TechNet MAGAZIN

2006. OKTÓBER-NOVEMBER

MÉLYVÍZ, CSAK ÚSZÓKNAK!

Windows Server®

A PARANCSNOKI FÜLKÉBEN ÜLVE

A hibákat a lehető leggyorsabban ki kell javítani – System Center Operations Manager 2007

A WINDOWS VISTA BEVEZETÉSE

Az új rendszert támogató új és megújult technológiák

A PKI-INFRASTRUKTÚRA KIÉPÍTÉSE

Fokozott biztonságú központ kialakítása – Windows PKI-infrastruktúra alapokon

INFORMATIKAI INFRASTRUKTÚRA – TERVEZÉS ÉS IMPLEMENTÁCIÓ:

A POWERSHELL HASZNÁLATA

TERVEZZ MERÉSZEN!

ISA SERVER 2006: ÚJDONSÁGOK

Microsoft TechNet

Gondoljon a jövőre!

Vegyen részt BIZTOSAN induló őszi/téli képzéseinken, és a tanfolyami díj 50%-át felhasználhatja 2007-ben!

Néhány képzés ízelítőül. Teljes akciós kínálatunkért figyelje internetoldalunkat!

november

- Windows XP üzemeltetés (2272) – november 13-17.
- Windows 2003 üzemeltetés (2273) – november 6-10.
- Windows 2003 hálózatüzemeltetés (2276/77) – november 6-10.
- Windows 2003 Active Directory (2279) – november 27-december 1.
- Visual Studio 2005 Windows fejlesztés haladó (2547) – november 16-17.
- ITIL Foundation – november 27-30.
- SQL Server 2000/2005 alapok, lekérdezések (2071) – november 13-14.
- SQL Server 2000 programozás (2073) – november 20-24.
- Exchange Server 2003 üzemeltetés (2400) – november 27-december 1.
- BizTalk Server 2006 alkalmazásfejlesztés (2933) – november 20-24.

december

- Windows 2003 hálózat rendszertervezés (2278) – december 4-8.
- Visual Studio 2005 osztott alkalmazásfejlesztés (2548) – december 6-8.
- SQL Server 2000 adminisztráció (2072) – december 4-8.

a mi tudásunk az Ön sikere

Az akcióval és jelentkezéssel, valamint induló tanfolyamokkal kapcsolatos további információkért, kérjük, látogassa meg weboldalunkat (Akciók), vagy keresse munkatársainkat! Képzéseinkre szakképzési hozzájárulás illetve SA oktatási utalványok is felhasználhatók!

Telefon: 203-0304/4122 m.

www.szamalk.hu/tisza

SZÁMALK

BIZTOS ALAPOKRA ÉPÍTENI

Budai Péter
Microsoft Magyarország

Még szerencse, hogy ez nem jelenthet kihívást a mai szoftverekkel. Olyan könnyű az egész.

A feladat általában egyszerűen hangzik: hatékonyabbá kell tenni a vállalat alkalmazottainak mindennapi munkáját. De mire van szükség ahhoz, hogy ezt meg tudjuk valósítani? Kiváló szakemberekre.

Olyanokra, akik képesek megfelelni a gyakran egymásnak ellentmondó, folyton változó igényeknek, és le tudják fordítani az üzleti igényeket az informatika nyelvére – és nemcsak elméletben, hanem a gyakorlatban is. Az elméleti, általános tudás megszerzése ma már szinte gyerekjáték. Elektronikus levelezésre van szükségünk? Telepítsünk egyet, a már jól bevált megoldások közül! Next–Next–Finish, és máris működik! Minek ehhez gyakorlat?

Hihetetlenül sokan gondolkodnak így, és ebben persze szerepük van az egyre könnyebben kezelhető szoftvereknek is. Mégis, ha hiányzik a telepítés előtt a körültekintő tervezés, vagy nincs meg a gyakorlati tapasztalatunk az igazán alattomos hibák elhárításához, bármikor összedőlhet, amit építettünk. Annak pedig senki sem örül, ha nem megy a levelezés, vagy eltűnnek dokumentumok, és nincs róluk mentés. Sőt. Kitör a pánik, és előbb-utóbb valódi szakembert hívnak, aki már lehet, hogy későn érkezik.

Kétségtelen, hogy szükség van a megfelelő alapozásra. Ismerni kell a bevált folyamatokat, módszereket, és tudnunk kell olyan biztos alapot adni informatikai rendszereinknek, ami minden pillanatban stabil, akár egy betonfal, de szükség esetén rugalmas is tud lenni, mint a gumi. Még mindig egyszerűnek tűnik?

TechNet MAGAZIN

SZERKESZTŐSÉG
Főszerkesztő
Sziebig Andrea – *asziebig@vogelburda.hu*
Szakmai lektor
Budai Péter – *i-pbudai@microsoft.com*
Vezető szerkesztő
Varga János – *jvarga@vogelburda.hu*
Nyomdai előkészítés
Budakeszi Bejárat Kft.
Korrektor
Matula Zsolt
Lapterv és címlap
Emotion Bt.

Szerkesztőség és kiadó címe:
Vogel Burda Communications Kft.
1077 Budapest, Kéthly Anna tér 1.
Tel.: 888-3400, fax: 888-3499

KIADÓ
A Microsoft Magyarország megbízásából kiadja a Vogel Burda Communications Kft.

A kiadásért felel
Walitschek Csilla
cswalitschek@vogelburda.hu
Tel.: 888-3450, fax: 888-3499

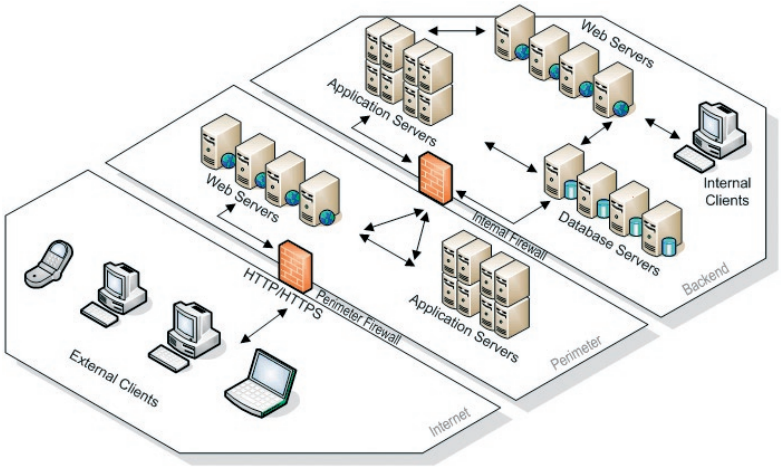
Hirdetési igazgató:
Farkas Viola – *vfarkas@vogelburda.hu*, tel.: 888-3459
Médiareferensek:
Harsányi Erika – *eharsanyi@vogelburda.hu*, tel.: 888-3452
Németh Krisztina – *knemeth@vogelburda.hu*, tel.: 888-3468
Rátóti Sarolta – *sratoti@vogelburda.hu*, tel.: 888-3453
Szendrey Szilvia – *szendrey@vogelburda.hu*, tel.: 888-3455
Fax: 888-3459
Hirdetési koordinátor:
Szőke Erika – *eszoke@vogelburda.hu*
Tel.: 888-3411, fax: 888-3459
Nemzetközi hirdetésfelvétel:
Eric N. Wicha – *ewicha@vogelburda.com*
Vogel Burda Holding
Poccistrasse 11, D–80336 München
Tel.: +49 89 74642-326, fax: +49 89 74642-325
A hirdetések körültekintő gondozását kötelességünknek érezzük, de tartalmukért felelősséget nem vállalunk.
Marketing:
Gajdos Barna – *bgajdos@vogelburda.hu*, tel.: 888-3494

Terjesztés
Terjesztett példányszám: 3000

Nyomda:
Pauker Nyomdaipari Kft.
1047 Budapest, Baross utca 11-15.
Felelős vezető: Vértes Gábor ügyvezető igazgató
ISSN 1586-5185

Címlapon

Ezúttal az informatikai infrastruktúra legáltalánosabb tervezési és implementációs kérdéseire koncentráltunk; megnézzük, egyáltalán milyen építőelemek állnak rendelkezésünkre



Biztonság

Infrastruktúra

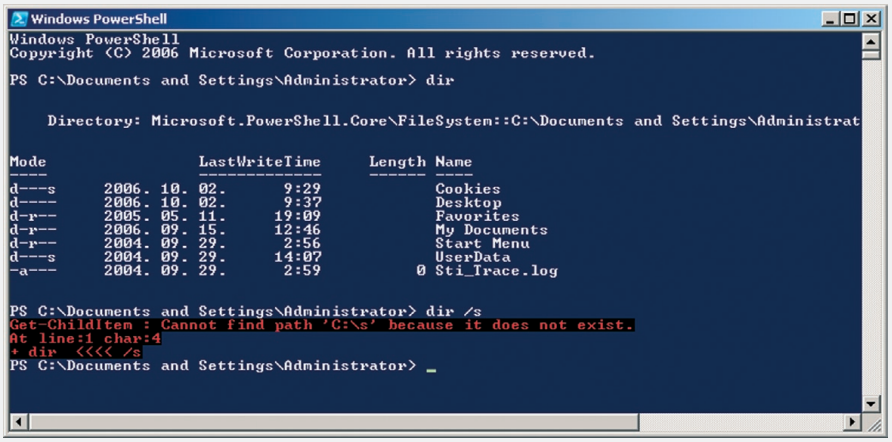
Alkalmazásplatform

Közösség

Tanuljunk együtt! A PowerShell használata

Íme a parancssor, ami forradalmasítja a rendszerfelügyeletet.

Aki látja a jövőt, az tudhatja, hogy a PowerShell mindent visz. Előbb-utóbb mindenkinek ugyanúgy meg kell tanulnia, mint annak idején a DOS-t – egyszerűen nincs nélküle élet a Windows-világban. Korábbi számunkban már adtunk egy alapos áttekintést arról, hogy mire is képes az új shell, azonban a következőkben tényleg az alapoktól kezdve, gyakorlatilag nulláról fogunk hozzá, hogy megmutassuk, hogyan érdemes elsajátítani az új parancssor használatát.



1. kép. A hagyományos, DOS-os belső parancsok nem pont ugyanúgy működnek

Az első olyan kiszolgálótermék, amelyik nem fog „vacakolni” a hagyományos VBScriptes, COM-os rendszerfelügyelettel és automatizálással, az Exchange Server következő változata lesz. Utána szép lassan az összes kiszolgálótermékénél is be fogják vezetni a .Net keretrendszerre épülő automatizálási lehetőséget, így például az új MOM-nál is, ami már a System Center Operations Manager 2007 nevet kapja megjelenésekor.

Nulláról úgy kell indulni, hogy telepítjük a jószágot. A Microsofttól letölthető 1,6 megabájtos csomag néhány másodperc alatt telepíthető a .Net Framework 2.0-val természetesen már régen felvértezett operációs rendszerre. Ahol nincs fent a legújabb Framework, ott a telepítő megáll, és reklamál. A hiányosság pótlása után pillanatok alatt elérhető az új parancssor. Az első parancs, amit ki „kell” adnunk, természetesen a DIR lesz. És minő meglepetés, egy könyvtárlistát kapunk eredményül. Eddig rendben, a nehezén túl vagyunk, ez is csak egy parancssor. Próbálkozzunk most valami nehezebbel, mondjuk a DIR /S paranccsal! Hát ez nem jött össze *(1. kép)*.

Szép piros hibaüzenet tájékoztat arról, hogy a C:\s könyvtár nem létezik. Ebben teljesen igaza van, de ki akart oda menni? Mi történik itt?

DOS-os fejjel...

Próbálkozzunk a jól bevált „fejjel a falnak” módszerrel (a dokumentációt pedig tartsuk távol magunktól!), és kérjünk egy helpet a DIR-parancsról!

```
HELP DIR
```

Talán meglepő, de működik a help parancs. A válaszból kitűnik, hogy a DIR becsületes neve Get-ChildItem, és az is kiderül, hogy hogyan kell leküldeni az alkönyvtárakba: -recurse, vagy egyszerűen: -r kapcsolóval. Joggal kérdezhetik, hogy miért változott meg az égvilágon minden, ha egyszer ugyanazt a feladatot hajtja végre a Get-ChildItem, mint a DIR parancs.

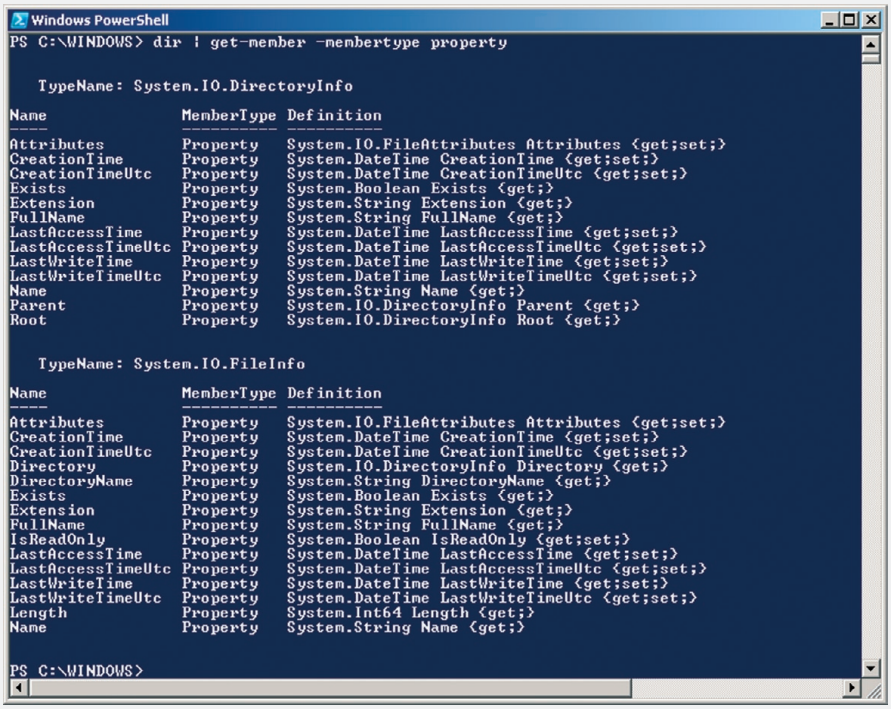
Mert nem ugyanazt végzi el, nem ugyanúgy, és nem ugyanazzal az eredménnyel! A Get-ChildItem elég sokféle típusú adathalmazból képes listát generálni, többek között könyvtárakról is. Továbbá például regisztrációs adatbázisból, tanúsítványtárból is...

A Help parancs úgy mutatkozik be ne-

künk, hogy ő bizony a Get-Help, ezért mostantól használjuk így. A következő feladat a könyvtárváltás (CD) parancs lesz. Kérdezzük le, ő ki fia-borja?

```
Get-Help CD
```

„Én vagyok a Set-Location! (Valójában a CD csupán egy alias, egy becenév a Set-Locationre, lásd: get-alias.) Ha csak úgy egylálnunk, de itt még nem tartunk. Egyelőre állapítsuk meg, hogy – ellentétben a DOS-os külső és belső parancsok elnevezésével és szintaxisával – egy konzisztens névtérben találjuk a parancsokat, amelyek hivatalos elnevezése: Commandlet, parancsocska. Amíg a korábbi shellekben egy parancsot pont úgy hívnak, ahogy azt készítője megálmodta (például NetSh, DriverQuery stb.), és a nevek között semmilyen rendet nem lehet felfedezni, most mindegyikük neve egy kéttagú kifejezés, ahol



2. kép. A Get-ChildItem a fenti adatokat kezeli a fájlrendszer objektumain

magamban használsz, már sokkal többet tudok, mint a jó öreg CD parancs, hisz át tudsz állni velem a registryre, a tanúsítványtárra, hogy ott listázgass testvéremmel, a Get-ChildItemmel. Sőt! Van memóriám! Bármikor elmentheted velem, hogy hol jársz éppen a Push-Location testvérparanccsal, ahová tetszőleges pillanatban visszaugrom, ha kiadod a Pop-Location parancsot!"

