a;
Console.WriteLine(i.CompareTo(b));
```

Vagy egy sorban:

```
Console.WriteLine(((IComparable)a).CompareTo(b));
```

Az IFormattable-t azok a típusok implementálják, amelyeket többféleképpen is lehet stringként, szövegként reprezentálni. Például tudjuk, hogy a dátumok szinte minden országban másként néznek ki, vagy ismerjük a tizedespont kontra tizedesvessző problémakörét a magyar területi beállítások kapcsán. A típusok kiíratásakor általában az aktuális területi beállításokat használja a .net, de ettől eltérhetünk ezen az interfészen keresztül.
Ennek is csak egy metódusa van:

```
string ToString(
        string format,
        IFormatProvider formatProvider);
```

Hogy szebb legyen az élet ez a metódus is hivatkozik egy interfészre, amelyet szintén meg kell valósítani. Ha az előbbi egészünket akarjuk megformázni, akkor például a FCLNumberFormatInfo osztály ad egy megvalósítást.

```
NumberFormatInfo fi = new NumberFormatInfo();
fi.NumberGroupSeparator = ",";
fi.NumberGroupSizes = new int [] {3,2,2};
int x = 1234567890;
```

Ezek után a formázott kiíratás:

```
Console.WriteLine(x.ToString("N", fi));
1,23,45,67,890.00
```

Az N jelöli, hogy csoportonként tagolt számkiírást szeretnénk, a második paraméter egy IFormatProvider megvalósítás. Ha mindenáron ragaszkodunk az explicit interfészeken keresztüli eléréshez, persze az is megy:

```
Console.WriteLine(
        ((IFormattable)x).ToString("N", fi));
```

Végül pár szó az IConvertible interfészről. Ez a típusok közötti konverzió kedvéért született *(például amikor egy szövegből egy számot kell kihámozni)*. A legtöbb típus explicit interfészimplementációval támogatja, azaz közvetlenül nem is érhető el, csak ha lekérünk egy IConvertible interfészt:

```
int y=8;
IConvertible yc = (IConvertible) y;
byte by = yc.ToByte(null);
```

Int32 esetén a tényleges konverzió a System.Convert osztály statikus metódusai segítségével történik, és általában mi is ezt szoktuk használni az IConvertible közvetlen hívása helyett. Máskor meg a System.Convert használja az adott típus IConvertible interfészét. Számunkra általában mindkét út nyitott, de az ajánlás szerint használjuk a Convert osztályt.

Delegatek

Egy delegate megközelítőleg nem más, mint egy típusos függvénypointer. A segítségével írhatunk olyan általános algoritmusokat, amelyeknek nem közönséges adatokat adunk át paraméterül, hanem egy delegatet, amelyen keresztül a hívott képes meghívni egy általunk szolgáltatott metódust. Azonban a CLR-beli delegatek nem olyanok mint amiket a C++-ban megszokhattunk, azaz nem egy sima pointer, aztán vagy tényleg egy olyan függvényre mutat amire a hívni szándékozó felkészült, vagy nem. Ebből nagy galibák tudnak adódni C++-ban, mert sokszor az általánosítások miatt a fordító nem tudja ellenőrizni, hogy megfelelő típusú függvénymutatót passzolunk-e át a hívottnak.
Emiatt a CLR-ben a delegatek szigorúan típusosak, azaz csak akkor adhatunk át delegaten keresztül egy metódusreferenciát, ha annak paraméterlistája és visszatérési értéke *(szignatúrája)* pontosan egyezik a hívott által elvárttal.
Lássunk egy atomfizikus példát! Tegyük fel, hogy egy atomreaktor szivattyúinak vezérlését kaptuk feladatul. Sokféle szivattyúnk van, amelyeket egy-egy osztályon keresztül érhetünk el, és mindegyiket más-más metódushívással lehet bekapcsolni. A feladat az, hogy ha a reaktor hőjét elvezető olvadt nátrium hőmérséklete 450 Celsius fok fülé megy, akkor be kell kapcsolni a hűtőszivattyúkat. A reaktoron vannak hőérzékelők, azokon keresztül lehet tudni a hőmérséklet pontos értékét. A kérdés, hogy hogyan kommunikáljanak a szivattyúvezérlő osztályok és a hőérzékelő osztály?
A nekiszaladós megoldásban az összes szivattyúosztály időnként ránéz a hőérzékelőre, és ha túl magasnak találja a leolvasott értéket, akkor bekapcsol. Általában ezt hívjuk pollozásnak. Számtalan sebből vérzik: túl sűrű lekérdezésekkel feleslegesen terheljük a hőérzékelő osztályt a szivattyúk kérdéseivel, túl ritka esetén pedig lehet, hogy később vesszük észre, hogy baj van.
A megoldás nyilvánvalóan az lehet, hogy maga a reaktor hőérzékelője értesíti a szivattyúkat a határ átlépéséről. Ehhez viszont ismernie kell az összes szivattyút, azaz hogy az egyes szivattyú osztályokban melyik metódust kell meghívni a szivattyúk elindításához. Például legyen két szivattyúvezérlő osztály az alábbi:

```
public class SzivattyuBosch {
  public void Bekapcs() {
    Console.WriteLine("Bosch elindult.");
  }
}

public class SzivattyuAKG {
  public void Indit()
  {
    Console.WriteLine("AKG elindult.");
  }
}
```

A hőérzékelő kódja:

```
public class Hoerzekelo {
  //Itt tároljuk a szivattyúk listáját
  private ArrayList szivattyuk = new ArrayList();
```

```
        private int hofok;

        public void SzivattyuHozzaad(object szi-
    vattyu) {
        szivattyuk.Add(szivattyu);
  }

  public void Meres() {
    //valahonnan (hardver)
    //elöszedi a homérsékletet
    hofok = 500;
    if (hofok > 450) GazVan();
  }

  private void GazVan() {
    //Szivattyúkat elinditani
    foreach(object sz in szivattyuk)
    {
      if (sz is SzivattyuBosch)
        ((SzivattyuBosch)sz).Bekapcs();
      if (sz is SzivattyuAKG)
        ((SzivattyuAKG)sz).Indit();
    }
  }
}
```

Az osztályokat létrehozó indító kód:

```
SzivattyuBosch b = new SzivattyuBosch();
SzivattyuAKG a = new SzivattyuAKG();
Hoerzekelo h = new Hoerzekelo();
h.SzivattyuHozzaad(a);
h.SzivattyuHozzaad(b);
h.Meres();
```

Igazából a Mérés metódust a hőérzékelő maga hívná meg belülről, periodikusan, de most az egyszerűség kedvéért kívülről hívtuk meg. Láthatjuk, hogy a megoldásunkban a Hoerzeklo osztályt felkészítettük a szivattyúk dinamikus hozzáadására, azonban csak ezt a két konkrét típust tudja meghívni. Ez tipikus példája a szorosan csatolt rendszereknek: a hívó osztálynak pontosan ismerni kell a hívandó osztályokat, azaz egy új hívandó típus esetén át kell írni a hívó osztályt. Ez magával vonja annak teljes újratesztelését, azaz jelentős idővel és költséggel jár a bővítés.
Általánosan megfogalmazva a probléma a következő: van egy osztály, akinek bizonyos belső állapotváltozásaira más osztályokat értesíteni kell, de úgy, hogy az osztálynak ne kelljen előre ismerni az értesítendő osztályok típusát. Ezt a problémát a Design Patternek világában Observer-Observable *(megfigyelő - megfigyelhető)* mintának nevezik; az állapotot tartalmazó osztályra Subject metaforával hivatkoznak, ennek változásait az Observerek figyelik meg. Annyiban egy picit sánta a pattern által sugallt kép, hogy tulajdonképpen nem az Observerek figyelik meg a Subjectet, hanem a Subject értesíti az Observereket az állapotváltozásról, bár sokszor ilyenkor az Observerek visszanyúlnak a Subjecthez további információkért.

Valahogyan csökkenteni kellene a csatolást a Subject *(Hoerzekelo)* és a két Observer *(a két szivattyú osztály)* között. És itt sietnek a segítségünkre a delegatek.
Ha megfigyeljük, mindkét szivattyút elindító metódus azonos szignatúrájú, azaz mindkettő void visszatérési értékű és egyik sem vár el paramétereket. Emiatt egy delagate segítségével egységes módon megfoghatjuk a metódusaikat:

```
public delegate void SzivattyuBekapcsolas();
```

Ez a delegate pont olyan metódusokat képes tárolni, mint amellyel a szivattyúkat be lehet kapcsolni.
A Hoerzekelo SzivattyuHozzaad metódusa változatlan, de most nem egy-egy osztályreferenciát adunk át neki, hanem a konkrét osztálypéldányok megfelelő metódusaira létrehozott delegatekét:

```
1 HoerzekeloDelegattel hd =
2   new HoerzekeloDelegattel();
3 hd.SzivattyuHozzaad(
4   new SzivattyuBekapcsolas(a.Indit));
5 hd.SzivattyuHozzaad(
6   new SzivattyuBekapcsolas(b.Bekapcs));
```

Fontos látnunk, hogy a 4 illetve 6-os sorokban nem történik meg a metódushívás, csak létrehozunk egy delegatet a megfelelő metódusokra.
Ezek után a Hoerzekelo osztálynak csak a GazVan() metódusát kell átírni, a többi változatlan marad:

```
foreach(SzivattyuBekapcsolas szbekapcsolo in
szivattyuk) {
  szbekapcsolo();
}
```

Mivel a tömbünkben most nem objektumreferenciákat, hanem delegate példányokat tárolunk, ezért a ciklusban is SzivattyuBekapcsolas *(delegate)* típusú objektumokat kapunk vissza. Egy delegaten keresztüli hívás ezek után láthatóan roppant egyszerű, egyszerűen a delegatet mint egy metódust meghívjuk. Ha lennének paraméterek, azok a delegate utáni zárójelek között lennének felsorolva.

folytatjuk...

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

A cikkben szereplő URL-ek:
[1]: Letölthető példakódok
http://technet.netacademia.net/download/net

XMLgessünk

Átjárás a relációs és az xml világ között II.

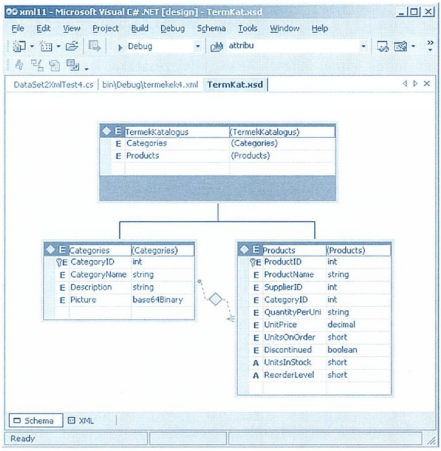
Folytatjuk a relációs adatbázisok és az xml világ közötti átjárást. Tovább boncolgatjuk a DataSet osztály szolgáltatásait, és megszemlélünk egy valódi virtuózt, az XmlDataDocument objektumot.

Típusos DataSet *(folytatás)*

Az előző alkalommal legeneráltuk a TermekKatalogus nevű típusos datasetünket *(ds)*, és feltöltöttük a Northwind adatbázis Products és Categories tábláival. Emlékeztetőül:

```
SqlConnection c = new SqlConnection("...");
SqlDataAdapter aProd = new SqlDataAdapter("SELECT *
FROM Products", c);
SqlDataAdapter aCat =
new SqlDataAdapter(
    "SELECT * FROM Categories", c);
//Ez az a típusos dataset osztályunk, amit az
// xsd.exe generált
TermekKatalogus ds = new TermekKatalogus();
//A dataset tábláinak feltöltése
aCat.Fill(ds, "Categories");
aProd.Fill(ds, "Products");
```

Hogy lássuk a séma hatását, egy kicsit átalakítottam a DataSet sémáját, a Products táblában a UnitsInStock és a ReorderLevel átment attribútumba:



A módosított sémában egyes adatbázisoszlopok átkerültek attribútumszintre („A" betű)

Az adatbázisból feltöltött típusos DataSet-ből xml kimenetet generálni már nem művészet:

```
ds.WriteXml("termekek4.xml");
```

```
<TermekKatalogus>
  <Categories>
    <CategoryID>1</CategoryID>
    <CategoryName>Beverages</CategoryName>
    <Description>Soft drinks, ...</Description>
    <Picture>FRwvAAIAAAANAA4AFA ...</Picture>
  </Categories>
  ...
  <Products UnitsInStock="0" ReorderLevel="0">
    <ProductID>5</ProductID>
    <ProductName>Chef Anton's Mix</ProductName>
    ...
  </Products>
...
```

Apró, de fontos információk:

- Az xml kimenet csak akkor lesz olyan struktúrájú, mint amit a típusos DataSet sémájában előírtunk, ha a DataAdapterek Fill() metódusaiban olyan táblaneveket adunk meg, amelyek egyeznek a megfelelő sémában definiált táblanevekkel.
- A WriteXml() metódusnak átadhatunk egy további paramétert, amellyel befolyásolhatjuk a generálódott xml formátumát. Például az XmlWriteMode.WriteSchema megadásával kérhetjük, hogy a generálódott dokumentum tartalmazzon egy beágyazott sémapéldányt is. Nem meglepő módon ez a DataSet mögötti séma lesz, azonban ez nincs összekötve a mögötte generálódó xml adatokkal, tehát így közvetlenül nem lehet feldolgoztatni egy validáló xml elemzővel.
- A Picture egy image típusú oszlop az adatbázisban, azaz nagytömegű bináris adatokat tartalmaz *(képeket)*. Mivel az xml szöveges szabvány, a DataSet BASE64 kódolással rakja bele az oszlopot a kimenetbe.

Még tovább alakíthatjuk az xml kimenetünket. A sémánk azt írta elő, hogy a gyökérelem alatt lesznek Categories és Products elemek. Ennek megfelelő xml kimenetet is kaptunk, először felsorolva az összes Categories elemet, majd jöttek a termékek elemei. Azonban mi jól tudjuk, hogy a két adathalmaz között szülő-gyermek kapcsolat van, ahol a Categories a szülő. Egy kategóriához több termék is kapcsolódhat. Ez a reláció nem igazán látszott a kimeneti xml dokumentumon, hiszen például egy csoportosított listázáshoz össze kell vadászni az egy kategóriához tartozó termékeket a CategoryID elemek mentén.

Amikor relációs adatokat képzünk le hierarchikus xml adatokká, akkor ez egy járható út. Azonban sokszor azt szeretnénk, hogy a szülőelemek közvetlenül tartalmazzák a hozzájuk tartozó gyermekelemeket, beágyazott *(nested)* gyermekelemekként. Ennek semmi akadálya sincs, csak akkor úgy kell megfogalmaznunk a DataSet sémát, hogy a gyökérelem alatt legyenek a Categories gyermekelemek, és azokban lehet tetszőleges számú Products elem. A DataSet szolgai módon az új sémának megfelelően fogja generálni a kimenetet. Az

egymásba ágyazás szabályát *(azaz, hogy a CategoryID mentén kell csoportosítani)* onnan tudja, hogy az előző részben látott módon xsd:key és xsd:keyref segítségével előírjuk az adatok kapcsolódását.

Azonban megspórolhatjuk magunknak az új séma elkészítését. Szerencsére a DataSet remekül tudja alakítgatni a sémát *(és ezen keresztül a kimeneti xml dokumentumot is)*, így az előbbi sémával is tudunk egymásba ágyazott, csoportosított kimenetet is generálni. Ehhez azt kell tudnunk, hogy a sémában leírt termék-termékcsoport kapcsolatot a DataSet Relations kollekcióján keresztül érhetjük el. Példánkban ez csak egy elemet tartalmaz, egy DataRelation típusút. Ha több tábla is lenne a DataSet-ben, vagy esetleg valamelyik tábla saját magára is hivatkozna, akkor többelemű lenne ez a kollekció. Egy relációnak több jellemzője is van, de ami minket most érdekel, az a Nested nevű jellemző. Ha kézzel, programból hozunk létre relációt két tábla között, akkor ennek értéke false, azaz nem egymásba ágyazott relációról van szó. Az előbbi sémánk is ilyen szerkezetű volt. Azonban a típusos DataSet-ünk létrehozása után nyugodtan átírhatjuk ezt a jellemzőt!

```
TermekKatalogus ds = new TermekKatalogus();
```

A sor lefutása után már ismert a DataSet számára a séma, hisz ő egy típusos DataSet. Így a relációt *(amelyet a sémában kijelölt kapcsolat miatt hozott létre a DataSet)* egy mozdulattal átalakíthatjuk beágyazottá:

```
ds.Relations[0].Nested = true;
```

Persze csak akkor szabad ilyen határozottan belecímezni a Relations kollekcióba, ha tudjuk, hogy létezik benne a számunkra fontos reláció. Összetettebb esetben végig kell gyalogolni a relációkon, és a ParentTable, ParentColumns, ChildTable, ChildColumns jellemzők alapján meg tudjuk találni a számunkra fontos kapcsolatot.
Emellett még azt is kihasználhatjuk, hogy a sémában a relációt megneveztük:

```
<xs:keyref name="CategoriesProducts"
           refer="CategoryKey">
```

Ezen a néven megtalálhatjuk a relációnkat a Relations kollekcióban is:

```
ds.Relations["CategoriesProducts"].Nested =
                                        true;
```

Miután háromféle módon beállítottuk a Nested jellemzőt, generáltassuk le az xml kimenetet a korábban látott módon:

```
<TermekKatalogus>
  <Categories>
    <CategoryID>1</CategoryID>
    <CategoryName>Beverages</CategoryName>
    ...
    <Products UnitsInStock="39" ...>
      <ProductID>1</ProductID>
      ...
    </Products>
    ...
    <Products UnitsInStock="57" ...>
      <ProductID>76</ProductID>
      ...
    </Products>
  </Categories>
  <Categories>
    <CategoryID>2</CategoryID>
    <CategoryName>Condiments</CategoryName>
...
```

Bemutatkozik az XmlDataDocument

Jelenleg ott tartunk, hogy képesek vagyunk szinte tetszőleges xml formátumra leképezni a lekérdezések eredményhalmazait, azonban az adatok szigorúan a forrásadatok által vezérelt hierarchiában és sorrendben jönnek elő. Mielőtt valaki lefitymálná ezt, szeretném jelezni, hogy ez nem csak SQL Serverrel megy, hanem bármilyen adatforrással! Mivel az OleDb... osztályokkal bármilyen adatforrásra rácsatlakozhatunk, gyakorlatilag mindenféle relációs forrásból generáltathatunk xml kimenetet. Ez igencsak jól jön régebbi vagy kevésbé fejlett adatbázisrendszereknél, amelyeket meg kell nyitni xml-en keresztül.
Az DataSet kimenete egy nagy, az xml szabályoknak megfelelő karaktersorozat, amelyet fájlba írtunk ki. Ha ezt XSLT-vel át szeretnénk transzformálni például HTML-lé, akkor egy XmlDocument vagy XPathDocument osztály segítségével értelmeztetnünk kell a generált kimenetet, azaz a fáradságos munkával generált kacsacsőröket azon nyomban kidobatjuk az xml parserrel *(értelmezővel)*, hogy az előállíthassa a memóriabeli xml fát. Azt, hogy mindez ne fájlon keresztül történjen könnyű kikerülni, mert a DataSet-nek van egy GetXml() metódusa, ami stringként adja vissza az xml kimenetet.
Na és? Ez csak fél siker. Valahogyan el kellene kerülni a közbenső szöveggé való átalakítást. Egy olyan eszköz kellene, amely anélkül képes felépíteni a memóriabeli xml objektumfát, hogy közben át kellene alakítani xml szöveggé a DataSet tartalmát. Másképpen fogalmazva a DataSet adatait Infosetként szeretnénk láttatni, anélkül, hogy közben még csak a kacsacsőrök gondolata is felmerülne. Ehhez természetesen egy olyan objektumra lenne szükség, amely képes közvetlenül olvasni a DataSet tartalmát, és azt úgy megjeleníteni a külvilág számára, mintha az InfoSet lenne. Így lehetne transzformálni a DataSet tartalmát, mintha csak az egy xml forrás lenne, lehetne XPath kifejezésekkel kiszűrni egyes részeit, satöbbi. Jöhet az XmlDataDocument!
Ő pont erre van kitalálva. Megadunk neki egy DataSet-et, és ő ezt kifelé úgy mutatja, mintha egy xml dokumentum lenne. Nem készít másolatot a DataSet DataTable-jeiből. Menet közben nem alakítja át a DataTablek sorait kacsacsőrös xml doksivá. Ehelyett hozzákapcsolja a DataTablek sorait a saját belső elemlistájához, és amikor egy-egy xml elemre vagy attribútumra hivatkozunk rajta keresztül, akkor visszanyúl az élő DataSet megfelelő táblájához, és onnan veszi az adatokat. Olyannyira élő ez a kapcsolat, hogy a DataSet bármilyen változása azonnal látható az XmlDataDocument-en keresztül, sőt, visszafelé is megy a kapcsolat, tehát ha az xml interfészen keresztül módosítunk valamit, az tulajdonképpen a DataSet módosítást jelenti!
Az egész mágia kulcsa az a jól megtervezett XmlReader, XmlWriter architektúra, amiről négy résszel ezelőtt kezdtünk el beszélni. Mivel a legelemibb író-olvasó funkcionalitáshoz szükséges metódusokat a két absztrakt alaposztály írja elő, ezért gyakorlatilag bármilyen adatforráshoz lehet írni olyan elérő osztályt, amelyen keresztül az adatok xml InfoSet-ként látszanak, azaz elemeken és attribútumokon keresztül. Mivel a bejáráshoz szükséges alapvető metódusokat és jellemzőket minden xml forrásként viselkedni akaró osztály kénytelen implementálni az alaposztályok által rákényszerített módon, ezért bármely xml-t feldolgozó osztály *(pl. XslTransform)* képes használni a leszármaztatott xml osztályokat, hisz mindig csak az alaposztályok metó-

dusait hívják, de azok mindig a konkrét megvalósító osztály metódusait érik el (mert az alaposztály metódusai virtuálisak).
Az msdn-ben **[2]** publikáltak egy cikket, amiben leírják hogy kell nem xml adatforrásokat xml-ként *(InfoSet-ként)* láttatni. Ott például a fájlrendszerhez írtak egy elérőosztályt, amelyen keresztül xml doksiként láthatóak a fájlok, könyvtárak és azok jellemzői. Az XmlDataDocument pont az ott leírt módon működik, azzal a hozzáadott plusszal, hogy figyeli a DataSet változásait is. Ehhez a DataSet-beli DataTablek változásait jelző eseményeket kapja el *(RowChanged, RowDeleted, stb.)*. Mindezen háttérinformációk nincsenek részletesen dokumentálva, azonban az Anakrino **[3]** nevű IL-ről C#-ra decompiler sokat segít a részletek megértésében :)
Lássuk hát működés közben ezt a csodaosztályt! Induljunk ki az előző példában összerakott, egymásba ágyazott elemeket tartalmazó DataSet-ből. Erre szeretnénk „ráhúzni" egy XmlDataDocument-et. Ez nagyon egyszerű lesz:

```
XmlDataDocument xd = new XmlDataDocument(ds);
```

Azaz a konstruktorban átadunk egy feltöltött DataSet referenciát, és máris megtörtént a szinkronizáció a két osztály között. Győződjünk meg erről, például úgy, hogy lefuttatunk egy XPath lekérdezést a kapott virtuális xml dokumentumon.
Ez a látszólag triviális feladat eléggé csúnya kinézetet fog kapni. A probléma forrása az, hogy a TermekKatalogus gyökérelemen van egy default névtér deklaráció:

```
<TermekKatalogus
  xmlns="http://tempuri.org/TermKat.xsd">
```

Azaz a dokumentumban az összes elem a fenti URL-rel azonosított névtérhez tartozik. Mivel eddig nem csináltunk semmit a DataSet xml kimenetével, ez nem zavart minket, most azonban úgy kell megfogalmaznunk az XPath kifejezéseinket, hogy a végrehajtó tudja, hogy mi erre a névtérre vonatkoztatjuk a kérésünket.
Az összes termékkategóriát így kérhetnénk le, ha nem lenne ez a névtérdeklaráció:

```
/TermekKatalogus/Categories/CategoryName/text()
```

Ugye milyen egyszerű lenne így az élet? A lekérdezést így pofonegyszerű lenne végrehajtani:

```
XmlNodeList n = xd.SelectNodes(
↳ "/TermekKatalogus/Categories/
↳ CategoryName/text()");
```

Azonban ez azt jelenti, hogy a null névtérben található elemekre építettünk egy lekérdezést. Ez működne, ha nem lenne a default névtérdeklaráció a doksi elején.
Így viszont egy bonyolultabb megoldást kell választanunk. Valahogyan létre kell hozzunk egy általunk megálmodott prefixszel rövidített névtér deklarációt pontosan arra az URL-re, amihez a default névteret lerögzítette a DataSet. Koncepcionálisan *(de nem a valóságban)* valahogy így nézne ez ki:

```
<TermekKatalogus
  xmlns="http://tempuri.org/TermKat.xsd"
  xmlns:sajatprefix="http://tempuri.org
↳/TermKat.xsd">
```

Azaz ugyanarra a névtérre mint ami a default névtér létrehozunk egy prefixet is, így a prefixen keresztül már explicit ki tudjuk jelölni az XPath-ban alkalmazott elemek befoglaló névterét:

```
/sajatprefix:TermekKatalogus/sajatprefix:
↳  Categories/sajatprefix:CategoryName/text()
```

Ez már rendben lesz, a kérdés az, hogy hogyan lehet utólag névtérdeklarációt létrehozni?
Az XmlDataDocument SelectNodes*()* metódusának van olyan változata is, amely második paraméterként egy XmlNamespaceManager osztályt vár el. Ez az az osztály a frameworkben, ami a névterek kezelésére szolgál. Nézzük meg a létrehozását:

```
XmlNamespaceManager nsmgr =
  new XmlNamespaceManager(xd.NameTable);
```

A konstruktorban át kell adjunk egy NameTable osztályreferenciát, amelyet az XmlDataDocument példányunktól kértünk el. Miért kellett ezt így megbonyolítani, miért kell ezzel az osztállyal is foglalkoznunk?
A .NET-es xml osztályok nagyon ki vannak élezve arra, hogy minél kevesebb memóriát használjanak, és minél gyorsabb legyen a működésük. Ezért a háttérben különböző optimalizálási technikákat vetnek be, és ezek miatt időnként bonyolultabb használni egyes funkciókat.
A NameTable egy xml dokumentumban *(XmlDocument, XmlReader, XmlDataDocument, ...)* található elem - és attribútumneveket tartalmazza táblázatos formában. Amikor a beolvasás során a forrásdokumentumban találnak egy új nevet, akkor azt berakják egy NameTable-be. Ha ezek után újra előfordul ugyanaz a név *(ami igen valószínű az ismétlődő adatok miatt)*, akkor nem foglal le neki helyet a befoglaló osztály, csak egy referenciával rámutat a NameTable megfelelő bejegyzésére. Azaz a különböző neveket mindig csak egyszer tárolják le, s ezzel jelentős mennyiségű memória spórolható meg.
De ez csak a dolog egyik fele. Ez jelentős sebességnövekedéssel is jár a beolvasott doksi feldolgozásakor, mert például két elem összehasonlításakor nem kell tényleges szövegösszehasonlítást végezni *(ami mint tudjuk költséges)*, hanem elég csak a NameTable-be mutató referenciákat összevetni. Ha egyeznek, a két elem *(attribútum)* neve azonos.
A NameTable osztályt decompilerrel átvizsgálva megfigyelhető, hogy a neveket egy tömbben tárolja. Minden stringhez egy hash értéket képez, amely a stringgel együtt tárolódik. Az azonos hash-ű elemek pluszban össze vannak láncolva, így egy string keresésekor a hash kiszámítása után már csak az elemek töredékén kell végigmennie a keresett string létezésének megállapításához.
Ez a hash-elős buli ugyan némi plusz memória felhasználással jár, de cserébe nagyon gyorsak lesznek az xml osztályok.
Ezeknek a finomságoknak örülünk, de általában nem használjuk őket tudatosan: elég hogy a háttérben így működnek.
Most már ismerjük a NameTable működését, itt az ideje használni is azt. Az XmlNamespaceManager a konstruktorában kapott egy referenciát az xml dokumentumunk NameTable-jére. Ezzel itt még nem túl sok mindent csinál, csak hozzáad egy-két, a névterek kezeléséhez szükséges

a fájlrendszerhez írtak egy elérőosztályt, amelyen keresztül xml doksiként láthatóak a fájlok, könyvtárak és azok jellemző

stringet. Ha szeretnénk új prefixeket és névtereket használni a lekérdezésekben, akkor az AddNamespace() metódust kell használnunk. Az első paraméter a prefix neve, a második a névtér teljes URN-je:

```
nsmgr.AddNamespace("alapnevter",
  "http://tempuri.org/TermKat.xsd");
```

Ez hozzáadja a prefix és URN azonosítókat a használni kívánt dokumentum NameTable-jéhez, és ő maga is lejegyzi *(egy saját veremben)*, az összetartozó párost.
És végre beérik munkánk gyümölcse, mehet a lekérdezés:

```
XmlNodeList n = xd.SelectNodes(
"/alapnevter:TermekKatalogus/alapnevter:Categories
/alapnevter:CategoryName/text()", nsmgr);
```

Látjuk, hogy második paraméterként átadjuk a névtérmenedzserünket a SelectNodes()-nak, aki most már ismeri az XPath kifejezésben használt „alapnevter" prefixet.
A metódus visszatérési értéke a kiválasztott node-ok listája, amit bejárhatunk egy ciklusban:

```
IEnumerator i = n.GetEnumerator();
while (i.MoveNext())
{
   XmlText t = (XmlText)i.Current;
   Console.WriteLine(t.Value);
}
```

Azért kasztoltam bátran az általános XmlNode-ot XmlText-re, mert az XPath kifejezésemben tényleg text node-okat válogattam le *(az elemek szöveges tartalmát)*.
Valójában az egész névteres problémát kikerülhettük volna, ha a DataSet mögötti séma targetNamespace attribútumát átírtuk volna üres stringre. Ha a kapott xml dokumentumot *(web-es megjelenítéshez)* azonnal áttranszformáljuk HTML-é, akkor ez rendben is van. Ha azonban a kimenetet szeretnénk más számára elküldeni *(B2B felhasználás)*, akkor mindenképpen adjunk egy értelmes névtérazonosítót, és ne használjuk üres névtereket.

Arcra fel!

Remélem az előbbi névtérmizériával nem vettem el senki kedvét a névterektől, de sajnos egyszer meg kell birkózni velük, aztán már nem annyira fájdalmasak.
Zárásul nézzük meg azt, amire a legtöbben fogják használni az XmlDataDocument-et: XSL transzformáció forrásául. Mivel az XmlDataDocument támogatja az IXPathNavigable interfészt, ezért nyugodtan szerepelhet az XslTransform osztály xml forrásaként.
Mi ebben a szenzációs? Az, hogy az adatok az adatbáziskiszolgálótól az adatbázis saját *(általában nagyon tömör)* protokollján jönnek át a DataSet-be, majd az XmlDataDocument képes ezt úgy átnyújtani a transzformáló osztálynak, hogy közben sehol nem alakul át a tartalom xml szöveggé! Végig xml-lel dolgozunk, de sehol egy kacsacsőr a feldolgozási láncban! Ennél hatékonyabb xml feldolgozást nehéz elképzelni.

Ha például weblapban gondolkodunk, akkor a webkiszolgáló és az adatbázis közötti kommunikáció a lehető leghatékonyabb, és még a weblapban sem kell oda-vissza alakítgatni az xml szövegeket. Ez gyökeresen ellentétes azzal, amikor például SQL Server-rel xml kimenetet generáltatunk *(kacsacsőr, redundancia megjelenik)*, ezt a weblapban azonnal értelmeztetjük egy xml parserrel *(kacsacsőr leesik)*, és abból transzformálunk. Vajon melyik a gyorsabb?
Mielőtt valaki azt a következtetést vonná le ezen információk ismeretében, hogy az SQL Server xml támogatása értelmetlen, álljon itt néhány tény:

- az SQL Server OLEDB providere képes arra, hogy egy XML kimenetű lekérdezést az eredeti formájában *(közönséges sql parancsként)* hajtassa végre az SQL Serverrel, az SQL Server saját protokollján lehozza a lekérdezés kimenetét mint hagyományos eredményhalmazt, és ügyféloldalon maga a provider alakítja ezt át xml-é. Ez a „Client Side XML" lehetőség.
- A közvetlen xml kimenet *(különösen webszerveren keresztül)* akkor jön nagyon jól, ha az SQL Serverből más rendszerek akarnak lekérdezni xml adatokat. Ilyenkor nem kell semmiféle átjátszóalkalmazás, csak ki kell használni az SQL Server beépített lehetőségeit. *(Az SQLXML .NET-beli programozásával hamarosan foglalkozunk sorozatunkban.)*

Maga a transzformáció teljesen azonos azzal, amit a kettővel ezelőtti számunkban már áttekintettünk, csak most nem XPathDocument vagy XmlDocument a forrásunk, hanem az előbbi XmlDataDocument:

```
XslTransform xsl = new XslTransform();
xsl.Load(@"..\..\trafo.xsl");
StreamWriter sw = new StreamWriter("ki.html");
xsl.Transform(xd, null, sw);
sw.Close();
```

A korábbi névtér okozta galiba végigkíséri az xslt-t is, a részletekért érdemes megnézni a példaalkalmazást az **[1]** címen.

Zárszó

A következő hónapokban egy picit szüneteltetjük a .NET xml osztályait, és előveszünk egy-két xml alapszabványt, például az xml sémát, és azokat fogjuk boncolgatni. A .NET-es xml osztályokban sokkal több lehetőség rejlik, mint amennyit ebben a négy részben be tudtam mutatni, úgyhogy még elő fog kerülni a téma.

Soczó Zsolt
Zsolt.Soczo@netacademia.net

A cikkben szereplő URL-ek:
[1]: A cikkben szereplő példakódok
http://technet.netacademia.net/download/xml
[2]: Implementing XmlReader Classes for Non-XML Data Structures and Formats
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/Custxmlread.asp?frame=true
[3]: Anakrino Download
http://www.saurik.com/net/exemplar

Microsoft Exchange 2000

Méltatlanul elfeledett termékre térünk vissza: az Exchange Serverre. Nem a felhasználók feledték el, hanem mi nem törődtünk vele ezidáig megfelelően. Most útjára indítjuk Exchange Server sorozatunkat, mely a telepítéstől kezdve végigvezet a termék használatának rejtelmein.

Nem hiszem, hogy ezt a cikket az Exchange 2000 bemutatásával kellene kezdeni, nem fogom leírni az Exchange 5.5 utáni újításokat sem, mert nem hiszem, hogy az olvasó soha nem hallott még - a Microsoft termékpalettáján csak Exchange néven ismeretes - üzenetkezelő és csoportmunka-támogató alkalmazásról. Ha mégis így volna, a tech.net magazin 2000. évi .NET különszámában található összefoglaló termékleírást, valamint a Microsoft weboldalain is - akár magyar nyelven is – fellelhető áttekintő ismertetőt tudom ajánlani. Ne feledjük, hogy a tech.net hasábjain többször volt már szó az Exchange 2000 egyes kiragadott részeiről, többször fogok utalni ezekre a cikkekre a későbbiekben.
Kezdjük a cikket gyakorlatias, a telepítés szempontjából fontos ismeretekkel. Sok olyan tulajdonsága van a terméknek, amelyet már a telepítés megkezdése előtt fontos tudni. A termékben rejlő lehetőségeket, és azok bővebb ismertetését inkább részletekben, mindig az adott területre koncentrálva említem.

Az Exchange 2000 pillérei

Az Exchange 2000 négy alapkőre építkezik. Ezek az Active Directory, a System Attendant, az Information Store, és az SMTP. A sorrend nem fontossági: bármely elem hiányzik, vagy nem működik ezek közül, az Exchange működésképtelenné válik. Nézzük röviden, mit is takarnak a fenti elemek.

System Attendant

A rendszer szolgája – röviden SA - természetesen szolgáltatásként fut az Exchange kiszolgálón, és rengeteg dolga van. Legtöbb feladata összefügg az Active Directoryval, általa léteznek és frissülnek a címlisták *(Address Lists)*, ideértve az „offline" címlistát is, rajta keresztül történnek az Exchangeből történő AD lekérdezések, továbbá az Exchange útválasztó tábláinak létrehozása, karbantartása is az ő feladata. Odafigyel a többi Exchange szolgáltatásra is, ha ő nem működik, akkor a többi szolgáltatás sem: utoljára áll le, és az elsők közt indul.
Tavalyi, 2001/7. számunkban megtalálható minden, amit erről a szolgáltatásról tudni érdemes. Mindenképp ajánlott olvasmány, ha valaki még nem olvasta volna.

Information Store Service

Az Exchange 2000-ben kétféle adatbázissal találkozunk. Az egyik a régről ismert .edb kiterjesztésű Rich Text tároló, ahova a hagyományos MAPI-s hozzáférés *(mikor az Outlookot használjuk)* esetén kerülnek az adatok, valamint van egy új ún. streaming adatbázis, amely azokat az adatokat tartalmazza, amelyek **nem** MAPI használat során *(jó példa az Outlook Web Access)* kerülnek az adatbázisba

Az adatok típusa szerint megkülönböztetjük a „Private Store"-t, és a „Public Store-t". Az első a felhasználói levelesládákat, a másik a nyilvános mappákat tartalmazza. Ez a legegyszerűbb esetben is összesen négy adatbázisfájlt jelent *(két-két .edb és .stm)*. Az Information Store Service a nevében hordozza a feladatát: ez az Exchange adatbázisait irányító szolgáltatás. *(Nem összekeverendő az IIS-el, az Internet Information Services-el!)* Egy későbbi cikkben részletesen lesz szó az Exchangehez tartozó adatbázisokról, most elég tudni azt, hogy milyen típusok léteznek.

SMTP

Teljes nevén Simple Mail Transfer Protocol: az Exchange 2000 üzenettovábbító rendszerének alapja. Az előző verziókban is jelen volt már, de akkor az MTA *(Message Transfer Agent)* volt felelős a belső üzenettovábbításért, az útválasztó folyamat meghatározásáért; az utólag telepített IMS *(Internet Mail Service)* biztosította az SMTP alapú levelezést a külvilággal. Az Exchange 2000-ben már SMTP dolgozik az összes üzenet továbbításán és az útválasztás is az ő feladata. Nincs MTA többé, nincs IMS, csak SMTP.

Az Exchange 2000 négy alapkőre építkezik. Ezek az Active Directory, a System Attendant, az Information Store, és az SMTP.

Jó cikk található az SMTP-ről a tech.net 2001. áprilisi számában, amely a NetAcademia weblapján, az **[1]** címen is megtalálható.

A kétezresek integrációja

Az Exchange szóban forgó verziója Windows 2000 környezetben működőképes csupán, mivel sok helyen és minden szinten beépül környezetébe. Legszembetűnőbb talán az Active Directoryval való integráció.
Az Exchange 2000 olyannyira beépül a Windows 2000 Active Directoryba, hogy nem is bír nélküle működni: alapfeltétel az AD megléte az Exchange telepítéshez. Az Exchange 5.5-nek még saját, különbejáratú címtáradatbázisa volt. Ennek helyét az Exchange 2000-ben teljes egészében a Windows 2000 Active Directory vette át. Az Exchangeről minden az Active Directoryban van nyilvántartva: itt vannak a felhasználók, a névjegyek, a disztribúciós listák *(olyan csoportok, amelyekhez e-mail cím is tartozik)*, valamint az Active Directory szolgáltatja a munkaállomások felé a címlistákat is.
Ezeken kívül az Exchange összes beállítása is a címtárban, közelebbről a Configuration névtérben tárolódik. Amikor a legelső

Exchange 2000 telepítésre kerül, az Exchange kibővíti az Active Directory sémáját. *(Emlékeztetőül, a séma az AD azon része, amelyben az összes objektum (felhasználók, számítógépek, csoportok, stb.) definíciója, lehetséges tulajdonságai vannak leírva.)*

A Windows 2000 szolgáltatásai közül nemcsak az Active Directory szükséges az Exchange 2000 működéséhez, hanem egy sor egyéb alapszolgáltatás is Windows 2000 alapokon nyugszik. Nézzük csak sorban.

Az Exchange 2000 biztonsági rendszere is a Windows 2000 infrastruktúrán alapszik. Mielőtt a felhasználók használhatnák az Exchanget, a Windows 2000-ben a felhasználói nevükkel és jelszavukkal azonosítani kell magukat. Csak valódi AD felhasználóknak lehet levelesládájuk az Exchange 2000-ben: külső fiók legfeljebb e-mail címmel rendelkezhet, tárolóhellyel nem. Ha valaki egyszer bejelentkezett, akkor az Exchange használatához már nem kell újra azonosítania magát, nincs újabb authentikációs kör. Az AD-ban tárolódnak az Exchange jogosultságok is *(gondolok itt az egyes mailboxokhoz tartotó felhasználói jogokra: ki tudja megnyitni az adott mailboxot és ki nem)*, vagy az Exchange rendszergazdai jogosultságokra, amelyekkel jól meghatározható, hogy melyik rendszergazda mely területeken garázdálkodhat. A biztonság témakörébe tartozik az üzenetek titkosítása is, bár ezt alapbeállításokkal nem használja az Exchange 2000, de a lehetősége megvan. Ezekhez szintén a Windows 2000 szolgáltatások adják az alapot, gondoljunk pl. az Ipsec-re. A levelek hitelesítéséhez és titkosításához a Key Management Server használatos, amely ugyan az Exchange része, azonban a hozzá szükséges bizonyítványok és kulcsok a Windows 2000 egy eleméből, a Certificate Services-ből is előállíthatók. Egy újabb pont az összefonódásra.

Ezen kívül ott vannak a Windows 2000 Internet Information Services *(IIS)* komponensei is. Az IIS azáltal, hogy egy sor - Interneten használatos - protokoll használatát támogatja, lehetővé teszi, hogy az ügyfelek ne csak MAPI-n keresztül, mondjuk Outlookból tudják elérni üzeneteiket, hanem például POP3-mal, vagy IMAP4-gyel is - vagy akár HTTP-n keresztül. Az üzenetforgalmat biztosító SMTP és NNTP szolgáltatás is az IIS része. Az NNTP-t azért emelném ki, mert az alapból nem települ az IIS-sel: ezt bizony külön fel kell rakni! Ne feledkezzünk meg a Windows 2000 Active Directory működéséhez nélkülözhetetlen DNS szolgáltatásról sem. Természetesen az Exchange 2000 is DNS-t használ névfeloldáshoz, bár az nem feltétel, hogy a DNS szolgáltatást Windows 2000 kiszolgáló adja.

A levelek hitelesítéséhez és titkosításához a Key Management Server használatos

Miért is jó ez a nagy összefonódás a két termék közt? Gondoljuk végig! A közös címtár egyszerűbb felügyeletet tesz lehetővé, ráadásul az ehhez szükséges eszközök összehozhatók egy jól kialakított MMC konzolba. A jogosultságok egységes kezelése szintén megkönnyíthetik a rendszergazdák dolgát. Például a Windows 2000 csoportokat a jogosultságok kezelése mellett használhatjuk disztribúciós célokra is. Az Exchange és az AD replikációja is teljesen összefonódott. Az előző verziókban az Exchange beállításainak, változásainak replikációja objektumalapú volt, az AD replikáció viszont tulajdonság *(attribútum)* alapú, így kisebb a replikációval kapcsolatos hálózati forgalom.

Általában elmondható, ha valamit csak egyszer, egy helyen kell beállítani, és azt mind a két rendszer használni tudja, nagy előny mindkét félnek, bár a nagy egymásrautaltság azt is jelenti, hogy egyik sem tud a másik nélkül teljesértékűen működni – ez leginkább úgy igaz, hogy az Exchange 2000 nincs Windows 2000 nélkül, viszont a Windows 2000 vígan boldogul az Exchange 2000 nélkül...

Az alábbiakban összefoglaltam, milyen módon fonódik össze a Windows 2000 és az Exchange 2000:

- Az Active Directory szolgáltatja a címlistákat az ügyfeleknek
- Az Exchange teljes nyilvántartása, beállításai az AD-ban tárolódnak
- Az Exchange a Windows 2000 SMTP, NNTP, HTTP szolgáltatásait használja
- Akárcsak a Windows 2000, az Exchange is a DNS-t használja névfeloldásra.

Pár szóban az ügyfelekről

Sokféle ügyfélprogramot, és hálózati protokollt használhatunk az Exchange adatbázisokban levő adatok elérésére. Most csak röviden szólnék ezekről a lehetőségekről, a sorozatban egy egész cikket szentelünk majd ennek a témának. Közelítsük a dolgot most a kommunikációra használt protokollok oldaláról. Ezek szerint a következőket tudjuk Exchange 2000-rel használni.

MAPI *(Messaging Application Programming Interface)* – ez egy redmondi fejlesztésű, elég régi protokoll, már az előző Exchange verziók is ezt használták az ügyfelek és a kiszolgáló közti kommunikációra. Pl. az Outlook vállalati vagy munkacsoport *(Corporate or Workgroup)* telepítésével használjuk ezt a protokollt. Öregecske kecske, de mind a mai napig a MAPI az egyetlen teljeskörű szolgáltatást nyújtó protokoll az Exchange választékában.

POP3 *(Post Office Protocol v3)* - ennek segítségével csak letöltögetni tudjuk a leveleket egy Exchange kiszolgálóról, levélküldésre ez esetben az SMTP-t használjuk. Nagyjából minden levelező ügyfél támogatja ezt a fajta kommunikációt. *(Az Outlook Internet módban telepítve, vagy az Outlook Expressz, Netscape, Eudora, PMail, Pine, stb.)*. Levelesládánk sok-sok ügyes mappája közül egyes egyedül az Inboxot éri el.

IMAP4 *(Internet Message Access Protocol)* - akárcsak a POP3, az IMAP is csak egyirányú kommunikációra képes, bár azt másként teszi, pl. egyszerre több mappát tud kezelni, míg a POP3 csak egyet. A legtöbb levelezésre használt ügyfél program támogatja. Ezzel már el lehet érni a kiszolgálón lévő összes mappát, de valamennyit e-mail tárolónak érzékeli az IMAP4 ügyfél - a naptár és a névjegyek egészen vicces látványt nyújt ilyen szemüvegen keresztül szemlélve.

NNTP *(Network News Transfer Protocol)* - ez is egyirányba viszi csak az üzeneteket, ez a protokoll használatos a hírcsoportok *(newsgroupok)* olvasgatására. Az kiszolgálóoldalon az Exchange 2000 tud NNTP kiszolgáló lenni, míg az Outlook Express vagy az Outlook Internetes telepítése is tud NNTP ügyfél lenni - de természetesen van egy rakás hírolvasgató program is forgalomban.

SMTP *(Simple Mail Transfer Protocol)* - ha az előző három protokollt használjuk a levelek lehúzogatására, akkor az SMTP-t kell használni arra, hogy mi is tudjunk válaszolni. Nemcsak ügyfél és kiszolgáló közt megy a levelezés SMTP-vel, hanem a levelező kiszolgálók egymással is SMTP-vel, illetve annak egy fejlesztett változatával, ESMTP-vel kommunikálnak.

HTTP - tipikus példa erre az Outlook Web Access segítségével történő levelezés. Amikor mi egy böngészőből kapcsolódunk az OWA-n keresztül az Exchange kiszolgálóhoz, akkor HTTP, vagy sokszor titkosítva, HTTPS protokollon megy a kommunikáció ügyfél és kiszolgáló közt.

LDAP - *(Lightweight Directory Access Protocol)* az Exchange ennek segítségével kapcsolódik az Active Directoryhoz. Az ügyfelek ugyancsak LDAP-pal kapják meg a címlistát, amikor Outlookot *(Internet telepítésben)*, vagy Outlook Expresst használunk. A Netscape levelezőprogramja is képes LDAP-ot használni.

EXIFS *(Exchange Installable File System)* - ez a „protokoll" MS fejlesztés, ennek segítségével úgy tudjuk elérni az adatokat az Exchange kiszolgálón, mintha az egy fájlkiszolgáló lenne; például közvetlenül a Wordből tudunk doksit az Exchange adatbázisba menteni.

egy Active Directory erdőben (forestben) csak egy Exchange szervezet (organization) létezhet

Melyik Exchange jó nekünk?

Miért, olyan sokféle van? Az Exchange 2000-ből három változat kapható, hogy mindenki megtalálhassa a pénztárcájának és igényeinek leginkább megfelelő kiszerelést.

Microsoft Exchange 2000 Server *(ES)*

Olyan esetben érdemes ezt a terméket választani, ha megelégszünk azzal, hogy egyetlen logikai tárolóban *(Store)*, maximum 16Gb-nyi adatot tudunk használni. Ezzel a változattal nem tudunk elosztott *(Front-end/Back-end)* konfigurációt megvalósítani, Chat kiszolgálót építeni, és nem tudjuk fürtön sem használni. Nincs benne X.400 csatoló sem. Hát akkor mit lehet, mi van? Az alábbi táblázat segít ebben:

Funkció	ES	EES	ECS
AD integráció	van	van	
Web Storage rendszer	van	van	
Instant Messaging	van	van	
POP3, IMAP4, SMTP, NNTP	van	van	
HTTP OWA	van	van	
MSMail, cc:Mail, Notes/Domino, GroupWise csatolók	van	van	
Chat		van	
X.400 csatoló		van	
Front-end/Back-end támogatás		van	
Korlátlan adatbázis méret		van	
Több Store egy kiszolgálón		van	
Windows Clustering		van	
Conferencing			van

Microsoft Exchange 2000 Enterprise Server *(EES)*

Olyan helyeken érdemes használni, ahol - mondjuk - fürtöt szeretnénk építeni, vagy ahol szeretnénk elkülöníteni funkciókat egymástól, esetleg nem elég 16GB az Exchange adatbázisoknak, illetve ahol több független adatbáziscsoportot *(Store)* szeretnénk létrehozni.

Microsoft Exchange 2000 Conferencing Server *(ECS)*

Egyetlen, de összetett funkciója van, az adat-, hang- és videokonferencia-szolgáltatás. Az Exchange másik két típusával együttműködve képes mindezt megvalósítani. Multimédiás konferencia szolgáltatáshoz ügyfélprogramként NetMeetinget használunk, ami természetesen jár hozzá. *(Tudta-e, hogy a NetMeeting és a Terminal Services ügyfélprogram ugyanarra a T.120-as protokollcsaládra építkezik? Bizony, rokonok.)*

A telepítés előkészítése

Van néhány alapvető tulajdonság amelyet nem tudunk áthágni. Először is, egy Active Directory erdőben *(forestben)* csak egy Exchange szervezet *(organization)* létezhet. Csak egy, nem több. Tartsuk ezt mindig szem előtt!

Másodszor, csak olyan Windows 2000 kiszolgálóra tudunk Exchanget telepíteni, amelyik egy Active Directory része. Elég ha csak tag a tartományban, nem kell tartományvezérlőnek lennie, de annak sincs semmi akadálya, hogy DC-re telepítsük az Exchange szolgáltatást.

Az IIS megléte az adott gépen szintén feltétele az Exchange telepítésnek; az SMTP és az NNTP is telepítve kell legyen az alapszolgáltatások mellett. Az Exchange telepítéskor felturbózza az IIS-ben fellelhető SMTP és HTTP szolgáltatást, valamint a POP3 és az IMAP4 protokollt hozzáadja az IIS-hez.

A leendő Exchange kiszolgálóra a telepítés előtt minimum az első Service Packot fel kell rakni, javaslom azonban inkább rögtön az SP2-t! A telepítés elbukhat, ha a DNS illetve maga az Active Directory nem működik hibátlanul, ezért érdemes a DNS és az AD működését ellenőrizni. A Windows 2000 Support Tools csomagban található NLTEST segítségével ugyanazokat ellenőrizhetjük az AD és a DNS működéséről, amit majd a telepítés során az Exchange is megtesz, ugyanazokkal az API hívásokkal:

3 kapcsolóját is érdemes használni. Futassuk az nltest programot a leendő Exchange kiszolgálóról.

Először:

```
C:\>nltest /dsgetsite
Default-First-Site-Name
The command completed successfully
```

A Site ellenőrzése

Ha a fenti parancs valami hibát ad, akkor érdemes körülnézni a telephely *(site)* és a hozzá tartozó alhálózatok (subnet) beállításainál az AD Sites and Servicesben.

Másodszor: futtassuk az nltest /dsgetdc:<domain név> parancsot, például így:

```
C:\>nltest /dsgetdc:hell
           DC: \\DC1
      Address: \\10.10.10.10
    Dom Guid: b2454a7c-ff69-4dfc-8fb0-4f94404a8770
    Dom Name: HELL
 Forest Name: hell.net
Dc Site Name: Default-First-Site-Name
Our Site Name: Default-First-Site-Name
        Flags: PDC GC DS LDAP KDC TIMESERV
               WRITABLE DNS_FOREST CLOSE_SITE
The command completed successfully
```

A domain és a forest ellenőrzése

Ezzel vagy a legközelebbi tartományvezérlőről kapunk információkat, vagy ha történetesen maga a gép tartományvezérlő, akkor arról fognak megjelenni az adatok. A Flags kezdetű sor talán a legérdekesebb, abban van leírva az adott gép összes tartománybeli funkciója. Könnyen ki lehet találni mi mit is jelent. Végül érdemes az nltest /dclist:<domain név> parancsot is futtatni, ami visszaadja a megadott tartomány összes vezérlőjének nevét, valahogy így:

```
C:\>nltest /dclist:hell
Get list of DCs in domain 'hell' from '\\DC1'.
DC1.hell.net [PDC] [DS] Site: Default-First-Site-Name
The command completed successfully
```

A tartományvezérlők listája

Ezen kívül használhatjuk az nslookup-ot is a DNS helyes működésének tesztelésére, az SRV bejegyzések ellenőrzésére.

Ha igaz az, hogy a hibák 130%-a a DNS-ben keresendő, és mi a fenti parancsok használata során nem kapunk vissza hibaüzeneteket, akkor szinte biztosak lehetünk benne, hogy a telepítő nem fog elbukni DNS vagy az AD nem megfelelő működése miatt.

Telepítés

Vegyük azt az egyszerű esetet, hogy van egy Windows 2000 Active Directory környezetünk, még soha nem használtunk Exchanget, de most úgy döntünk, erre van nekünk szükségünk, és telepítjük a tartomány egy gépére. *(Persze a legtöbb esetben az Exchange 5.5 migrációja okoz nagy fejtörést, bár maga a telepítés akkor is igen hasonló: több lépésből áll, némileg bonyolítja a helyzetet az Active Directory Connector, amely arra való, hogy az Exchange 5.5 Directory információkat összehozzuk az Active Directoryval.)*

Még mindig nem ugorhatunk neki a setup.exe-nek, mert előtte fel kell készíteni az AD-t arra, hogy be tudja fogadni az Exchange sajátos bejegyzéseit. Mit jelent ez? Bár az Exchange 2000 teljesen integrált az AD-ba, ugyanakkor az AD helyből nem rendelkezik minden objektummal ami kell az Exchangehez; egyes objektumok léteznek ugyan, de kevesebb tulajdonsággal rendelkeznek, mint kellene. Ahhoz, hogy az Exchange otthon érezze magát, elő kell készíteni az AD sémát, és még pár apróságot is el kell követni. Ehhez tudjuk használni a setup.exe /forestprep és /domainprep kapcsolóját.

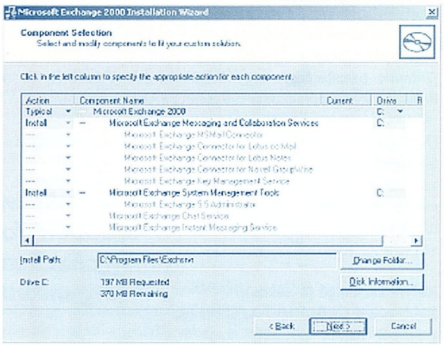
A tech.net magazin 2001/5. számában egy egész cikk szól arról, mit hogy csinálnak a fenti parancsok, mikor kell és mikor nem kell használni őket. Ez a cikk a **[2]** címen is megtalálható.

Most csak annyit, hogy ezeket külön kell futtatni a telepítés előtt, vagy - bizonyos esetekben - maga a telepítés megcsinálja a szükséges AD módosításokat.

Az első Exchange 2000 kiszolgáló telepítéséhez a következő jogokkal rendelkező felhasználói fiók szükségeltetik:

- A leendő Exchange kiszolgálón az Administrators csoport tagja legyen *(ne feledjük, hogy a Domain Admins csoport helyből tagja a helyi Administrators csoportnak)*
- Exchange Full Administrator szerepkörrel rendelkező account legyen – ez a leghatalmasabb szerep ami létezik Exchange vonatkozásában, és a forestprep során lehet meghatározni ezt az felhasználót vagy csoportot. Értelemszerűen, ha forestprep során felhasználót adtunk meg akkor annak nevében kell belépni a telepítés előtt, ha csoportot adtunk meg, akkor ennek a csoportnak legyen tagja az a felhasználó, akinek nevében futtatjuk a telepítőt.

Ha mindent jól előkészítettünk, akkor maga a telepítő varázsló nem kérdez sokat, nagyjából annyit, hogy mely összetevőket és hova szeretnénk telepíteni.



A összetevők közt itt válogathatunk

Érdemes néhány szót ejteni arról, mi történik a háttérben, amikor a bitkolbászok meg-megtorpanva növögetnek.

A telepítőnek négy jól elkülöníthető szakasza van.

1. Az első, amikor a rendszergazda kiválogatja, mit is szeretne telepíteni: ez tulajdonképpen a varázsló futtatása. Megjegyzem, hogy mielőtt válogathatunk az összetevők közt, elvégez a telepítő egy sor ellenőrzést, az AD-ra, DNS-re, valamint az aktuális felhasználóra vonatkozóan, továbbá meglevő Exchange után is kutat. Csak akkor tudjuk a telepítést folytatni, ha minden rendben van.
2. Miután útjára indítottuk a varázslót, a telepítő leállítja az összes olyan szolgáltatást, aminek köze van az Exchangehez. Ilyen például az SMTP, az NNTP, a Windows Management, a Licence Logging, a Web kiszolgáló, az IISAdmin stb.
3. Harmadik lépésben következik a szükséges könyvtárak és fájlok létrehozása, másolása a CD-ről, registrymódosítások és az új szolgáltatások telepítése.
4. Végül a telepítő elindítja az összes Exchange 2000-rel összefüggő szolgáltatást.

Ne türelmetlenkedjünk a telepítés során, egy fél órát is elmatathat a telepítéssel a kiszolgáló. Én nem csodálom, rengeteg dolga van. Aki nem hiszi járjon utána, ehhez ajánlom a **[3]** weboldalt, 11+97 pontban van összeszedve mi minden történik a telepítés során.

Dorner Csilla
MCSE

A cikkben szereplő URL-ek:
[1] http://technet.netacademia.net/2001/apr/
[2] http://technet.netacademia.net/2001/maj/
[3] http://www.microsoft.com/exchange/techinfo/reskit.htm

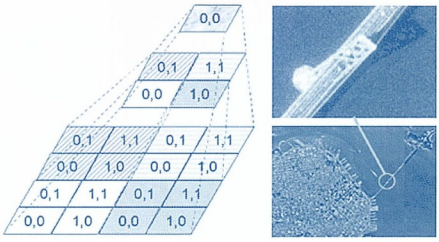
Aki kíváncsi ...

... az lehet, hogy tényleg nagyon hamar megöregszik, különösen akkor, ha követi a Microsoft egyik kutatója, Tom Barclay példáját. Ő arra gondolt, vajon megoldható-e egy terabájtnyi adat kezelése úgy, hogy azt a lassú kommunikációs vonalakon szörföző weblakók is könnyen használhassák? Tom költőívé minősítette a kérdést, és ha nekem nem hiszik el, nézzék meg a saját szemükkel is az Egyesült Államok geológiai felméréseinek *(U.S. Geographical Survey – USGS)* légifelvételeit! **[1]**

Mostanában jómagam is foglalkozom *(a kelleténél talán többet is)* a digitális fényképezéssel, így hadd vegyek ebből, a cikk témájához meglehetősen közel álló világból egy analógiát, amelynél egyszerűbben nem lehetne elmondani, mit csinált Tom az adatbázis tartalmát alkotó képekkel. Mi a teendő akkor, ha a látvány „körbevesz"? Sajnos ezen még a legjobb széles látószögű objektív sem segít, de körbefordulva készíthetünk több képet, amelyeket egy képfeldolgozó programmal össze tudunk illeszteni egyetlen képpé. A lényeg nagyjából ennyi, de merüljünk kicsit a felszín alá.

A Tom ötletéből kihajtó projekt eredeti célja a Windows 2000 Server és az SQL Server megbízhatóságának és skálázhatóságának bemutatása volt. Az elfogulatlannak éppen nem mondható állítás szerint a .NET rendszer és az XML bizonyult a legmegfelelőbbnek a terve kivitelezéséhez – ezt a tények is alátámasztják. A webes TerraService-hez API-t is készítettek, amit felhasználva a MapShots nevű cég már el is készítette a TerraFetch nevű ügyfélprogramot **[2]**.

Következzenek most a platformfüggetlen részletek. Az USGS-től kapott felvételek nagysága egyenként 50–1500 MB, felbontásuk 1 méter/pixel *(ez az alapfelbontás)*. Első lépésben kiszámítják a képek pozícióját, majd – az átfedéseket is figyelembevéve – nagyobb képekké állítják össze azokat, JPEG formátumra alakítják, és 200x200 pixeles képelemekre szabdalják. Ezt követően elkészítik a kisebb felbontású, nagyobb területet átfogó képelemeket is, végül mindegyiket különféle azonosítókkal jelölik és tárolják az adatbázisban.



A különféle képfelbontások elkészítése

Mik ezek az azonosítók? A különféle képfelbontásokat *(S: Scale)* a webes felhasználás megköveteli. Egy adott felbontású képből rosszabbat az ábra szerint úgy készíthetünk, hogy minden négy pixelből csak egyet veszünk, az utolsó pixelnél *(legrosszabb felbontás)* pedig átlagolunk. Eszerint a felbontás a 2 hatványai szerint csökken: **1**, 2, 4, 8, 16, **32**, 64, 128 méter/pixel. Tovább nincs, hiszen a 200x200 pixeles darabokból ennyi telik. A vastagon szedett felbontásokat az ábra jobb oldalán látható képek illusztrálják. A különböző forrásokból származó képek más-más típusúak lesznek az adatbáziban is *(T: Theme)*. Nem kezelhetők egyformán például a légifelvételek, az űrfelvételek és a rajzok. Külön kategóriákat képeznek továbbá a gömb felszínét síkra leképező matematikai módszerek is. Mivel nem csak egy nagy felszeletelt képünk van, a képelemek egy további azonosítót kapnak a hovatartozásuk meghatározására *(Z: Zone)*. Ezenkívül tudnunk kell még a kép bal alsó sarkának a teljes képhez viszonyított X és Y koordinátáit. Végül, a hatodik adat mondja meg, hogy mely képelemek jeleníthetők meg ugyanazon a weboldalon *(W)*. Ez a hat adat egyértelműen leír egy képelemet, és ha tudjuk mit keresünk, akár önállóan is összeállíthatjuk a lekéréséhez szükséges URL-t, a **[4]** szerkezete szerint.

Látszik, hogy az adatbázis felépítésénél a lehető legegyszerűbb megoldásra törekedtek. De akkor végülis mi ebben a nagy durranás? Egyrészt az adatmennyiség: az Interneten manapság elérhető online adatbázisok méretének ez 10–100-szorosa. Másrészt a kivitelezés ideje: a TerraService webszolgáltatás kódjának velejét például Tom néhány nap alatt készítette el. Tom így anekdotázott erről: egy szakértőt küldtek hozzá, hogy felvilágosítsa a .NET-ről. Úgy érezte, az XML a leggyengébb pontja, ezért vett egy könyvet, hogy ne legyen teljesen lámer. A szakértő, mikor meglátta a könyvet az asztalán, fogta, és a kukába hajította, majd annyit tett hozzá, hogy erre nem lesz szükség, mert ami a könyben van, azt a .NET tudja, és elvégzi helyettük. A projekt mellesleg 1996 végén indult, 1997 májusában mutatták be az első prototípust, a webhely pedig 1998 júniusában nyitotta meg kapuit.