Nézzünk néhány példát!

Minden parancs valójában egy Cmdlet

Amit az eddigiekben láttunk, csupán a jéghegy csúcsa, hisz a mélyben valahol valami dotnetes döbbenetet kellene találnunk, de itt még nem tartunk. Egyelőre állapítsuk meg, hogy – ellentétben a DOS-os külső és belső parancsok elnevezésével és szintaxisával – egy konzisztens névtérben találjuk a parancsokat, amelyek hivatalos elnevezése: Commandlet, parancsocska. Amíg a korábbi shellekben egy parancsot pont úgy hívnak, ahogy azt készítője megálmodta (például NetSh, DriverQuery stb.), és a nevek között semmilyen rendet nem lehet felfedezni, most mindegyikük neve egy kéttagú kifejezés, ahol elöl áll a művelet rövid neve (például Get, Set, Format, Start, Stop stb.), majd kötőjellel elválasztva egy objektumnév (például Service).

Ezen túlmenően a parancsok szintaxisa és kapcsolói is konzisztensek, mert maga a PowerShell felelős a paraméterek kezeléséért, nem pedig a parancsocska. Ennek az a praktikus oka, hogy a parancsocskák a háttérben nem stringekkel, hanem .Net objektumokkal manipulálnak, ezekkel végzik a műveleteket, amelynek gyakran az a végük ugyan, hogy kiírjuk a képrenyőre az eredményeket – de ez nem feltétlenül van így.

DIR *.TXT /s	Get-ChildItem . –Include *.txt -Recurse
A HKLM/Software ág kilistázása a registryből	Get-ChildItem registry::HKLM/Software
DIR *.EXE	Get-ChildItem * -I *.exe
Átállás a Q: meghajtóra	Set-Location Q:
Jelenlegi „állás" elmentése	Push-Location
Átállás a HKEY_CURRENT_USER ágra a registryben	Set-Location HKCU:
Visszaugrás egy korábbi mentési pontra	Pop-Location
Átállás a tanúsítványtár „meghajtóra"	Set-Location Cert:

Csövezés .Net-módra

Számtalan shellt ismerünk, ahol a futtatott parancsokat egymáshoz lehet csatolni a pipe (|) karakterrel, ám a korábbi parancskörnyezetek az egyes parancsok között csak és kizárólag stringeket tudtak közvetíteni. Ha az input-output valójában dátum – akkor is string. Emiatt aztán minden parancs kénytelen az input stringből kimazsolázni és saját belső adattípusára konvertálni azt az adatot, amire szüksége van. Ez sok esetben annyira bonyolult lenne, hogy hozzá sem kezdünk.

Próbált-e már valaki egy DIR-listából dátumokat vagy Read Only flageket kivonatolni? Esetleg egy IPConfig outputból kiemelni a Default Gateway címét? Hogy futna ez a kód egy német Windowson, vagy uram bocsá' egy kínain? Elég lehetetlen feladat, a 100 százalékos pontosság eléréséhez mesterséges intelligenciára lenne szükség. Mivel ez utóbbit még nem találták fel, marad a kézihajtány-megoldás, illetve egy olyan kapcsoló a parancsba, ami feldolgozási hiba esetén is engedi továbbfutni. Nem elegáns!

Natív formátumban

Mi lenne, ha minden adat megmaradna a saját natív formátumában, és így menne át a csöveken? Közelítünk a lényeghez – ugyanis a PowerShell pontosan ezt teszi! Valahányszor lekérdezünk egy listát, valójában a mögöttes .Net-objektumok kupacát (collection) kapjuk vissza, amit akár ki is boríthatunk a képernyőre (ilyenkor persze stringgé alakul), de tovább is passzolhatjuk a pipe karakterrel, anélkül, hogy információt veszítenénk!

A motorháztető alá be is lehet kukkantani, ugyanis a speciális get-member Cmdlet arra való, hogy kilistázza a „belepájpolt" .Net-objektumok tulajdonságait (property) és metódusait. Példaképpen *(2. kép)* íme a Get-ChildItem-kimenet Get-Memberbe csövezésének eredménye (csak a propertyk).

Jól látható, hogy minden adat megőrzi a típusát, tehát például a FullName String, de a

LastAccessTime DateTime típusú, míg az alsó listában szereplő IsReadOnly egy Boolean (logikai) típust képviselő adat.

A futtatókörnyezetről és a változókról

Bizonyítsuk be, hogy minden adat megőrzi a típusát! Ha beírjuk a parancssorba, hogy 1, ezzel nem egy szintaktikai hibát adtunk meg, hanem egy int típusú konstansot. Írjuk be azt, hogy 1+1, és a válasz 2 lesz – a Shell tehát elvégezte az alapműveletet! Akár úgy is tekinthetjük, hogy nem egy parancssorban, hanem egy programozási környezetben, sőt, méginkább egy lépésenkénti futásra kényszerített programban vagyunk. Állunk a feldolgozás közepén, és látjuk, sőt, módosíthatjuk a futás feltételeit – mint egy debuggerben.

Mi több, változókat is létrehozhatunk, adatokat tölthetünk bele, és amíg a Shell fut, ezek meg is maradnak. Mit tehetünk egy PowerShell-változóba? Bármit! Akár egy DIR teljes eredményét! Lássunk erre egy példát *(3. kép)*!

Az első sor nem ad vissza eredményt a képernyőre, mivel azt „lenyeli" a $akarmi nevű változó. A második sor borítja elénk a $akarmi tartalmát, ami egy komplett, objektumorientált könyvtártartalom-kollekció! Később látni fogjuk, hogy van egy speciális változó, a $_, ami mindig az aktuális objektumot tartalmazza.

Most térjünk vissza a csőbe...

Szűrés a csőben

Gyakran lehet szükség arra, hogy csak bizonyos objektumokat dolgoztassunk fel a csőben, vagyis meg kellene szűrni az átjutó objektumokat valamilyen feltétel szerint. Bizonyos parancsok eleve tudnak szűrni néhány mezőre (például a DIR épp ilyen), azonban még ennél is szükség lehet egy általános szűrőre, amelyik tetszőleges mezőérték alapján képes szűrni. Erre találták ki a Where-Object Cmdletet, amelyik egy megadott szűrőfüggvény alapján rostálja át az objektumokat. Az alábbi példa az Event Logból csak és kizárólag az Error típusú bejegyzéseket listázza ki, vagy adja tovább a csőben, ha van még feldolgozóegység:

```
get-eventlog system | where-object -filterscript
{$ _ .entrytype -eq „error"}
```

Egy kis segítség a fenti furcsa képlethez: $_ az aktuális objektum a csőben, -eq nem más, mint „egyenlőségjel". Ami most a csőből kipotyog, egyenesen a képernyőre kerül. Írjuk fájlba! Ehhez mindössze a végére kell csövezni az out-file parancsot:

```
get-eventlog system | where-object -filterscript
{$ _ .entrytype -eq „error"} | out-file c:\aaa.txt
```

Ez a szűrés után fennmaradt összes Error típusú objektum valamennyi mezőjét gondolkodás nélkül kiírja a fájlba. Lehetőség van azonban az adatok kimazsolázására is. A csőben érkező objektumkupac ugyanis nem más, mint egy foreach-csel bejárható gyűjtemény, magyarul collection!

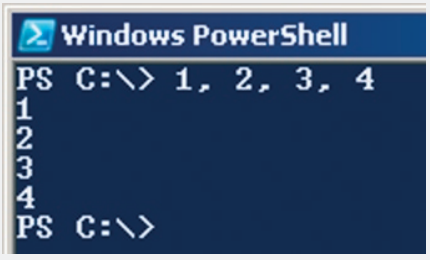
Objektumkupacok – ciklussal

Az objektumkupacok egyenkénti feldolgozását – mint minden rendes nyelvben – itt is a foreach konstrukció teszi lehetővé. (Ennek pontos neve: ForEach-Object, de működik a foreach alias is.)

Alkossunk egy egyszerű objektumkupacot a feldolgozás megértésének egyszerűsítésére! A legegyszerűbb collection egy statikus felsorolás, mondjuk például:

```
1, 2, 3, 4
```

Ha valaki ezt a négy számot így begépeli a PowerShellbe, érdekes módon nem „syntax error"-t kap válaszul, hanem ezt:



4. kép. Egyszerű objektumkupac

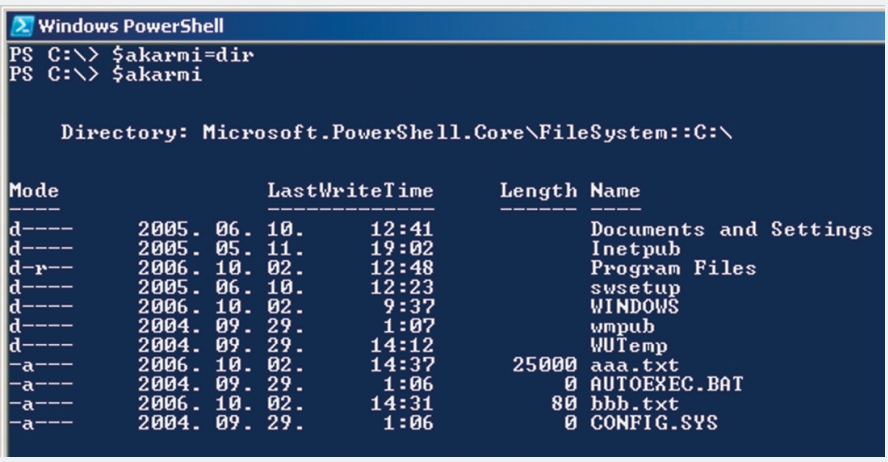
Itt egy négyelemű collectionnal van dolgunk, ami most lustán kifolyik a StdOut-ra (képernyő). Azonban a kupac elemeit fel is tudjuk dolgozni! Például számoltassuk ki a parancssorral a csőben kapott sugarú körök területét, vagyis szorozzuk meg mindet 2×3,14-gyel (5. kép.)!

Most már nyugodtan nekieshetünk az Error típusú Event Log-bejegyzések ciklikus feldolgozásának. Legyen az a feladat, hogy csupán a hibaüzeneteket írjuk ki, a többi adat nem kell!

```
get-eventlog system | where-object -filterscript
{$ _ .entrytype -eq „error"} | foreach -process
{add-content aaa.txt $ _ .Message}
```

Ugye, milyen egyszerű? Hát nem. Fél óráig kellett bütykölni, amíg ez így összejött, mégpedig azért, mert korábban egyszer belenavigáltuk a tanúsítványtárba, és ott próbáltuk meg lefuttatni, amire a következő érdekfeszítő, pirospozsgás, de teljesen semmitmondó hibaüzenetet kaptuk:

```
Cannot use interface. The IContentCmdletProvider interface
is not implemented.
```



3. kép. A $akarmi változóba betöltjük a DIR kimenetét, majd ki is listázzuk

Nesze neked, rendszergazda! Lesz itt még mit tanulni!

Konklúzió és további lehetőségek

A Microsoft célja alapvetően az, hogy minden parancs vagy lista, ami az MMC konzolon vagy egy webes adminfelületről elérhető, illetve amit az adott szerveralkalmazás kiajánl magából automatizációhoz vagy saját felügyeletéhez, az Windows PowerShell scriptletek vagy providerek formájában is rendelkezésre álljon. Így a rendszergazda eldöntheti, hogy a grafikus vagy a parancssoros interfészt használja feladatainak elvégzéséhez. A grafikus felület nyilván sokak számára egyszerűbb, azonban a parancssor használatával messze több lehetőségünk adódik; például ugyanazokkal az eszközökkel barangolhatunk a registry, a fájlrendszer, a tanúsítványtár vagy éppen az Exchange-mailboxok között, és egy nagyon korrektül összerakott, típusos, .Net alapú scriptnyelvvel dolgozhatunk az ott talált adatokkal.

Az Exchange Server 2007-ben szinte minden funkció elérhető lesz az új MMC konzoljáról, azonban ha automatizálni szeretnénk működését, vagy parancssorból szeretnénk matatni a levelezőszerverünkön, megtehetjük a Windows PowerShell alapokra épülő Exchange Management Shellel is. Az Exchange 2007 – nagyon hasonlóan az SQL Server 2005-höz – képes már arra is, hogy ha a grafikus felületen módosítunk valamit, azt scriptként is visszaadja nekünk. Mégpedig egy olyan PowerShell-script képében, amit azonnal tudunk használni az új parancssorban vagy automatizáláshoz.

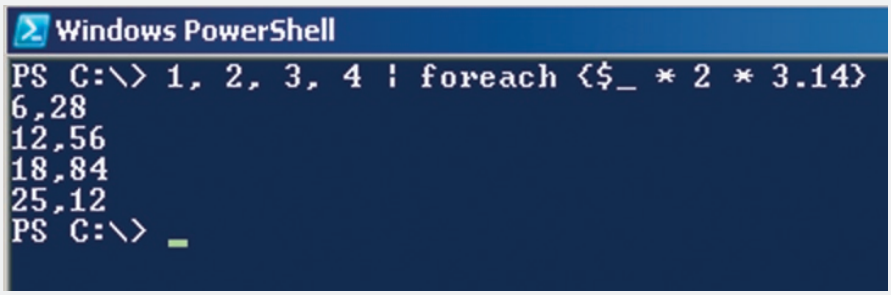
Az pedig, hogy a PowerShell szinte teljesen .Net alapú, még egy érdekességet mutat: a szerverszoftverek korábban szinte kizárólag programozóknak szóló API-jai és objektumkönyvtárai végre elérhetők lesznek egyszerűbb formában a rendszergazdák számára is, a szeverszoftverek testreszabásához és automatizációjához kevesebb alkalommal lesz szükség fejlesztő beavatkozására. Ezzel egy időben pedig a rendszergazdák is közelebb kerülnek a .Net-es programozáshoz.

A cikk megírásakor jelent meg a Windows PowerShell RC2-es változata jó néhány újdonsággal (például van benne közvetlen ADSI-támogatás is Active Directory eléréséhez és WMI provider!). Elképzelhető, hogy a Magazin megjelenésekor (várhatóan az Exchange 2007 kódjának elkészültével egy időben) már elérhető lesz a weben az új parancssor végleges telepítője.

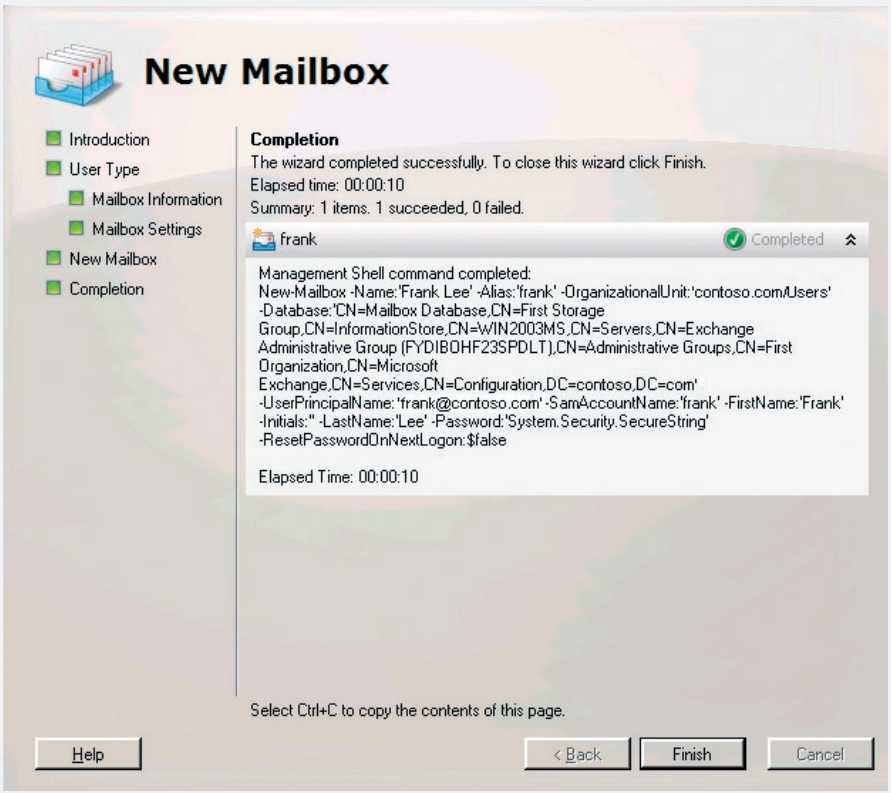
Ugyancsak az elmúlt napokban lett elérhető a http://powershell.com/ oldalon a Scriptinternals PowerShell scriptszerkesztő és debuggoló fejlesztőeszköze, ami teljesen ingyenesen letölthető.

Érzésre valahol félúton van a Turbo Pascal és a Visual Studio között, a design pedig az Office 2007-et idézi – de nagyon hasznos tud lenni komplexebb scriptek írásához.

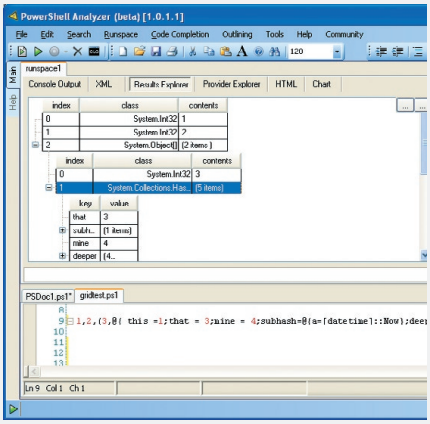
Fóti Marcell
(marcellf@netacademia.net) MCSE, MCT, MZ/X since 1995



5. kép. Excel tábla helyett használjon ön is PowerShellt!



6. kép. Azonnal használható PowerShell-script az Exchange Server 2007 varázslójában



Egy másik tool, a PowerShell Analyzer

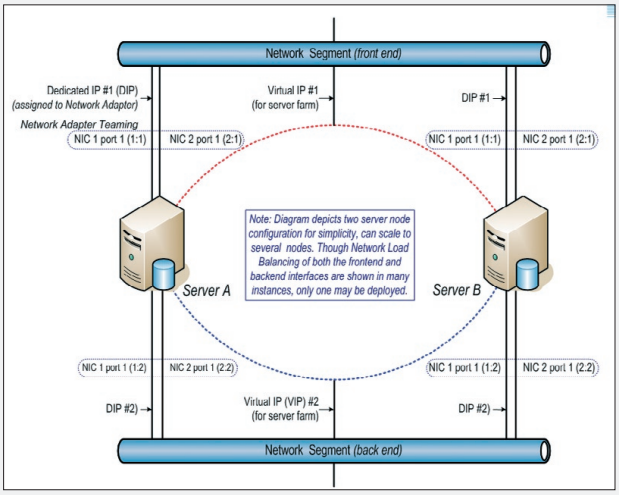
Windows Server System Reference Architecture

Egy informatikai vezető vagy rendszermérnök gyakran ütközhet olyan akadályba, amilyennel még sohasem találkozott korábban. Ilyenkor kellemes volna, ha már azonnal használható ötleteket kaphatna az adott problémához.
A WSSRA ezért jött létre. Pontosabban: ezért is…

Régóta létezik, mégis alig kapott visszhangot, Magyarországon pedig egyáltalán nem lehetett látni-olvasni olyan publikációt, amely ezt a hatalmas és sok tekintetben példaértékű dokumentumhalmazt bemutatta volna. Pedig a problémák, amelyek miatt megszületett, teljesen általánosak, nap mint nap küzdünk velük.

Hogyan cseréljük le az informatikai infrastruktúra tervezésének hagyományos látásmódját? Honnan kaphatunk kipróbált ötleteket egy adott problémához?

Hogyan készítsünk üzemeltetési kézikönyvet egy szolgáltatáshoz? Milyen napi/heti/havi feladatok tartoznak egy szoftver üzemeltetéséhez?



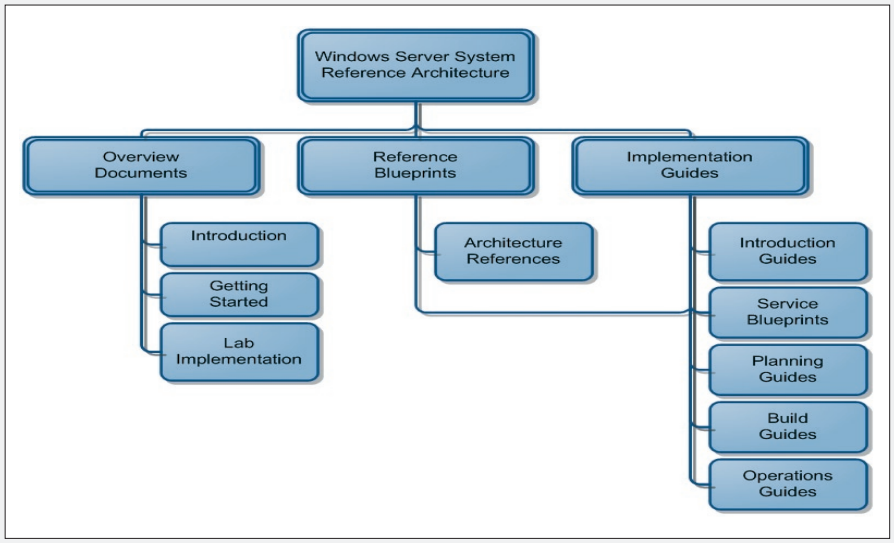
Terhelésmegoszlás clusterekkel a Security Architecture Blueprintben

Képzeletbeli infrastruktúra

A Microsoft partnereivel együtt elkészítette, majd dokumentálta egy képzeletbeli vállalat teljes IT-infrastruktúráját. Mindez több célt is szolgált. Bizonyítani szerették volna a Windows együttműködési képességét, skálázhatóságát, megbízhatóságát, komplex feladatokra való alkalmasságát. Ezen túl a cég egy „élő", működő, letesztelt példán keresztül igyekezett tapasztalatokat átadni vevők/felhasználók számára. A WSSRA nem csupán a Microsoft-termékekre igaz, habár elsősorban a Windows Server és a Microsoft szerveralkalmazások felhasználhatóságát hivatott bemutatni. A leírás többek között hálózati elemeket (Cisco), tárolórendszereket (EMC, HP), szerverhardvert is felvonultat (alkalmaz, méretez, üzembe állít stb.). A cél egy teljes és működőképes rendszer létrehozása volt.

A WSSRA nem előzmények nélküli. Az első verziót a Windows 2000-hez készítették el, annak még „Microsoft System Architecture" volt a címe, és két részből állt: egy „Enterprise Data Center (EDC)" és egy „Internet Data Center (IDC)" fejezetből. Később úgy döntöttek, hogy minden egyes Windows-verzióhoz elkészítenek egy referencia architektúrát, így jelent meg a mostani WSSRA 2.0-s verziószámmal. A jelenlegi megoldás még az eredeti Windows Server 2003, és nem a legfrissebb R2 változat alapján készült, az azonban bizonyos, hogy a Windows-szerver következő nagy verzióváltásával együtt a dokumentációs csomag is meg-

újul majd. A megelőző verziót persze felhasználták, de az EDC, IDC a 2.0-s változat szemszögéből már csak egy modell, a teljes IT-infrastruktúra része. Az új verzióban a modellek száma kibővült, olyan szituációkat is kidolgoztak, mint a Department, a Branch Office, a Satellite Branch Office és az Extranet. A WSSRA egyik legnagyszerűbb tulajdonsága, hogy a szolgáltatás oldaláról közelíti meg a teljes problémát, olyannyira, hogy már maguknak a dokumentumrendszereknek a felépítése is a szolgáltatásokat követi.



A WSSRA szerkezete

A teljes WSSRA-anyag összecsomagolva 41 megabájtnyi, kibontva 104 állomány és 74 megabájt, tehát meglehetősen méretes valami. A tömörítés eltávolítása után három fő könyvtárat kapunk, ezek rendre Overview Documents (Általános ismertetők), Reference Blueprints (Elméleti tervezési ismeretek), Implementation Guides (Implementációs útmutatók).

A legkisebb, mindössze három Word-állományt tartalmazó fejezet az első rész. A csoporton belül a legelső dokumentum címe „Introduction to Windows Server System Reference Architecture” (27 oldal): úgy értelmezi az egész anyagot, hogy leírja a problémákat, amelyekkel az foglalkozik, a módszereket, amelyekkel a problémákhoz nyúl, és tisztázza, hogy az olvasók (munkakörük alapján) mit nyerhetnek a betekintéssel.

A „Getting Started with Windows Server System Reference Architecture” (26 oldal) azt rögzíti, milyen vállalatmodellekkel foglalkozik a leírás, milyen egyedi dokumentumokat tartalmaz a WSSRA, azoknak mi a szerepük és hasznuk. Végül a „Lab Implementation of Windows Reference Architecture” (68 oldal) azt a tesztkörnyezetet írja le, amelyen felépült a WSSRA, továbbá olyan elnevezési és jelölési standardokat ismertet, amelyek alapján például a nevek és ábrák értelmezhetővé válnak. Ha valaki szeretne korrekt névkonvenciót bevezetni, ebben a fejezetben találja meg az egyik legjobb példát.

A WSSRA-dokumentumok második nagy csoportja a „Reference Blueprints”. Sokáig kellett keresni, mit fed a Blueprint kifejezés: a másolaton túl „terv”-et, „részletes terv”-et is jelent. A tartalom alapján a Blueprints olyan elméleti háttérinformációkat tartalmaz, amelyek a későbbi tervezési és üzemeltetési útmutatók megértésében segítenek. Ha egyetlen mondatot sem jegyzünk meg abból, amit a WSSRA megoldáshalmaza kínál, a Blueprintnek nevezett dokumentumokat akkor is érdemes elolvasni, mert ezek lényegében erősen a lényegre törő tankönyvek. Alapismeretek birtokában a Blueprintek képesek a fejekben lévő ismereteket rendszerezni, kontextusba helyezni.

Ha valaki végigolvassa őket (80–90 oldal egy-egy állomány), akkor úgy érzi, hogy az adott témát (például Storage, Network, Active Directory stb.) minden aspektusában átlátja, a lehetséges problémákat ismeri, és bizonyos, hogy a saját rendszerére vonatkozóan ötletei lesznek a jobbításra.

A Blueprintek két csoportba sorolhatók. Az első csoport az „Architecture Blueprints”, amely öt fő területtel foglalkozik általánosságban: Application, Management, Network, Security, Storage.

A második csoport a „Service Blueprints”, egy-egy adott szolgáltatás tervezési kérdéseit (de nem a konkrét tervezését!) tárgyalja. A Service Blueprintek egyébként nem is a „Reference Blueprints” mappában találhatók, hanem az egyes szolgáltatásokhoz helyezték el őket. Logikailag mindkét helyre illenek, ezért a furcsa vonal a fenti ábrán a „Reference Blueprints” ágból a Service Bluprintekbe.

A harmadik nagy dokumentumhalmaz az „Implementation Guides”, magyarul megvalósítási útmutatók, amelyek minden egyes szolgáltatáshoz öt dokumentumot tartalmaznak. Ezek sorra:

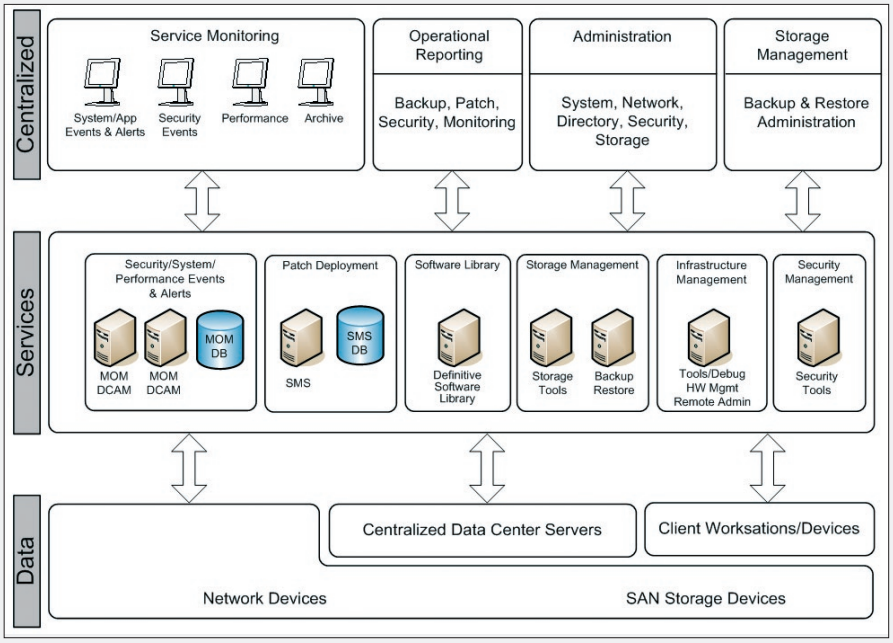
- **Introduction (Bevezetés).** Alapkérdések tárgyalása, miért van szükség egyáltalán a szolgáltatásra stb.
- **Service Blueprints (A szolgáltatás elméleti háttere).**
- **Planning Guides (Tervezési útmutató).** Ez már a konkrét (valójában a fiktív Contoso nevű) vállalatnál a szolgáltatás felépítésének konkrét kérdéseit, döntési pontjait ismerteti minden egyes modell esetén (EDC, IDC Brach Office stb.) Miután az adott szituációban az adott megoldást kiválasztották, megválaszolják azt a kérdést is, hogy miért így döntöttek.
- **Build Guides (Telepítési útmutató).** Pontosan, lépésről lépésre megvalósítják a kiválasztott megoldást. Ha tehát valakinek olyan problémája van, hogyan kell valamit felépíteni, telepíteni, konfigurálni, jó eséllyel megtalálhatja azt a konkrét szolgáltatás Build Guide-jában.
- **Operations Guides (Üzemeltetési útmutató).** A megvalósított megoldás üzemeltetési feladatainak rögzítése, valamint az eljárások definiálása.

Nagy méretek sok-sok apró tanulsággal

A WSSRA egy nagy multinacionális vállalatot képzel el, több tízezer felhasználóval, sok száz szerverrel. Jogosan kérdezheti az olvasó, vajon neki, akinek csak 20–50–100–500 felhasználója van, vajon mi haszna ilyen óriási méretű rendszer dokumentációján átrágnia magát? Miért jó a WSSRA a középvállalatok informatikusainak és döntéshozóinak, ami-

kor náluk nagyságrendekkel kisebb rendszerekről van szó, kvázi „ez nem róluk szól”.

Meggyőződésünk, hogy ez nem így van. Még ha a konkrét implementációt nem is lehet egy az egyben megvalósítani, az IT-infrastruktúra logikai felépítése lényegében alig különbözik, akár nagyvállalatról, akár középvállalatról van szó. A szolgáltatások száma szinte ugyanannyi, a problémák hasonlóak, és láthattuk, hogy ugyanúgy szükség van standardizálásra, méretezésre, biztonsági intézkedésekre, jól definiált üzemeltetési eljárásokra. Ezek mind-mind megtalálhatók a WSSRA-ban, olyan példaként, amely alapul szolgálhat, és amelyet felhasználva gyorsabban és semmit ki nem felejtve hajthatók végre a helyi projektek. Ha csak a WSSRA alapjait alkalmazzuk, már akkor is egy szabványosított IT-infrastruktúrát hozhatunk létre, amely rengeteg IT-problémától megszabadíthat minket – legfőképpen csökkenti, csökkentheti a tűzoltó munkát az értelmes fejlődés javára. Aki viszont a részletekbe menően mintaként tekint a „mintamegoldásra”, az egy minden szeletében precíz, teljes rendszert építhet fel, dokumentálhat és üzemeltethet. Nem tudjuk, hogyan nézzen ki egy rendszerdokumentáció? – Koppintsuk le az itt látható struktúrát. Nem tudjuk, mi szerepeljen az üzemeltetési kézikönyvben? – Nyissuk ki a megfelelő szolgáltatás „Operation Guide”-ját, és máris látni fogjuk! És mindezt nem csak a Windowsra vonatkozóan.