Mit sem ér azonban az olyan műszaki alkotás, ami nem érdekli a nagyközönséget, ezért szólni kell arról is, hogy a TerraService nem várt sikert aratott. Eddig körülbelül 50 milliónan látogatták meg az oldalt, és más szervezetek is jelezték az igényüket arra, hogy a TerraService-t integrálják saját szolgáltatásaik közé. Ennek hatására csiszoltak a rendszerbeállításokon, így a Terra Service most naponta 300 ezer felhasználót képes kiszolgálni. Ez önmagában persze nem jelent komoly alkalmazást, de az már igen, hogy a farmerek információkat kapnak a talaj minőségéről, és így javíthatják a termelékenységüket, vagy a környezetvédők illegális szemétlerakó-helyeket kutathatnak fel egy számítógép segítségével stb.

A nagyon kíváncsiak további részleteket találnak a **[3]** címen. Jó szórakozást!

Zacco

A cikkben szereplő URL-ek:

[1]http://terraservice.net/ts4/about.asp?W=0
[2]http://www.mapshots.com/downloads/TerraFetch.asp
[3]http://terraserver.homeadvisor.msn.co/terra_story_images.asp
[4]http://terraserver.microsoft.com/image.asp?T=1&S=10&X=2342&Y=10368&Z=10&W=1

Következő számunk tartalmából

XMLgessünk

XML sorozatunkban lépten-nyomon használtuk az XML sémát, láttuk, hogy rengeteg technológia épül rá. A következő részben megnézzük milyen sémák vannak a nagyvilágban, mire használható az XML séma, hogyan épül fel, hogyan kell írni és milyen segédeszközeink vannak a sémával kapcsolatos fejlesztésekhez.

.NET Academia

Valójában sokkal többet tudnak a delegate-ek mint amit ebben a számban felvázoltuk, például képesek sok metódusreferencia tárolására és automatikus meghívására is. Majd miután kiteljesítjük a korábban megkezdett atomreaktoros példánkat rájövünk, hogy az ott végzett munkánk jelentős részét megspórolhatjuk, ha felhasználunk egy, a delegate-ekre épülő technológiát: a .NET beépített eseményeit.

.NET

Valójában sokkal többet tudnak a delegate-ek mint amit ebben a számban felvázoltuk, például képesek sok metódusreferencia tárolására és automatikus meghívására is. Majd miután kiteljesítjük a korábban megkezdett atomreaktoros példánkat rájövünk, hogy az ott végzett munkánk jelentős részét megspórolhatjuk, ha felhasználunk egy, a delegate-ekre épülő technológiát: a .NET beépített eseményeit.

Exchange

Áprilisban először az exchange felügyeleti eszközeivel ismerkedünk, valamint megnézzük az Active Directory Users and Computers snap-inbe integrált Exchange varázslókat, majd sorra vesszük a felhasználók tulajdonságait. Ugyanebben a lapszámban megismerkedünk az Exchange Web Storage Sytem programozásával is!

Research

Egy átlagos kaliberű, komplett asztali számítógép ma megvásárolható mintegy 150 ezer Ft-ért. A legtöbb esetben - amikor egyáltalán be van kapcsolva - szövegszerkesztésre, játékra, internetezésre stb. használjuk, miközben a merevlemeznek a legjobb esetben is csak töredékét tudjuk értelmes adatokkal feltölteni. Bob Metcalf 90%-ra saccolja a parlagon hagyott hardvererőforrások arányát. Nyugodtan higgyük el, és nyugodjunk bele, hogy 135 ezer Ft-ot kidobtunk az ablakon. Bár arra senkit sem bíztatok, hogy várjon rá, a világháló horizontján már felbukkant a megoldást hozó világgép!

Dinamikus DNS 2. rész: Secure DDNS

Folytatjuk a Windows 2000 dinamikus DNS protokoll bemutatását: nem biztos hogy egészséges, ha bárki csak úgy frissítgetheti cégünk DNS zónájának tartalmát. Rend a lelke mindennek: ezúttal a biztonságos DDNS frissítés *(Secure DDNS)* kerül terítékre.

Ingyenes hálózatfigyelő: Ethereal

Magazinunkban hónapokon keresztül futott Network Monitor sorozatunk. A teljes funkcionalitású Network Monitor-hoz azonban *(legalábbis hivatalosan)* nem mindenki fér hozzá. Most bemutatunk egy ingyenes, teljeskörű hálózatanalizátor programot: az Ethereal bizonyos területeken még jobb is, mint a NetMon!

infoBYTE

INFOKOMMUNIKÁCIÓ
A távközlési piac liberalizálása és a mobiltelefóniában várható generációváltás érdekegyeztetési törekvéseiről tudósít ez a rovat.

CÍMLAPSZTORI
A hónap vezető cikke új technológiákról, megoldásokról.

NEMZETKÖZI SZEMLE
Külfödi hírek, kitekintés.

E-KORMÁNYZAT
Az Informatikai Kormánybizotsság 2001– 2002-ben összesen 36 különféle programot koordinál. Az információs társadalom kiépítésének lépcsőfokai ezek.

INFORUM
Az Inforum célja, hogy párbeszédet folytasson a szakma és a kormányzat között. Aktív szerepet vállalt a szerzői jogi, az egységes hírközlési és az elektronikus kereskedelmi törvény megalkotásában.

EU-INFORMATIKA
Ebben a rovatban nem elsősorban a technológiára, hanem a megvalósult projektekre, az EU-kompatibilitás kérdéskörére, pályázatokra koncentrálunk.

CIO.INFOBYTE.HU
Az infopen.hu webmagazin és az infoBYTE közös rovata, kifejezetten IT vezetők számára. A rovat egy aktív CIO-val készült átfogó interjúval indul, amelyet stratégiai jellegű technológiai áttekintések, szakcikkek követnek. A rovat hangsúlyos részei a vállalati IT megoldásokat bemutató esettanulmányok, de interjúk, konferenciatudósítások formájában a piac meghatározó megoldásszállító cégeinek üzleti és termékstratégiájának bemutatása is helyet kap.

KARRIER
Tapasztalatok szerint három-négy évente változtatunk mi, informatikusok állást, szakmát vagy szakirányt, és ez a szám a jövő évtizedekben valószínűleg csökkenni fog.

KONZOL ELŐTT
E rovatunkban olvasóink nevében és helyett Novell szakértőket kérdez a szerző.

DR. WATSON
A NetAcademia-féle mélyvíz-tanfolyamokra iratkozhatnak be azon olvasóink, akik Dr. Watson nyomában járnak.

MÉRLEG
Hardver- és szoftvertesztek röviden, velősen.

PROCESSZOR
Sokakat érdeklő CPU-újdonságok mélységei a fejlesztéshez közel álló szakértők tollából.

Kérjen mintapéldányt: minta@infobyte.hu

PANNON SUPPORT RENDSZERHÁZ Kft.
Tel.: 269-2233, 269-2797
Bp 1055. Honvéd u. 40. Fsz.8. F: 269-3058
Tel.: 382-0313, 382-0314
Bp 1119. Etele út 10. Fsz.1. F: 204-9292
Info@psr.hu
IBM Business Partner
Microsoft CERTIFIED Partner

Komplett, korszerű kisvállalati megoldás!

IBM xSeries 200 Server (107C252)

Inter Pentium PIII866 Mhz
128MByte RAM (max. 1.5 GB)
18.2 Gbyte SCSI HDD
Adaptec 29160LP SCSI vezérlő
FDD, CD 10/100 eth, 3 év garancia

MS Small Business Server 2000

Windows 2000 Server Stendard Edition
Exchange 2000 Server,
SQL Server 2000, ISA Server,
Health Monitor, Managment Console,
Windows FAX és Modem megosztási,
szolgáltatás

Munkaállomások

Válasszon!

IBM NetVista A21 (PBD38HN)

Inter Pentium C900 Mhz, 128MB RAM
20 Gbyte HDD, CD, 10/100 ethernet
Ms Windows 2000 Pro. Magyar,
3 év garancia

IBM NetVista A22p (PDD72HN)

Inter Pentium IV 1,5Ghz 128MB RAM
20 Gbyte HDD, CD, 10/100 eth
Ms Windows 2000 Pro. Magyar,
3 év garancia

Cel.-os munkaállomásokkal: = **1.419.800,-***
P4-es munkaállomásokkal: **2.490.400,-***

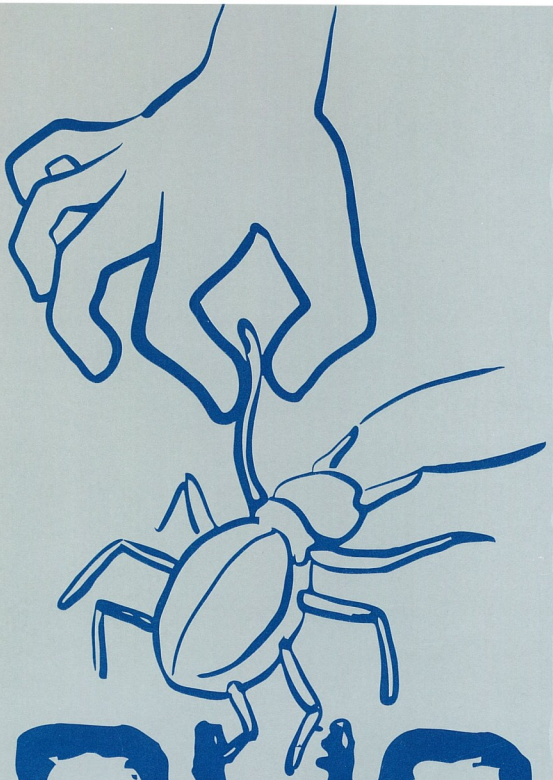
Cel.-os munkaállomásokkal: = **1.849.800,-***
P4-es munkaállomásokkal: **3.333.110,-***

***A felüntetett ár tartalmazza:**
- 1db IBM xSeries 200Server; MS Small Business Server 2000
- 5 felhasználó számára a szerver hozzáférést
- 5db IBM munkaállomást (operációs rendszerrel, monitorral, billentyűzettel, egérrel)

Igény esetén a rendszer megvásárlásához, lízing konstrukció kialakításában is segítünk!
Január és február hónapban minden 100e Ft feletti Microsoft és IBM terméket vásárlók között
IBM WorPad-et sorsolunk ki!
Áraink a 25%-os ÁFA-t nem tartalmazzák! Az árváltozás jogát fenntartjuk!

Nyerjen egy organizert!
Töltse ki az alábbi táblázatot és küldje vissza postán vagy e-mailen a kért adatokat.
A visszaküldők között sorsolunk egy Palm m100 -at! A megadott információkat bizalmasan kezeljük!

Cégnév:
Kapcsolattartó:
Beosztás: Fax:
Telefon:
E-mail cím:



BUG hunter

Legyen Ön is BUG Hunter! A hivatalos egyenruhát képező speciális vadász trikó kiérdemléséhez mindössze annyit kell tennie, hogy az Ön által felfedezett BUG-okat levadássza! Hogy hol találhatja Őket? Ismeri a természetüket, természetesen bárhol a magazinban. A vadászat eredményéről értesítsen minket az info@netacademia.net e-mail címen, hogy mielőbb elküldhessük Önnek a hivatalos egyenruhát.

Szerencsés vadászatot!

NetACADEMIA

A LEGJOBBAKAT TANÍTJUK.