A rendszermenedzsment sematikus váza a Management Architecture Blueprintben

Ott találjuk majd a hálózat, a storage, a szerverhardver leírását is. Végeredményben a WSSRA egy példán keresztül a tervezési, telepítési és üzemeltetési szaktudás transzferét jelenti. Pontosan azt, amire egy középvállalat erőforrás-hiányos IT-csapatának a legnagyobb szüksége van.

A dokumentáció tartalmaz egy ábrát, amely az általános IT-projekt költségeit ábrázolja WSSRA-segédlettel és anélkül. Mivel van, amit már más kitalált, felépített, megvalósított, ezért annak átvétele, testre szabása jóval kevesebbe kerül, mintha ez magunknak kellene kitalálni. Amikor egy nagyobb projektet tervezünk, nagyon is konkrét, pénzbe kerülő mérnökórákat takaríthatunk meg a módszerek átvételével ahelyett, hogy mindent magunk találnánk ki.

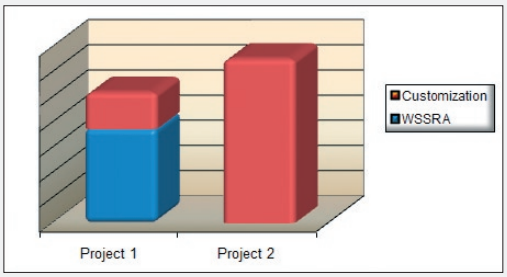
Ez az írás arra buzdít, hogy bátran éljünk azokkal a mérnöki dokumentációkkal, amelyeket kifejezetten nekünk írtak. Nem lenne azonban teljes, ha nem említenénk meg, mely témákról és technológiákról nem esik benne szó. A WSSRA kizárólag a központi infrastruktúra vonatkozik, és nem ejt egyetlen szót sem a kliensoldali technológiákról, sem a Windows XP-ről, sem az Office 2003-ról, sem az SMS-ről nem készült implementációs útmutató. Sőt, még a felhasználóknak kiajánlott „Terminálszerver” szolgáltatás sincs a rendszerben, hisz az nem más, mint egyfajta klienstechnológia. A „Deployment Services” fejezet is csak a kiszolgálók automatizált telepítésével foglalkozik, a kliensekével már nem.

A teljes csomag elkészítése, tesztelése hosszú időt vett (vesz) igénybe, így a nagyon gyorsan változó IT-világot egy kis lemaradással követi. Nem fogunk utalást találni a Windows Server 2003 R2 újdonságairól, és egyelőre adós a megoldás a szervervirtualizációs módszerek alkalmazásának bemutatásával is. A Microsoft akvizíciós tevékenysége felgyorsult az utóbbi időben, és számos olyan vállalat (Sybari, Softricity stb.) került a szoftveróriáshoz, ahol korábban és most is meghatározó infrastruktúramegoldásokat állítottak elő. Ezeket a szoftvereket nem volt alkalma a WSSRA-t készítő csapatnak beépíteni a referenciarendszerbe, így mi sem juthatunk innen tapasztalatokhoz róluk. Remélhetjük viszont, hogy a Longhorn generációhoz megjelenő referenciarendszer már egy részüket beépíti majd, demonstrálva az integrációból fakadó előnyöket.

Mindent egybevetve kijelenthetjük: a Windows Server System Reference Architecture az eddigi legtejesebb, nyilvánosan dokumentált mintarendszer, amely hosszú-hosszú ideig szolgálhat nagy és apró ötletekkel az azt forgatók számára.

Kellemes olvasgatást!

Lepenye Tamás
(tamasl@microsoft.com) MCSE, Microsoft Magyarország



Egy projekt költségei WSSRA-val és nélküle

További információk

A WSSRA dokumentumcsomag elérhető innen:
http://www.microsoft.com/hun/windowsserversystem/Refarch/default.mspx

TERVEZZ MERÉSZEN!

Sokan azt gondolják, hogy a szerverparticionálás a nagyvállalatok játékszere. Ez egyáltalán nem így van. A virtualizáció olyan lehetőségekhez juttathatja a kis- és közepes vállalatokat, amilyenekről eddig csak álmodni lehetett.

Tegyük fel, hogy egy ügyfélnél 30–50 felhasználó dolgozik egy hálózatban. Egyetlen telephely, szokásos igények: fájl- és nyomtatószolgáltatás, levelezés és csoportmunka, tűzfal, esetleg egy kliens–szerver alapú mini-ERP.

A nem túl nagy, de jól prosperáló és innovatív cég megfogalmaz „néhány" elvárást. A szolgáltatás legyen zökkenőmentes, és amennyire lehet, folyamatos. Az egyik szolgáltatás ne zavarja a másikat. Az infrastruktúra legyen biztonságos és naprakész, és bárhonnan, akár otthonról is elérhető. Legyen minden egyszerű, hogy bárki átláthassa: nem szeretnének egyetlen külső szolgáltató szervezettől sem függeni, sokkal inkább versenyeztetni lenne szerencsés őket. A mentésről is gondoskodni kell, sőt megoldás kellene az esetleges katasztrófák ellen. Ne legyen sok szerver, mert azok rengeteg pénzbe kerülnek, sokat fogyasztanak, és drága lenne hozzájuk a megfelelő áthidalást biztosító szünetmentes tápegység, a hűtésről már nem is beszélve. És még két apróság: a cég adatbázisokhoz fejleszt szoftvereket, és a teszteléshez szükség van minél többféle adatbázisra, lehetőleg úgy, hogy egymást a lehető legkevésbé zavarják. Ezen felül néhány munkatársnak okostelefonja van, az e-maileket szeretnék azokon is olvasni. Három szerverrel mindez megoldható, ugye?

Ha most valaki széttárja a kezét, és azt mondja, hogy nem, akkor kellemesen fog „csalódni": igenis, három, okosan méretezett szerverrel mindez megvalósítható. Kezdjünk tervezni, és meglátjuk!

Szolgáltatásleltár

Mindenekelőtt vegyük sorra, milyen szolgáltatásokat kell biztosítania a rendszernek a felhasználók számára:

- fájl- és nyomtatószolgáltatás;
- levelezés;
- internetelérés;
- betárcsázás/VPN;
- üzleti alkalmazás elérése (mini-ERP);
- hozzáférés adatbázisokhoz a szoftverek teszteléséhez.

Vegyük sorra, mindehhez milyen háttérszolgáltatásokra van szükségünk (itt a „háttérszolgáltatás" nem minden esetben azonos a Windows-kiszolgálók „szolgáltatás" (service) fogalmával):

- címallokáció és névfeloldás: DHCP, DNS WINS;
- címtár (Active Directory);
- RADIUS (IAS);
- elosztott fájlrendszer (DFS);
- szoftverfrissítés: WSUS;
- antivírusrendszer;
- mentés;
- Management SQL;
- rendszerfelügyelet (MOM);
- tűzfal;
- tanúsítványszolgáltatás.

Az elvárásokból nem lehet ugyan kiolvasni, de bizonyosan szükség lehet operációs rendszerek és alkalmazások telepítésére is.

Biztonság, rendelkezésre állás

A biztonságos rendszer követelmény, márpedig biztonsági sablonokat akkor tudunk hatékonyan létrehozni, ha egy, vagy csupán néhány szolgáltatás fut egy kiszolgálón. Itt az egyik látható és feloldandó ellentmondás: adott rengeteg elvárt, igényelt funkció, ugyanakkor nagyon kevés a rendelkezésre álló hardver.

Az ügyfél megfogalmazott folyamatos szolgáltatási igényt. Kisvállalatról lévén szó, itt elsősorban az alapszolgáltatások (DHCP, DNS, WINS, AD, IAS) redundanciájának megvalósítása jöhet szóba. Ez azonban tovább nehezíti a feladatot: a duplikálás szaporítja az elhelyezendő szolgáltatások számát.

Összességében tehát hat felhasználói szolgáltatás, tizenkét háttérszolgáltatás és öt tartalék-háttérszolgáltatás biztosítása a feladat.

A szolgáltatások csoportosítása

A Small Business Server jellegű rendszerek szélsőséges módon bizonyítják, hogy néhány ritka kivételtől eltekintve szinte minden szolgáltatás együtt futtatható más szolgáltatásokkal. Ez a gyakorlat azonban több problémát is felvet:

I. Teljesítmény. Ha a szolgáltatások azonos típusú erőforrást (például memóriát) na-

gyobb mértékben igényelnek, akkor együttes futtatásuk teljesítményromláshoz vezethet.

II. Kompatibilitás. Egy adott szolgáltatás más szolgáltatás futtatását kizárhatja.

III. Biztonság. Egy szolgáltatás olyan beállításokat igényelhet, amely egy másik biztonságosságát csökkentheti. Több szolgáltatás futtatása nagyobb támadási felületet jelent. Szeparálás esetén egy esetleges behatolás csak egy-egy funkciót érint, nem többet.

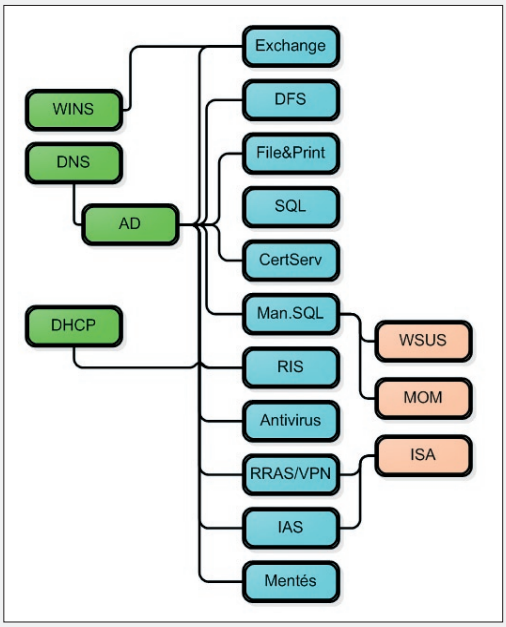
IV. Karbantartási idő. Egy szolgáltatásra szükség lehet, miközben egy másik karbantartása miatt állni kényszerül.

A fenti problémák kezelésére az általános megoldás a szerepkörök szerinti rendszerfelépítés. Ha egy kiszolgálónak csak egy (vagy néhány logikailag összeillő) meghatározott szerepköre van, akkor biztonsági sablonokkal menedzselhető, tehát állapota pontosabban definiálható és jobban megőrizhető. Egy tartományvezérlőn például szigorúbb biztonsági sablonok érvényesíthetők, ha nincs más funkciója. A levelezőkiszolgálók biztonsági beállításairól, ha csak ez az egyetlen funkció a szerveren, tucatnyi dokumentáció áll rendelkezésre, de más szolgáltatásokkal együtt már nekünk magunknak kell kitalálnunk, miképp legyen ténylegesen a lehető legbiztonságosabb.

Biztonságos, a legjobb gyakorlat elvét nem megsértő módon is lehetséges a szolgáltatások csoportosítása – a különböző Security-útmutatók ezekre felkészülnek. A mi példarendszerünkre egy lehetséges csoportosítás az alábbi:

I. Fájl- és nyomtatómegosztás, DFS
II. Levelezés
III. Tűzfal (vírusfal), betárcsázás, VPN
IV. Üzleti alkalmazás, SQL
V. Teszt-adatbáziskezelők szoftvertesztelésihez
VI. AD, IAS, DHCP, DNS, WINS, CertSrv
VII. AD-2, IAS-2, DHCP-2, DNS-2, illetve WINS-2
VIII. Management SQL, WSUS, mentés, antivírus, MOM, RIS

Úgy tűnik tehát, hogy a szélsőségek elkerülhetők. Nem szükséges 16 kiszolgáló beszerzése (100 százalékos szolgáltatásizoláció), és szigorúbb biztonsági sablonokat alkalmazhatunk, mint tennénk azt egy SBS-megoldás esetén (0 százalékos szolgáltatásizoláció).

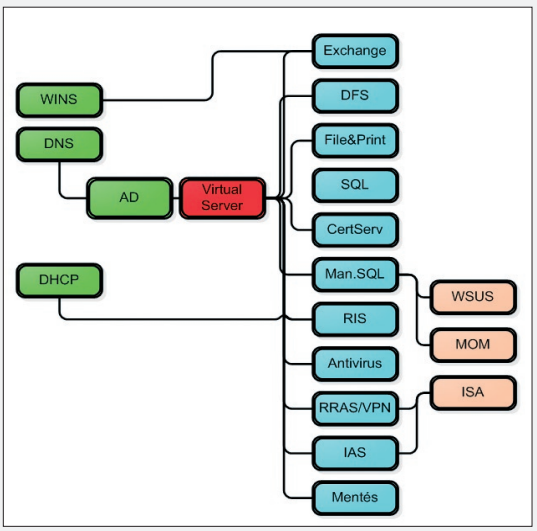


1. ábra. A szolgáltatások függőségi viszonyai

A nyolc csoport ugyanakkor nyolc szervert is jelentene, ez pedig még mindig nem elégíti ki a követelményeket.

Függőségek

Mielőtt a fenti nyolcas csoportot valahogy hárommá gyúrnánk össze, érdemes felvázolni a szolgáltatások egymástól való függőségét. Ez azért fontos, mert amikor majd a rendszer leállását/indulását tervezzük, akkor gondoskodnunk kell arról, hogy előbb azok a service-ek induljanak, amelyektől a többiek függnek. (Az *1. ábrán* látható struktúra akkor igaz, ha a szolgáltatásokat Microsoft-technológiákkal valósítjuk meg. Minden komponenst csupán egyszer tüntettünk fel.)



2. ábra. Függőségek Virtual Serverrel

Nem állítható, hogy minden egyes apróságot tartalmaz az 1. ábra, de az jól látszik, hogy mindössze négy alapszolgáltatás (DNS, WINS, DHCP, AD) indulási prioritását kell biztosítani.

A szerver–hardver problémája

Minden valamirevaló rendszermérnök tudja, hogy a mai processzorokkal szerelt szervergépek számítási teljesítménye kis felhasználószám esetén nem gazdaságos: nem tudunk 1/5, 1/10 processzort vásárolni, holott annyi is elég lenne. A korszerű gépekre ugyanakkor szükség van: a menedzsment, a távolról való vezérlés, a távolról való diagnosztizálás a legkorszerűbb eszközökben a legkifinomultabb, és persze a szerver élettartama is akkor a leghosszabb, ha új gépben gondolkodunk.

Vegyük emellett számításba, hogy a memória – szemben a korábbi évekkel – sokkal inkább megfizethető, továbbá már 5-6 lemezből is terabájtos tárolót lehet összeállítani úgy, hogy szó sincs még SAN építéséről. Látható az is, hogy mára egyetlen „egyszerű” szerverben olyan kapacitások és képességek halmozódnak fel, mint korábban egész adatközpontokban.

Rakjuk össze a mozaikokat: adott egyik oldalon egy sokszerveres – jelen esetben éppen nyolcszerveres – igény. Adott továbbá egy olyan kiszolgáló, amely eleve többletkapacitással vásárolható meg. Szinte adódik az ötlet, hogy „vágjuk fel” az oszthatatlannak tűnő szerver–hardvert valamivel. Ez a felvágás, felosztás a szerverparticionálás, más néven virtualizáció. Az eszköz pedig lehet például egy Microsoft Virtual Server 2005 R2 szoftver.

Ha egyetlen kiszolgálót virtuálisan több részre osztunk, akkor gazdaságossá válhatnak olyan beruházások, amelyek egyébként számosságuknál fogva nem lennének azok. A kiszolgálók környezetét és magukat a szervereket úgy tervezhetjük, hogy minél keve-

sebb leállás történjék akár tervezetten, akár váratlanul. Az ilyen vállalatméretnél gazdaságos megoldás lehet a kettős tápellátás; segít a menet közben cserélhető merevlemezek, a hardveres RAID-védelem és a memória meghibásodása ellen védő ECC vagy Chipkill technológia beépítése, betervezése. Mindezt pedig csak egyszer (vagy néhányszor) szükséges megvenni, nem pedig nyolcszor, miközben minden partíció rendelkezésre állása növekedhet.

A virtualizáció és hatása a tervezésre

Ha amellett döntünk, hogy a megoldás részeként virtualizációs technológiát is alkalmazunk, egy újabb réteget, újabb szolgáltatást építünk a rendszerünkbe. A Microsoft Virtual Server 2005 R2 akkor menedzselhető hatékonyan, ha tartományban üzemel, ezért módosítanunk kell az 1. ábrát, méghozzá a *2. ábrán* látható módon.

Mindebből úgy tűnik, az alapszolgáltatások nem virtualizálhatók, hiszen maga a szerverparticionálás is (hangsúlyozandó: teljes funkcionalitás esetén!) az alapszolgáltatásoktól függ.

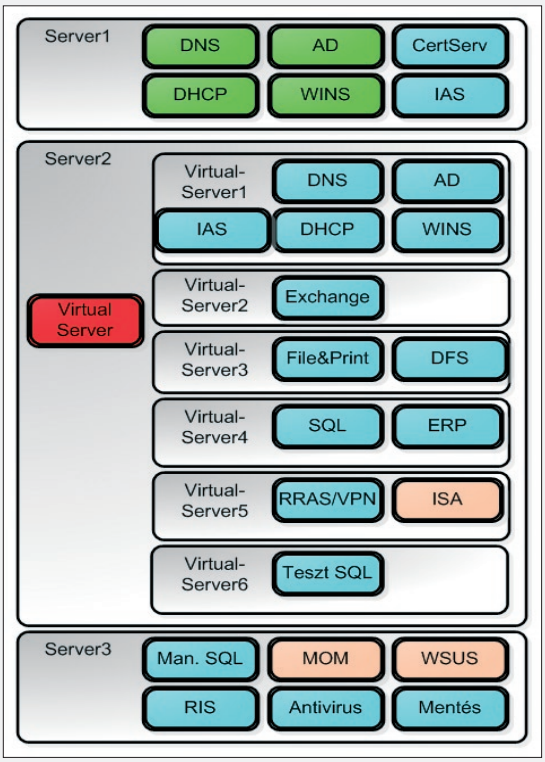
De nem ez az egyetlen probléma. Az egyes partíciók meghatározott és meglehetősen kötött virtuális hardvereszköztárral rendelkezhetnek, s ezek között nem található szalagos egység sem. Amennyiben nem szeretnénk lemondani erről a tárolási típusról – és másodlagos tárolóként bizony nagyon fontos a katasztrófavédelem szempontjából is –, kénytelenek leszünk a mentési szolgáltatást kiemelni a virtualizáció „alól". (Elvileg nem lehetetlen szalagos egység használata virtuális gépekben sem iSCSI segítségével. Az iSCSI alapú szalagos egységek ára azonban jelenleg nem teszi lehetővé, hogy az üzleti megoldásnál figyelembe vegyük őket.)

A szolgáltatáscsoportokat, a redundanciát, a függőségeket és a virtualizáció hozta további elvárásokat figyelembe véve a *3. ábrán* látható logikai rendszerfelépítés lehet az egyik megoldása az ügyfél igényeinek.

Az első szerver alapszolgáltatásokat is nyújtó tartományvezérlő, amely emellett tanúsítványkiszolgáló feladatot lát el. A hasonlóan szigorú biztonsági előírások miatt célszerű a szolgáltatások ilyen csoportosítása. Teljes rendszerindulásnál ez az első kiszolgáló, amely elindul, biztosítva a másik kettő és a virtuális szerverek számára a névfeloldást és a hitelesítést. A második kiszolgáló egy dedikált szerver, amely kizárólag virtuális rendszerek futtatását végzi. Összesen hat partíciót tartalmaz, és a korábban már meghatározott szolgáltatáscsoportokat futtatja egy-egy virtuális gépben.

Végezetül a harmadik rendszer menedzsment- és egyéb háttérszolgáltatásokat végez, többek között a mentést, a szoftverfrissítést és a központi antivírus-feladatokat.

Nézzük az igények teljesülését. A rendelkezésre állást a kevés fizikai eszközben érvényesített hardverredundancia, valamint az alapszolgáltatások megkettőzése biztosítja. A szolgáltatások előírás szerint elválasztottak, a rendszert „konzerv biztonsági sablonok"



3. ábra. Szolgáltatások három fizikai szerveren

gyors testreszabásával lehet megerősíteni. A szoftverfrissítés nagy részét elvégzi a WSUS. Az ISA szerver révén a rendszer bárhonnan elérhető (RAS/VPN). Az okostelefonokat az Exchange 2003, az ISA 2006 és a CA-szerver együttesen kiszolgálja. A Virtual Server 2005 R2 a Volume Shadow Copy igénybevételével hamarosan képes lesz a virtuális szerverek lemezeinek mentésére (harmadik gyártótól származó mentési rendszerrel). A mentések és a virtuális gépek rendszerpartícióinak szalagra másolásával a megoldás tényleges katasztrófák esetén is védelmet nyújthat. Felhasználói adatokat kizárólag a Server2 tartalmaz, ez megint csak a mentés konfigurálását egyszerűsíti. Teljesült a háromszerveres kritérium is, a megoldás akár 4U rackmagasságban is elhelyezhető. Ez végeredményben kisebb hűtési kapacitást és kisebb szünetmentes tápegységet igényel.

A megoldás még három járulékos előnyt hordoz. Amennyiben a Server2 operációs rendszere Windows Server 2003 R2, a hat virtuális gépből négynek az operációs rendszeréért nem kell külön fizetni.

Bár a 3. ábra csak egyetlen tesztjellegű SQL-kiszolgálót jelez, de a cég tevékenységéből adódik, hogy akár több ilyen virtuális gépre is szükségük lehet. A Microsoft licence lehetővé teszi, hogy korlátlan számú lekapcsolt virtuális géppel rendelkezzünk, s csupán azokért kell fizetnünk, amelyeket ténylegesen használunk, elindítunk.

Ez pedig sarokpontja lehet a megoldásnak. Hat–nyolc tesztgép esetén ugyanis már bizonyos, hogy a hardver- és szoftverárak a virtualizációt alkalmazó megoldást teszik olcsóbbá.

Végezetül egy majdani hardvercsere a Server2 esetén nem áll majd másból, mint a virtuális gépek lemezeinek másolásából, ami jelentős idő- és költségmegtakarítást eredményez.

Ennek a rövid példatanulmánynak a célja elsősorban az volt, hogy megmutassuk, miként befolyásolhatja a tervezés menetét a virtualizáció alkalmazása. Az információs rendszerek tervezésekor gyakran egymásnak ellentmondani látszó igényeket kell és – ahogy láthattuk – lehet kielégíteni.

A valóságban a tervezés itt nem állhat le, hiszen az eszközök fizikai kialakítása, méretezése, adott esetben teljesítménymutatóik előrejelzése, nem utolsósorban pedig teljes birtoklási költségük becslése, más alternatívákkal való összevetése mind-mind a beruházási döntést megelőző tevékenységek.

Jó tervezést!

Lepenye Tamás
(tamasl@microsoft.com) MCSE, Microsoft Magyarország

Van új a nap alatt

ISA Server 2006 – a hitelesítéssel kapcsolatos újdonságok.

Szeptemberben jelent meg az ISA-kiszolgálók harmadik generációs képviselője, az ISA Server 2006. Megjósolható volt ugyan, hogy első ránézésre drasztikus változást nem fogunk tapasztalni az előző verzióhoz képest, de ha kicsit alaposabban megvizsgáljuk, akkor azért több fontos, hasznos és nagyléptékű újítást is felfedezhetünk.

Egyelőre viszont csak egy témakört érdemes kiemelni szélesebb áttekintésre, és ez a hitelesítés vagy más szóval az autentikáció. Rögtön a témába vágva, nézzük meg, hogy egy ISA Server esetén milyen fő szakaszokra osztható a hitelesítési procedúra:

- a kliens jogosultságainak elfogadása;
- a jogosultságok érvényesítése, amelyhez egy hitelesítésszolgáltató szükséges (például Active Directory, RADIUS, SecurID);
- a hitelesítési adatok delegálása egy, az ISA „mögött” működő szerver (például egy SharePoint Portal Server) közreműködésével.

Az első két komponenst az adott listener (az ISA egy-egy, a beérkező kéréseket figyelő „füle”) segítségével konfiguráljuk, míg a harmadikat a vonatkozó publikálószabályban. Ez egyúttal azt is jelenti, hogy ugyanazt a listenert természetesen használhatjuk több publikálószabályhoz is, sőt esetenként változó delegálási típusokkal is. Ezek után viszont tekintsük át csoportokba szedve az összes, ISA Server 2006-ban megtalálható hitelesítési technológiát és módszert.

1. Nincs hitelesítés
2. HTML űrlap alapú klienshitelesítési metódus
   - Windows (Active Directory)
   - LDAP (Active Directory)
   - RADIUS
   - RADIUS OTP
   - RSA SecureID
   - SSO
3. HTTP alapú klienshitelesítési metódus
   - Basic
   - Digest/wDigest
   - Integrated
4. SSL klienstanúsítványon alapuló hitelesítés

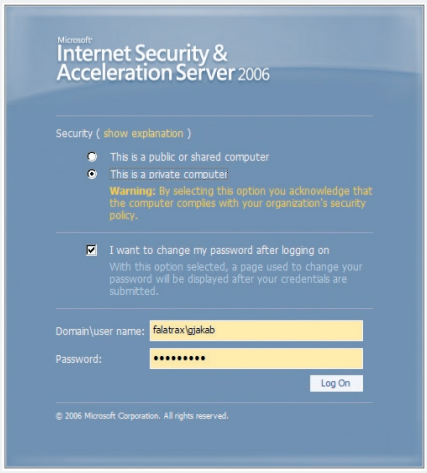
Űrlap alapú hitelesítés (Forms-Based Authentication)

Egyre több olyan webes alkalmazás, szolgáltatás jelenik az intranet/internet/extranet hálózatokban, amelyek miatt biztonságos, részletesen szabályozható és testreszabható hitelesítési formákat kell(ene) alkalmazni. Az intranet szó nem elírás, tényleg komolyan el kell gondolkoznunk azon, hogy a belső felhasználók, valamint a vezetékes vagy vezetéknélküli „alkalmi” kapcsolódók (szállítók, vendégek, ügyfelek, a laptopokkal, PDA-kkal stb.) azonosítás és hitelesítés nélkül érhessék-e el a belső hálózat eleddig semmilyen módon nem korlátozott erőforrásait (webszerverek, portálok stb.).

Az ilyen típusú igények megjelenése miatt az ISA 2006-ban gyakorlatilag bármilyen webszerver-publikálásnál használhatjuk az FBA-t, három fő csoportba osztva:

- **jelszó-űrlap:** felhasználónév/jelszó, az Active Directory, az LDAP és a RADIUS-jogosultságok ellenőrzésére;
- **passcode-űrlap:** felhasználónév/passcode, a SecureID- és a RADIUS-hitelesítésnél;
- **passcode/jelszó-űrlap:** felhasználónév/passcode a hitelesítésre és felhasználónév/jelszó a delegálás céljából.

Az űrlapos hitelesítés további újdonságai közül az egyik legfontosabb a jelszókezelés. Ez azt jelenti, hogy újra lehetőségünk van a felhasználók számára megengedni, hogy akár távolból is megváltoztassák a jelszavukat, valamint azt is elérhetjük, hogy az általunk be-



Egy szimpla weboldalhoz tartozó hitelesítés jelszóváltoztatási kéréssel

állított időhatáron belül a jelszóváltoztatás szükségességéről értesítést (és ennek apropóján rögvest egy jelszóváltoztató űrlapot is) kapjanak. A két opció nem szükségszerűen jár együtt, szerencsére külön-külön is szabályozhatóak.

További megjegyzések és tudnivalók az űrlap alapú hitelesítésről

- Az ISA Server 2006 képes ezt a hitelesítési formát a mobilkliensek számára is bizto-

sítani, persze csak akkor ha a mobileszközről beérkező kérés User-Agent fejléce alapján megfelelően detektálta a klienst. Az ide tartozó űrlapok az ISA telepítési mappájának \CookieAuthTemplates\ISA\CHTML illetve XHTML mappájában találhatók meg.

- Közvetlenül ide tartozik még az is, hogy a szimpla HTML-űrlapokat is teljesen testre lehet szabni, a telepítési mappa CookieAuthTemplates könyvtárában immár két almappa van: egy Exchange nevű, amelyben az eddig is ismert módon lehetséges az OWA FBA logon képernyőt testreszabni, és a már említett ISA nevezetű, amely az alap HTML-logon képernyő elemeit is tartalmazza, például az összes szöveges részt a Strings.txt-ben.
- Az ISA Server 2006 összesen 26 különféle nyelven képes megjeleníteni ezeket az űrlapokat, szintén egy vizsgálat, azaz a böngésző nyelvi beállításai alapján. Természetesen ettől eltérő beállítást is megtehetünk a publikálószabályon belül (Listener/Properties/Forms), azaz elképzelhető olyan felállás is, hogy más lesz az űrlap és más a böngésző nyelve.

SSO

Mivel egyre nő a publikált szolgáltatások száma, könnyen elképzelhető, hogy egy szervezeten belül több különböző erőforrást is szeretnénk „megmutatni” a külső felhasználóknak. Ez rendben is lenne, de azonnal adódik egy probléma is: mégpedig az ismételt bejelentkezések szükségessége.

Ezzel lényegében el is értünk az SSO (Single Sign-On, egyszeri belépés) módszer alkalmazásához, amelynek a lehető legtöbb esetben együtt kell érvényesülnie az új hitelesítési megoldásokkal, mert különben – elsősorban a felhasználók számára – nagyon fárasztóvá válhat egy-egy belépési folyamat. Ezt az opciót az ISA Server 2006-ban immár egy külön panelen találjuk az adott listener tulajdonságai között.

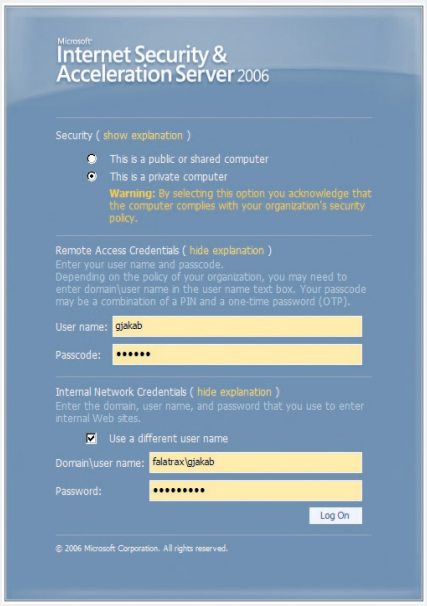
Multifaktoros hitelesítés

Teljesen egyértelmű, hogy a hitelesítési folyamatnak igazán biztonságosnak kell lennie, de hogyan lehet még jobban fokozni a biztonságosságot? Erősebb titkosító kulcsokkal, bonyolult, gyakran változó jelszavakkal, publikus és privát környezetben eltérő módon működő környezeti beállításokkal? Igen-igen, de mégsem csak így. Ha viszont másfelől közelítünk, és a többszörös hitelesítést, vagyis az úgynevezett multifaktoros módszert alkalmazzuk, akkor valószínűleg nagyobbat lépünk előre. Miről van szó?

Például az ISA Server 2004-ben bemutatkozó és az ISA 2006-ban alaposan kibővített kétfaktoros hitelesítésről, amely a felhasználót kettős hitelesítésre kényszerítheti, azaz egy user/password kombináció, valamint egy hardvereszköz vagy egy tanúsítvány „bemutatására”.

A lehetséges variációk szerint a felhasználónak rendelkeznie kell:

- egy tanúsítvánnyal;
- vagy egy egyszeri jelszót/kódot (One-time password, OTP) generáló szoftveres modullal/hardvereszközzel;



Űrlapos, dupla hitelesítés

- vagy egy SecurID-eszközzel, amely minden szükséges esetben (hosszú érvényességi időben, például 2-3 évig) egy úgynevezett passcode-ot (nagyon rövid lejáratú számkombináció) képes generálni.

Tipikus példa, hogy a felhasználó rendelkezik egy smartcardon elhelyezett személyes tanúsítvánnyal, amelyet az ISA ellenőrizhet a belső hálózatban elhelyezett CA-kiszolgáló segítségével, és ha pozitív a vizsgálat eredménye, akkor biztonságosan hitelesíti a felhasználót.

Még két gondolat a multifaktoros hitelesítés alkalmazásához az ISA 2006-ban:

- az űrlap alapú hitelesítés teljeskörűen együtt használható ezzel a módszerrel;
- az OTP az ISA 2004-ben maximum a SecurID megoldáshoz volt alkalmazható, mostantól viszont a RADIUS hitelesítésénél is támogatott.

Hitelesítésdelegálás

Egyre több esetben fordul elő, hogy a hálózatokat elválasztó elemnek, azaz a tűzfalnak kell az elsődleges ellenőrzést elvégeznie, azért is, hogy a belső hálózatban működő kiszolgálót ne terheljük, és ne kockáztassuk az állapotát, például egy „próbálgatós” kedvű, az OWA-ba jogosultság nélkül belépni óhajtó internetes vendég miatt.

Maradva a példánál: sokkal jobb, ha a tűzfal Exchange szervernek „hazudja be” magát, és már a határvonalon visszautasítja a jogosulatlan elérést, mint ha minden hitelesítési csomagot előzetes vizsgálat nélkül, nyomban továbbítana az Exchange szervernek.

Ezért van az, hogy egy – az ISA kiszolgálón keresztüli – hitelesítési folyamatban csak a jogosultságok érvényesítése után következik a jogosultsági adatok végleges ellenőrzése, amelyet delegál az ISA a megfelelő belső szerver felé. Viszont ezzel kapcsolatban is van fontos változás, mert míg az ISA Server 2004-ben csak a Basic hitelesítésénél élt ez a lehetőség, a 2006-os verziónál a Digest, az Integrated és a SecurID hitelesítési metódusnál is alkalmazható.

Sőt, a publikálószabályoknál a következő lehetőségeink vannak még ezeken kívül is:

- nincs delegálás és a kliens nem hitelesíthet közvetlenül;
- nincs delegálás, de a kliens hitelesíthet közvetlenül;
- kikényszerített Kerberos-delegálás.

A jogosultságok tárolása

Az ISA Server 2006 képes tárolni a Basic és az űrlap alapú hitelesítésnél használt felhasználói adatokat. Ha kérjük ezt a lehetőséget (alapértelmezésben tiltva van), akkor az ISA egy TCP sessionben egyszer, és kizárólag csak a legelső HTTP-kérés alkalmával követeli majd meg ezeket az adatokat. Ezt a módszert az Integrated (az AD és az LDAP-féle egyaránt) és a RADIUS hitelesítési mód is támogatja.

Gál Tamás
(gtamas@tjszki.hu) MCSE, MCSA, MCT, MVP

Mentés és visszaállítás

Egyre több, kritikusan fontos adatot tárolunk a számítógépeinken. Legyen szó akár egy önálló otthoni számítógépről vagy egy bonyolult informatikai rendszerről, elengedhetetlen, hogy biztosítsuk: a benne tárolt adatok ne vesszenek el.

Kritikusan fontos adatok nem veszhetnek el! Ennek a követelménynek a megvalósításához legfontosabb elem az, hogy rendszeres, megbízható adatmentéssel rendelkezzünk. Bár ezen a téren a Windows Vista és a „Longhorn" Server jelentős újdonságokat fog felmutatni, érdemes megnézni, hogy milyen eszközök állnak már most rendelkezésünkre a Windows Server 2003-ban. Ebben a cikkben a mentés és visszaállítás alapvető tervezési kérdéseinek áttekintésén túl bemutatunk előre elkészített scripteket is a mentési folyamat automatizálásának megkönnyítésére.

Tévhitek

Először azonban érdemes megismerkedni néhány tévhittel a mentéssel kapcsolatban:

„Megfelelő redundáns lemez-alrendszerrel rendelkezünk, tehát nem fordulhat elő adatvesztés." Ez az elképzelés alapjaiban téves. A mentést nem helyettesíti semmi. Egy redundáns rendszer növeli a rendelkezésre állást, ugyanakkor nem nyújt megoldást több előforduló problémára:

- adatok véletlen vagy szándékos törlésére;
- szoftverhibából adódó adatsérülésre;
- külső támadásra (cracker, vírus stb.);
- katasztrofális hardversérülésre.

„Van mentésem, tehát biztonságban vagyok." Ebben az esetben feltehetjük a kérdést: megpróbálta valaki, valamikor azt a mentést visszatölteni?

Vegyük tudomásul: az, hogy beállítottuk a rendszeres mentést, nem garancia arra, hogy ezt vissza is tudjuk állítani. A mentést rendszerben gondolkodva kell felépítenünk. Ennek a folyamatnak elengedhetetlen része a tesztelés. Nemcsak a mentést magát kell tesztelnünk, hanem a visszaállítást is. Ez a gondolat rögtön elvezet a mentési rendszer tervezése felé.

Tervezési szempontok

Ahhoz, hogy biztonságban érezhessük magunkat, el kell készítenünk és tesztelnünk kell egy katasztrófatervet. Ez a terv tartalmazza azokat a feladatokat, folyamatokat, amelyek képessé tesznek bennünket arra, hogy egy komoly rendszerhiba esetén, a legrövidebb időn belül újra működőképessé tudjuk tenni a rendszerünket, és biztonsággal vissza tudjuk állítani adatainkat. Néhány alapvető fontosságú szempont, amelyet figyelembe kell vennünk:

Mentési ablak. Az az idő, ami napi, heti rendszerességgel rendelkezésünkre áll a mentés lefuttatására. Manapság egyre nagyobb adatmennyiséget kell egyre rövidebb idő alatt menteni, ezért a mentési ablak gondos tervezést és tesztelést igényel.

Visszaállítási idő. Már a mentés tervezésénél figyelembe kell vennünk, hogy egy meghibásodás esetén mennyi idő alatt leszünk képesek működő állapotba visszaállítani rendszerünket. Ahogy a vállalkozások működésük során egyre jobban támaszkodnak az informatikára, egyre kevesebb az az idő, amit képesek működő számítógépek nélkül átvészelni.

Mit mentsünk, milyen rendszerességgel? Az informatikai rendszer felépítésekor végig kell gondolnunk, hogy milyen adataink, milyen rendszerességgel változnak. Mik azok az adatok, amelyeknek online elérhetőeknek kell ugyan lenniük, de esetlegesen már csak olvasásra használjuk őket, valamint hogy melyek azok az adatok, amelyekre csak ritkán van szükség. Egy ilyen jellegű adatelemzés nagymértékben segítheti a mentés tervezését, csökkentheti mind a visszaállítási időt, mind a szükséges mentési ablakot.

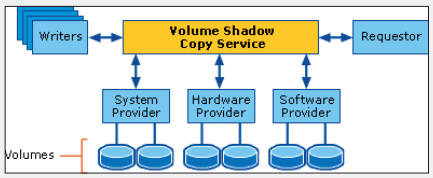
Mentések tárolása. Mi az az időpont a múltban, ameddig vissza kell tudnunk nyúlni? Hol helyezzük el a mentéseinket?

Adathordozó-választás. Nagyon sok szempontot kell itt figyelembe venni, azonban figyeljünk oda, hogy olyan adathordozót válasszunk, amire nemcsak ma, hanem a jövő-

ben is képesek leszünk elhelyezni a mentendő adatmennyiséget: megfelelő a sebessége, a visszakereshetősége, a megbízhatósága, a tárolhatósága, az újrahasznosíthatósága és természetesen a fajlagos költsége.

Mit mentünk?

Alapvetően meg kell határoznunk, hogy a rendszerünkben mik azok az elemek, amelyeket menteni szeretnénk. Ezek a dolgok alapvetően két kategóriába sorolhatók. Az egyik kategória az éppen zárt fájlok mentése. Ez az, ami különösebb gondot nem okoz, hiszen a mentési folyamat kapcsán nem csinálunk mást, mint a fájlról készítünk egy másolatot. A második



A Volume Shadow Copy Service felépítése

kategória a rendszer által használt adatbázisok és nyitott fájlok mentése. Ezt a feladatot nem tudjuk egyszerűen elvégezni. Az ilyen típusú adatok mentésére különböző technológiákkal és eszközökkel rendelkezünk. Ide sorolható a Volume Shadow Copy Service, a különböző mentési rendszerek speciális adatbázis agentjei, valamint a különböző export–import eszközök, amelyek ismerik az adott alkalmazás vagy szolgáltatás tárolási módszereit.

Az NTBackup

A Windows beépített mentőeszköze az NTBackup. Ennek az eszköznek a lehetőségeihez kétféleképpen férhetünk hozzá. Az egyik egy egyszerű grafikus felület, a másik egy parancssori felület. A továbbiakban elsősorban ez utóbbiról lesz szó. Először vizsgáljuk meg, hogy milyen feladatokra is alkalmas ez az eszköz!

- **A kiválasztott fájlok és mappák mentése/visszatöltése.**
- **Másolat készítése a rendszerállapotról (System State).** A rendszerállapot olyan adatokat tartalmaz, amelyek fájlszinten nem érhetők el, mert általában folyamatosan nyitott adatbázisokban tárolódnak. Ide tartozik például a registry. Ezeknek az adatoknak a mentéséről azért kell gondoskodnunk, mert egy teljes rendszer-visszaállítás nem képzelhető el nélkülük.
- **Automated System Recovery (ASR).** Ez egy olyan technológia, amelyik teljes rendszer-visszaállítás esetén gondoskodik az operációs rendszer minél gyorsabb megfelelő állapotba hozataláról: így válhat lehetővé az adatok visszatöltése.
- **Remote Storage és felcsatolt lemezek mentése.**
- **Logfájl készítése a mentési műveletről.**
- **Rendszerpartíció, boot-partíció, valamint a rendszerindításhoz szükséges fájlok mentése.**
- **Mentések időzítése.** Az ütemezett feladatok (Scheduled Tasks) segítségével képes a különböző mentési feladatokat időzíteni. Saját felülettel rendelkezik e feladatok kezelésére.
- **Médiakezelés.** Cserélhető média esetén – mint amilyen például a szalag – képes a szükséges feladatokat (media pool kezelése, szalagformázás stb.) ellátni.
- **Online adatbázison alapuló Microsoft-termékek adatainak mentése.** Képes az összes olyan online adatbázis mentésére, amelyik kompatibilis a Volume Shadow Copy Service-szel.

Ezeken kívül azonban rengeteg olyan mentési feladat is van, amire az NTBackup nem alkalmas. Ilyen feladat például a nem Microsoft gyártmányú, VSS-szel nem kompatibilis online adatbázisok mentése.

Vannak ugyanakkor olyan hiányosságai is az NTBackupnak, amelyeket kiegészítésekkel, ötletekkel orvosolni tudunk; ezek megvalósítására születtek a különböző scriptek.

A mentés típusai

Azt, hogy a fájlok mentésének kapcsán mit kell mentenie az NTBackupnak, alapvetően a fájlok két paramétere határozza meg. Az egyik a fájl utolsó módosításának időpontja, a másik az úgynevezett archive bit. Ez utóbbit minden, a fájl módosításával járó művelet köteles bekapcsolni. Ezt a bitet törli a mentőalkalmazás mentésnél, amivel a fájlt mentettnek jelöli.

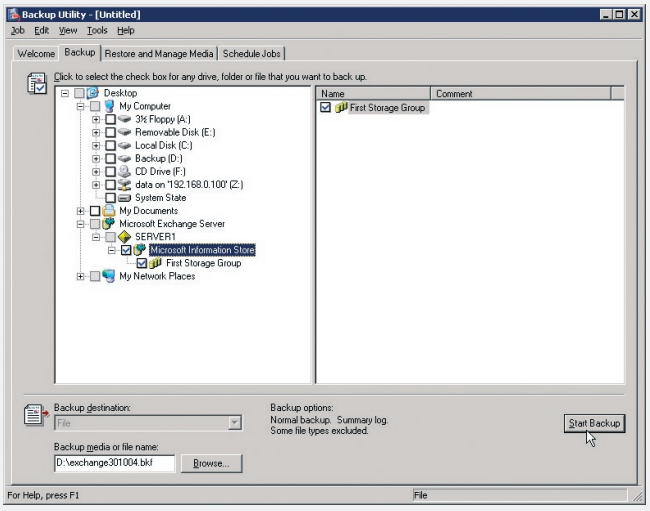
- **Copy backup (másolat).** Minden kijelölt fájlról másolatot készít, ugyanakkor nem jelöli mentettnek a fájlokat.
- **Daily Backup (napi mentés).** A kijelölt fájlok közül azokról készít mentést, amik a mentés futtatásának napján módosultak. Nem jelöli mentettnek a fájlokat.
- **Differential Backup.** Az utolsó normal vagy incremental típusú mentés óta módosult fájlokat menti. Nem jelöli mentettnek a fájlokat.
- **Incremental Backup.** Az utolsó normal vagy incremental mentés óta módosult fájlokat menti. Mentettnek jelöli a fájlokat.
- **Normal Backup.** Minden kijelölt fájlról másolatot készít. Mentett jelzéssel látja el a fájlokat.

NTBackup parancssorból

Mentéseink során három kérdést kell tisztáznunk. Mit mentünk, hogyan mentünk és mikor mentünk? A továbbiakban bemutatott megoldások alapvetően merevlemezre történő mentéshez készültek.

Ha szalagos mentést választunk, akkor a bemutatott példákat, scripteket ki kell egészítenünk a szalagos adattároló kezeléséhez szükséges lehetőségekkel. Vizsgáljuk most meg a fent említett hármast!

Mit mentünk? Azt, hogy mit mentünk, egy úgynevezett selection fájlban adhatjuk meg. Ez egy egyszerű szövegfájl, amelyik sorónként tartalmazza a mentendő meghajtók, könyvtárak, fájlok elérési útvonalát. Lehetőségünk van arra is, hogy egy megadott nagyobb egységből kisebb szeletek mentését



Az NTBackup grafikus felülete

tiltsuk. Ez utóbbit a sor végére tett /exclude kapcsolóval tudjuk megtenni.

Példa egy egyszerű selection fájlra:

```
c:\
c:\recycler /exclude
d:\
d:\recycler /exclude
```

A fenti példa menti a rendszer C: és D: meghajtóját, kihagyva mind a két meghajtóról a lomtárat. A selection fájlnak kötelezően Unicode formátumúnak kell lennie, így akár egy egyszerű notepaddel is előállíthatjuk, azonban ennek hibátlan működéséhez szükségünk lesz a Windows Server 2003 SP1-re, ellenkező esetben trükközni kell az elmentett szövegfájl Unicode-fejlécével.

Hogyan mentünk? A mentés különböző paramétereit parancssori kapcsolókkal tudjuk befolyásolni. A parancssor szintaxisát a legegyszerűbb, fájlba történő mentésen keresztül értelmezzük:

```
ntbackup backup „@selection fájl neve" /F „célfájl neve" /L:s /M normal
```

backup – megadjuk, hogy a feladat, amit el akarunk végezni, az a mentés;

„@selection fájl neve" – annak a fájlnak a neve, amelyik meghatározza, hogy miket mentünk;

/F „célfájl neve" – annak a fájlnak a neve, ahová történik a mentés;

/L:s – meghatározza, hogy mi kerüljön az NTBackup logjába. Ebben az esetben csak összefoglaló információt jelöl.

/M normal – normál mentés (ez lehet még copy, incremental, differential vagy daily).

Mikor mentsünk? Ezt a kérdést is megközelíthetjük több irányból. Az egyik lehetőségünk, hogy az előzőekben összeállított parancssort, egyszerűen a grafikus felületen keresztül felvesszük az ütemezett feladatok (Scheduled Tasks) közé, a másik, hogy a parancssorból az schtasks parancsot paraméterezve tesszük ezt meg.

Az NTBackup automatizálása

Különböző scriptekkel akár még azt is meg tudjuk oldani, hogy a mentési művelet befejeződéséről e-mailt kapjunk. Itt rögtön felmerülhet kérdésként, hogy mi is legyen ebben az e-mailben.

Alapvető problémát okoz, hogy az NTBackup működéséről készült log egy igen eldugott helyen található a rendszerben, valamint az, hogy nem minden információ található meg benne, ugyanis sok, a működéshez kapcsolódó információ a rendszer eseménynaplójába kerül. Felmerül még elvárásként, hogy az NTBackup parancssorát kicsit egyszerűbben lehessen kezelni.

Az itt megfogalmazott elvárások teljesítésére készítettünk három olyan JScript-osztályt, amelyeket felhasználva kifejezetten egyszerű scriptek készíthetők a mentési feladatok megoldására. Ezek a scriptek példakódokkal együtt szabadon elérhetők a magyar TechNet-oldalon.

- **Backup.js.** Kezeli az NTBackup parancssori paramétereit, valamint összegyűjti a generált logjait.
- **Mail.js.** Képes levelet küldeni a Collaboration Data Objects segítségével.
- **EventLog.js.** Begyűjti a saját működésének idején keletkezett eseménynapló-bejegyzéseket.

Lássunk rögtön egy példát virtuális gépek mentésére ezek használatával!

A Virtual Server 2005 R2 mentése

Az informatikai infrastruktúrában egyre jobban terjed a virtualizáció, azonban a virtualizáció bevezetése magával hozza a mentés problematikáját is.

Alapvetően kezelhetjük a virtuális gépeket úgy is, mint a fizikaiakat; ilyenkor a vendégoperációsrendszerben, az azon lévő eszközökkel oldhatjuk meg a mentést.

A másik lehetőségünk, hogy a virtualizálásra szolgáló szoftver, esetünkben a Microsoft Virtual Server irányából közelítjük meg a feladatot. Ebben az esetben mindössze az egyedi virtuális gépeinkhez tartozó néhány fájlt kell mentenünk.

Miután ezek a fájlok a vendég-operációsrendszer mentése idején használatban vannak, kézenfekvő megoldásnak tűnik egy Volume Shadow Copy Service alapú mentés. Ezt sajnos ma még nem alkalmazhatjuk, mert a Virtual Server mai verziója (2005 R2) még nem támogatja a megoldást, azonban az SP1 már orvosolja ezt a hiányosságot is. Az SP1 a cikk írásának pillanatában Beta 2 kiadásban érhető el.

A fentiek miatt a mentésre ma még más megoldást kell választanunk. Ez az egyedi virtuális gépeknél a következő lépésekből áll:

- elmentjük a virtuális gépet (save state), vagy leállítjuk;
- másolatot készítünk a virtuális gép által használt fájlokról;
- visszaállítjuk az eredeti állapotot.

Ezt a folyamatot elvégezhetjük kézzel, a grafikus felület használatával, vagy miután a Virtual Server rendelkezik megfelelő programozói felülettel, akár scriptből automatizálhatjuk is. Az automatizációhoz készítettünk egy JScriptet (VSBackup.js), amelyik szintén illeszkedik a korábban ismertetett csomagba, és könnyebbé teszi számunkra az automatizációt. A script használatára álljon itt egy példa:

```
<?xml version="1.0" encoding="utf-8" ?>
<package xmlns="http://schemas.microsoft.com/
WindowsScriptHost">
  <job>
    <script language="JScript" src="Mail.js"/>
    <script language="JScript" src="EventLog.js" />
    <script language="JScript" src="VSBackup.js" />
    <script language="JScript">
      var mail = new Mail();
      var el = new EventLog(elLogFileApplication +
                elLogFileSystem +
                elLogFileVirtualServer);
      var vs = new VSBackup();
      mail.MailServer = „mail.test.local";
      mail.From = „VSBackup@test.local";
      mail.To = „administrator@test.local";
      mail.Subject = „Test Backup";
      vs.BackupPath = „c:\\backup\\20060921.VS";
      vs.BackupAll();
      vs.AttachLog(mail);
      el.AttachLog(mail);
      mail.Send();
    </script>
  </job>
</package>
```

A mentés és visszaállítás témakörét ennél részletesebben is bemutattuk nemrég egy TechNet-webcast alkalmával, ami mellett online elérhetőek az itt bemutatott scriptek és példák, valamint útmutatás található az SQL Server, az Exchange 2003, a DHCP és a RIS szolgáltatások mentéséről és visszaállításáról is. Mindezek az információk elérhetők a http://www.microsoft.hu/technet/ntbackup linken.

Gömöri Zoltán
(suf@freemail.hu)

IT-BUSINESS TODAY

goes mobile ...

Az IT-BUSINESS TODAY SMS küldésével WAP rendszerben már mobiltelefonon is elérhető, így számítógép segítsége nélkül is elérhetővé válnak a legfontosabb ICT-piaccal kapcsolatos hírek, történések, események.

Próbálja ki most!

Küldje el az ITBTODAY szót SMS-ben a +36 30 285 5441 számra és kövesse az instrukciókat!

WAP cím: **wap.it-business.hu**

BIZTONSÁGOS ARCVONAL

Védelem üzleti alapokon.

Az egyre üzletcentrikusabb fenyegetések miatt a szervezetek is mind gyakrabban ébrednek rá, hogy a védelemnek is üzleti szemléletűnek kell lennie. A Microsoft 2006 júniusában bejelentett Forefront koncepciója olyan termékcsalád, amelyet kifejezetten az üzleti rendszerek védelmére szánnak. A „családtagok" könnyen integrálhatók egymással és a szervezet meglévő IT-rendszerével, valamint kiegészíthetők és együtt is működnek más fejlesztők technológiáival. Az alábbi cikkben azt tekintjük át, mik is várhatóak pontosan a Microsofttól a következő egy évben a biztonsági szoftverek terén, és mik a mozgatórugói ennek a kezdeményezésnek.

A hálózat határainak védelme

A Forefrontban a perembiztonság és a hozzáférés feletti ellenőrzés az ISA 2006 feladata. Az ISA integrált biztonsági átjáróként védi a belső környezetet az internet fenyegetéseitől, és a jogosult felhasználók számára lehetővé teszi a belső erőforrások gyors és biztonságos elérését. Az ISA szolgáltatásait a TechNet Magazin korábbi számaiban már bemutattuk. A Forefront filozófiájában az alábbi alapvető forgatókönyvek szerint használható az ISA:

- hozzáférés biztosítása külső felhasználók számára a hálózat belső alkalmazásaihoz, alkalmazáspublikáció – Exchange, SharePoint, webes alkalmazásszerverek;
- átjáróként, hogy a távoli munkahelyek hozzáférhessenek a központi hálózathoz;
- biztonsági képességei révén alkalmas a külső vagy belülről induló fenyegetések elleni védelemre (itt érdemes megemlíteni, hogy az ISA 2006 már a forgalmi anomáliák alapján is fel tud ismerni egy elosztott szolgáltatásmegtagadási – DDOS – vagy féregtámadást, annak ellenére hogy protokoll szerint a forgalom teljesen szabályos, és nem tartalmaz káros kódot, de az adott protokoll vagy IP-cím által generált forgalom eltér a szokványostól).

A család legújabb tagja, a Whale Communicationstől származó IAG (Internet Application Gateway) felel az SSL–VPN kapcsolatért, védi a webes alkalmazásokat, és a meglévő felügyelt és nem menedzselt végpontok kavalkádjában is gondoskodik a végpontbiztonságról.

A szerveralkalmazások védelme

A szervezetekben szükség van az üzlet szempontjából kritikus üzenő- és csoportmunkarendszerek vírusok, férgek, kéretlen küldemények elleni védelmére. A Microsoft Forefront Security for Exchange Server (FSES), Security for SharePoint és Security for Office Communications Server triásza védi azokat a kiszolgálókat, amelyeken az Exchange Server, a Windows alapú SMTP-átjárók, az Office Communications Server és a SharePoint-szolgáltatások működnek.

Az együttes előnyök:

- a több pásztázóeszköz több rétegben ellenőrzi a kommunikációs struktúra biztonságát;
- a szoros integráció a Microsoft-szerverekkel fokozza a hozzáférhetőség és a felügyelet hatékonyságát;
- a tartalom-ellenőrzés miatt a külső és belső kommunikációban is kiküszöbölhetők a helytelen tartalmak és a veszélyes küldemények.

Az egyedi Forefront védelmi megoldások önmagukban is hatékonyak, de ha a réteges védelem miatt valamelyik szinten mégis átcsúszna egy támadás, akkor egy további rétegben jó eséllyel blokkolódik.

A Forefront védelmében nem kevesebb, mint kilenc(!) kártevőellenes motort lehet használni: Microsoft (Sybari), CA Inoculate-IT, CA Vet, Norman, Sophos, Authentium, Kaspersky, a magyar VirusBuster és AhnLab. Mindegyik technológiának megvan az erőssége, és a Forefront védelme úgy épül fel, hogy erősítik egymást. A motorokat természetesen együtt is lehet használni, de ötnél többet nem célszerű kombinálni a védelem és teljesítmény egyensúlya miatt.

A Forefront termékvonal elemei

Az egységes rendszer a következő megoldásokból áll össze (zárójelben az adott technológia korábban érvényes elnevezése):

- Internet Security and Acceleration (ISA) Server 2006;
- Forefront Security for Exchange Server (Antigen for Exchange);
- Forefront Security for SharePoint (Antigen for SharePoint);
- Forefront Security for Office Communications Server (Antigen for Instant Messaging);
- Forefront Client Security (Microsoft Client Protection).

Az FSES a vírus- és féregszűrésen túl mind a csatolt állomány típusa, mind annak tartalma alapján képes elemezni, és a megfelelő házirend szerint engedélyezni, törölni vagy karanténba zárni a beérkező üzeneteket.

Csatolt állomány szűrési lehetőségei. Valós állománytípus alapú szűrés (a csatolt állomány bináris fejléce egyértelműen meghatározza annak típusát, kiterjesztéstől függetlenül):

- állománykiterjesztés alapú szűrés;
- állománynév alapú szűrés;
- állományméret alapú szűrés.

Kulcsszó alapú tartalomszűrés. A szűrés történhet sima szöveg vagy HTML szöveges üzenet törzsében, a fejlécben vagy a tárgymező adata alapján. Ha mind a sima, mind a HTML szöveg tartalmazza a keresett kulcsszót, két különböző riportbejegyzés történik. Megadható, hogy hány kulcsszóegyezés esetén hajtódjon végre a megadott akció (karantén vagy törlés).

Kétszintű levélszemétszűrő rendszer

Alapvetően két nagy csoportra oszthatjuk a levélszemétszűrő technológiákat. A minta alapú rendszerek adott (már ismert) levelet, annak egy részét vagy szófordulatot keresnek. Ezt a „levélszemétgyártók" különböző technológiákkal próbálják megkerülni (a teljesség igénye nélkül lássunk kettőt):

Hash busting. A hash alapú szűrők kijátszását célozza. A hash-minta alapú szűrők az interneten felállított érzékelőkön áthaladó üzenetekről egy hash-t (ujjlenyomatot) vesznek és ezt hasonlítják össze a postafiókba érkező üzenettel. A hash-képzés matematikájának köszönhetően az eredeti üzenet tartalma nem fejthető vissza (így személyiségi jogokat nem sért), de egyben ez a gyenge oldala is. Az eredeti üzenet néhány karakterrel vagy egy néhány pixeles képpel kiegészítve már másik hash-t ad. A spammerek pont ezt használják ki, és több ezer (néhány karakter vagy pixel megváltoztatásával) verziót küldenek szét.

Snowflaking (hópihe-módszer). Akárcsak a hópehely, minden üzenet lehet egyedi, néhány apró rész megváltoztatásával vagy hozzáadásával. A hash bustinghoz hasonlóan itt is az a cél, hogy a minta alapú szűrőn ne akadjon fenn a levél. Fontos tudni, hogy minden spamszűrő pontozza az üzeneteket. Vannak olyan elemek, amelyek egyértelműen spammé, míg mások egyértelműen nem spammé minősítik a levelet. Sok esetben a nem spam mindősítés felülírja a spam jelölést, hiszen „inkább csorogjon be egy spam, mint hogy kitöröljünk egy fontos üzenetet". Ezt használja ki a snowflaking, amely HTML üzenetek esetében rejtett, láthatatlan, fehér szöveget helyez el a sorok között. Mivel a spamszűrők sok esetben gyanúsan kezelik a fehér alapon fehér szöveget, a spammelők sok esetben a 65 535 színkombináció közül a fehér –1 ... 8 árnyalatot választják. A láthatatlan szöveg pedig üzleti szöveg (például: Hivatkozva a legutóbbi megbeszélésre stb...), aminek hatására a spamszűrő „nem spam" mindősítést ad a levélnek, annak ellenére, hogy a levéltörzs többi része lehet, hogy Viagra vásárlására buzdít.

Vannak olyan spamszűrők, amelyek az e-mail feladója alapján döntik el, hogy kéretlen-e az adott levél. Nincs azonban jelenleg garancia arra nézve, hogy a levél valóban attól származik, aki feladóként fel van benne tüntetve. Egyre inkább terjed a „spoofing", vagyis az e-mail feladójának meghamisítása, mert a kéretlen levelek feladói viszonylag egyszerűen kijátszhatják a szűrőket. Ennek kivédésére született meg a üzenetet feladó tartomány ellenőrzése – lényegében úgy, ahogyan a telefonok hívóazonosítójának köszönhetően megjelenik a hívó fél telefonszáma.

- Az e-mailt küldő kisebb vagy nagyobb szervezetek közzéteszik kimenő e-mail-kiszolgálóik IP-címét a DNS (tartománynév) rendszerben, az „e-mail-küldőazonosító" specifikációjában előírt formátumban.
- A fogadó e-mail-rendszerek minden üzenetet megvizsgálnak, hogy megállapítsák, milyen tartományból valónak állítja magát az üzenet (azaz, hogy milyen internetes tartomány van feltüntetve az üzenetben feladóként).
- A fogadó e-mail-rendszerek lekérdezik a DNS-ből az állítólagos feladótartomány kimenő e-mail-kiszolgálóinak IP-címeit. Ezután ellenőrzik, hogy szerepel-e a listán az az IP-cím, amelyről a levél érkezett. Ha nem, akkor az üzenetet nagy valószínűséggel hamis címről adták fel.

A másik nagy csoportot a heurisztikus szűrőt tartalmazó megoldások alkotják, ahol szabályok és algoritmusok, egyfajta „mesterséges intelligencia" használatával próbálják eldönteni, hogy a vizsgált üzenet levélszemét vagy sem. Mint minden szabályon alapuló hipotézisre, erre is igaz az a mondás, hogy „Kivétel erősíti a szabályt!".

A heurisztikus szűrők a folyamatosan változó levélszemét-áradattal csak részben képesek megbirkózni, hiszen ahogy az emberi viselkedés sem írható le teljes egészében szabályokkal, a változó levélszemét sem (amit, ugye, emberek gyártanak).

Érdemes még egy kérdést megvizsgálni! Hogy kerülünk fel a spammerek küldőlistájára? Míg korábban a levélszemetelők valamennyi lehetséges címre (válogatás nélkül) elküldték „ajánlatukat", addig manapság már sokkal célzottabban küldik azokat, kifejezetten az általuk kiszemelt célcsoportnak. Ez az egyik oka annak, hogy a minta alapú (hash-ujjlenyomat) rendszereknél használt internetes csapdák csak a spamek egy részének kiszűrésére alkalmasak. Jogosan merülhet fel a kérdés, honnan veszik a spammerek a címeket? Lássunk néhány példát:

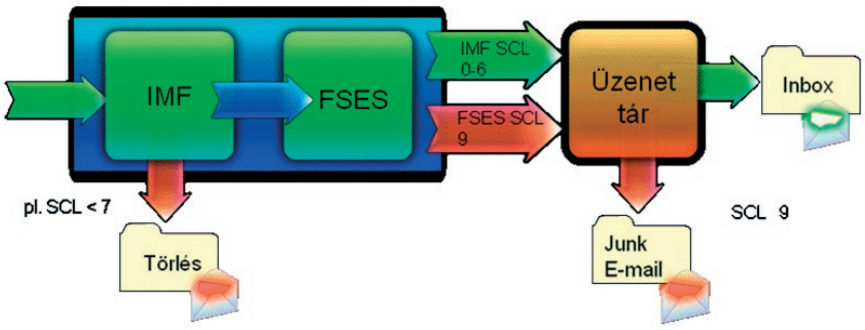
Doménpróbálgatás. A kéretlen levél küldője kiválaszt egy adott domaint (például contoso.com), majd ismert nevekkel párosítja (például: kiss.gyula, k.gyula, kgyula, gyula.kiss@contoso.com). Minden olyan levél, ami nem jön vissza „címzett ismeretlen" hibaüzenettel, potenciálisan valós cím.

Továbbá ha valaki bekapcsolja a „házon kívül" szolgáltatást külső címre, a spamgyáros még megerősítést is kap hogy ki a cím gazdája. Sok felhasználó ugyanis büszkén sorolja valamennyi elérhetőségét aláírásában, amit odailleszt még az automatikus „házon kívül" válaszok végére is.

Intelligens keresőmotorok. Sajnos a keresőmotorok hatékonyságát sokan rosszra használják, és a téma ráadásul tetemes szak-

irodalommal is rendelkezik (Google hacking). Az ismert keresőmotorok jól paraméterezhetők, és nagy pontossággal keresnek ki valós e-mail-címeket fórumokból, nyilvános adatbázisokból, cégek weboldalairól, sajtóanyagokból vagy éppen konferenciák webes tájékoztatóiból.

A folyamatos rabló–pandúr harcban a levélszemétgyártók újabb és újabb eszközöket vetnek be, de a védelmi technológiák gyorsabb ütemben fejlődnek, és ami még ennél is fontosabb, hatékonyságuk sokkal jobb, mint korábban, így a tévesen levélszemétnek minősített (falsh positive) levelek száma – a minta és heurisztikus szűrési technológiák kombinált használatával – 1 százalék környékére csökkenthető. Sokan azt mondhatnák: ez már egész jó érték, azonban nem szabad elfelejteni, hogy a elektronikus levelek és ezen belül a levélszemét mértéke exponenciális értékben növekszik évről évre. Egymillió üzenetet feltételezve már az 1 százalék is soknak mondható. Mint minden biztonsági megoldásra, erre is igaz, hogy réteges védelemre van szükség: „Egy zár – nem zár", azonban a két technológia egyidejű alkalmazása jó hatásfokkal szűri ki a növekvő ütemben érkező kéretlen levelek hadát.



Az FSES és az IMF közös SCL taget használ

Az FSES egy minta alapú levélszemétszűrő motorral egészíti ki az Exchange 2003-ban már megismert, heurisztikus elven működő IMF (Intelligent Message Filter) szűrőt.

Az FSES motorja három alapvető funkciót tartalmaz:

- a Bullet SignatureDatabase-t, amely lehetővé teszi a spamküldők és leveleik felismerését, illetve a Spammer Trick Analysis and Response rendszert, amelyik felismeri a spamküldők módszereit és trükkjeit;
- a SpamCure motor kilenc üzenet-, fejléc- és forgalomelemzési kategóriát tartalmaz a DNS-információ, a tárgymező, az IP-cím, a küldő és a levéltörzs-információk alapján, a motort úgy alakították ki, hogy adatbázisát napi több alkalommal tartják karban és frissítik;
- IMF–FSES együttműködés: az FSES és az IMF közös SCL taget használ.

Az FSES három különböző üzenetmegjelölési (tag-) módszert alkalmaz.

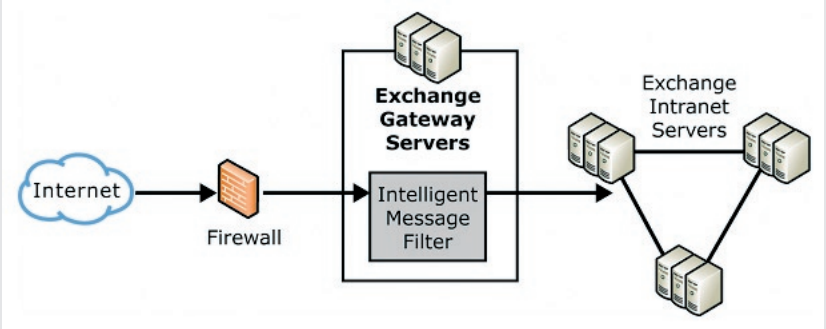
Tárgy-tag. Adott szöveggel (például „levélszemét") egészíthető ki a levél tárgysora, ami alapján – ha erre van szükség – kliensoldali szabály használatával lehet a „junk mail" mappába irányítani a levelet.

Üzenetfejléc-tag. Hasonló módon a tárgysori jelöléssel, itt az üzenet fejlécsora módosítható.

SCL-felülírás. Az IMF által megadott spam-valószínűségi mutató felülírása (csak + SCL -szintemelés lehetséges).

Az Intelligent Message Filter működése

Amikor egy külső felhasználó e-mailt küld egy olyan Exchange-kiszolgálónak, amelyre telepítve van az Intelligent Message Filter, az Intelligent Message Filter megvizsgálja az üzenetek szövegét, és egy SCL-pontszámot rendel az üzenethez, amely azt jelzi, hogy az üzenet milyen valószínűséggel kéretlen levél. Ezt a pontszámot az üzenet egy tulajdonságában tárolja, amelynek spam-valószínűségi szint (spam confidence level, SCL) a neve. A pontszám tartósan része marad az üzenetnek, még akkor is, ha más Exchange-kiszolgálókra továbbítja őket a rendszer.



Tipikus Exchange Server-topológia

A rendszergazda két küszöbértéket állíthat be arra vonatkozóan, hogy hogyan kezelje az Intelligent Message Filter a különböző SCL-pontszámot tartalmazó leveleket: egy átjáró-küszöbértéket, amelyhez egy, a küszöbértéket meghaladó üzenetekre vonatkozó művelet társítható, valamint egy postafióktár-küszöbértéket. Az átjáró küszöbértékénél nagyobb pontszámú üzenetek esetén az Intelligent Message Filter végrehajtja az előírt műveletet (például a törlést). Ha az üzenet pontszáma kisebb az átjárón érvényes küszöbértéknél, a kiszolgáló továbbítja az üzenetet a címzett postaládáját tároló Exchange-kiszolgálónak. Ha az üzenet pontszáma nagyobb az Exchange Server postafiók-tárolójára vonatkozóan megadott küszöbértéknél, a postafiók-tároló nem a felhasználó Beérkezett üzenetek mappájába, hanem a Levélszemét mappába kézbesíti az üzenetet.

Kliens- és platformvédelem

A Forefront Client Security a Windows alapú kliens- és szerver-operációsrendszereket használó számítógépeket védi. Ugyanolyan hatékony a „jó öreg" vírusok, férgek és trójaiak elleni védelemben, mint a „modern" kémprogramok és a bujkáló rootkitek ellen. Természetesen ez is integrálható a meglévő infrastruktúrához – például az Active Directoryhoz – és a többi biztonsági megoldáshoz. A Client Security nyilvános próbaváltozata 2006 vége felé várható.

Kelemen László
kelemen@hungary.com

Messze vagyunk még? Messze...

Legutóbbi cikkünkben a párhuzamosság alapvető kérdéseivel foglalkoztunk, most pedig megnézzük, hogyan lehet a szerverszoftverek skálázhatóságát alapul véve megoldást találni a kliensszoftverek teljesítményének javítására.

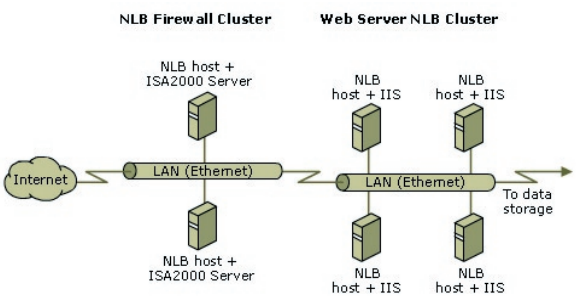
Jelenleg azok a legjobban skálázható szoftverek, amelyek felhasználók és lekérések ezreit szolgálják ki – látszólag teljesen egyszerre. Az ok egyszerű: eddig igazán csak ezeknél volt kérdéses a skálázhatóság, itt foglalkoztak a témával a kellő alapossággal.

Ilyen, már most is skálázhatóságra tervezett szoftverek például a web- és alkalmazásszerverek (például az IIS), valamint az SQL Server, és ide tartoznak olyan keretrendszerszintű megoldások is, mint például a .Net CLR thread poolja, végül pedig a tudományos számítások elvégzésére is alkalmas fürtözött HPC- (High-Performance Computing) rendszerek, mint például a Windows Server 2003 Compute Cluster Edition. De mitől is olyan skálázhatóak ezek, és miben különböznek egymástól?

Melyek a skálázhatóságot befolyásoló tényezők?

A szerverszoftverek általában lekérdezések kiszolgálására vannak optimalizálva. A lekérdezésekkel szemben a következő általános igények fogalmazódnak meg:

- Mennyi az egyszerre feldolgozandó lekérdezések száma?
- Mekkora a lekérdezések elvárt válaszideje?
- Mekkora a lekérdezések átlagos feldolgozásigénye, mennyire terheli le egy ilyen lekérdezés a rendszer egészét (milyen komplex egy átlagos lekérdezés)?
- Mekkora adatmennyiségre van szükség a feldolgozás során (közös erőforrásokról olvasva, illetve a lekérdezés saját paramétereit tekintve)?
- Mennyire kell friss, aktuális adatokkal számolni a feldolgozás során (mennyire használhatunk cache-elést, mennyire tudjuk kihasználni a lokalitás elvét)?
- Mennyire függnek egymástól időbeli egymásutániság vagy adatfüggőségek miatt az egyes lekérdezések, vagyis mennyire párhuzamosíthatók (lockolás, időbeli szinkronizáció)?



IIS webszerver-farm terhelésmegosztással és ISA Serverrel

- Milyen közös erőforrásokat használnak a függő vagy független lekérdezések (sávszélességigények, adat-dependenciák)?
- Történik-e írási művelet közösen használt erőforrásokba, és az befolyásolja-e a soron következő, adott sessionön belüli lekérdezések működését (stateful vagy stateless)?
- Ha történik írási művelet a közös erőforrásokba, milyen ACID izolációs szintre van szükségünk?

Nem fogunk minden egyes kérdést részletesen vizsgálni, hiszen hatalmas mennyiségű, tapasztalati úton és kutatással szerzett információ áll mögöttük. Ami igazán fontos minden esetben, hogy az egyes szerveralkalmazás tervezésekor vagy egy arra épülő megoldás készítésekor pontosan el tudjuk dönteni, milyen teljesítményigényeink vannak, és tudnunk kell, milyen lehetőségeink állnak rendelkezésre az optimalizációhoz. Ezek után nekünk kell megtalálnunk az egyensúlyt az egyes igények és a lehetőségek között, és ez egyáltalán nem egyszerű feladat.

A webszerver és az adatbázisszerver esete

Nézzünk pár példát! A webszerverek az esetek többségében egy közös lemezalrendszerről dolgoznak (vagyis a közös erőforrás jelen esetben a lemezen található HTML- vagy dinamikus webes állományok), a lekérdezése-

ket azonban párhuzamosan több szerverszámítógép és azon belül több processzor is kiszolgálhatja. Jellemzőek rájuk az aszinkron, stateless lekérdezések, legalábbis ha csak adott weboldal lekérdezéséről van szó, amit a kliens meg szeretne jeleníteni, mondjuk egy böngészőben. Ezeknél a lekérdezéseknél a cache-elés is erőteljesen használható, és az nem csak a szerver, de a kliens oldalán is erősen megjelenik (képek, statikus weblapok lokális cache-elése).

Ha szükségünk van arra is, hogy adatokat módosítsunk egy webes lekérdezés következtében (mondjuk egy online bolt esetében, amikor a felhasználó rendel valamit, és megadja adatait, valamint lefoglal az árukészletből), akkor azt ideálisan egy háttéradatbázis segítségével tehetjük meg, ami már a webszervertől eltérő módon skálázható. Emellett persze használhatjuk akár a webszerver saját (stateful) session-kezelését is, ez azonban ismét teljesen más esetet eredményez, hiszen több a közösen használt erőforrás, nagyobb a lekérdezés memóriaigénye, és ilyen esetekben nagyon oda kell figyelni, hogy egy session minden egyes lekérdezése ugyanarra a szerverre érkezzen be – ebben az ISA Server tud például segíteni.

Mi az, ami itt végül is párhuzamosítható volt? Láthatóan nincs igazán jó megoldás webszerverek esetében egy adott lekérdezés gyorsítására több processzormag felhasználásával; ugyanakkor viszont képesek vagyunk egy időben akár több ezer lekérdezés kiszolgálására is úgy, hogy a lekérdezéseken a szerverek és a processzorok valóban párhuzamosan dolgoznak.

Az SQL Server esetében jellemzően processzoronként egy-két szál fut csak, amelyek mind egy adott lekérdezést szolgálnak ki. Az SQL Server célja, hogy egy adott lekérdezést mielőbb végre tudjon hajtani, ugyanis egy adott lekérdezés vagy tranzakció állapotának kezeléséhez igencsak sok memóriára lehet szükség, és egyáltalán nem kedvező, ha ezt a memóriaterületet szálváltogatásokkor kell felcsorgatni a felsőbb cache-ekbe, vagy ha egyszerűen megtelik a fizikai memória. Az SQL Server – mint ahogy a legtöbb adatbázis is – alapvetően stateful működésre van felkészítve, de azon belül szinte minden kontroll a fejlesztő kezében van: a tranzakciók kezelése révén, a számunkra szükséges izolációs szint kiválasztásával mi magunk dönthetjük el, hogy a teljesítmény vagy az ACID elvek szigorú betartása a fontosabb számunkra.

Az adatbázisszerverek kivételesen jól tudnak scale up irányban nőni, akár egészen 64 processzorig is. A Microsoft SQL Server azonban jelenleg csak akkor képes igazán scale out irányban nőni, ha az alatta lévő adatok lehetővé teszik (például az adatok adott szisztéma szerint particionáltak egy fürt szervereire, ami módot ad több felhasználó más adatbázis-partícióból történő egyidejű kiszolgálására).

Az SQL Server egy adott lekérdezést is képes több processzormag között megosztani, ha az számításai szerint valóban gyorsabbá teszi a feldolgozást; de az sem okoz problémát, ha több, különböző lekérdezést kell párhuzamosan futtatnia az egyes magokon.

A lokalitás elve

A lokalitás lényege, hogy a számítógépekben és nagyobb, elosztott rendszerek esetében is az a lehető leghatékonyabb, ha a feldolgozandó, illetve az adott feladat elvégzéséhez szükséges adatok a lehető legközelebb vannak a feldolgozó egységhez, vagyis a processzormaghoz.

Minél közelebb van az adat, annál gyorsabban éri el azt a processzormag, illetve gyorsabban juttatható el a processzormag közvetlen közelébe. A leggyorsabbak a regiszterek, majd az adott mag vagy a processzor közös használatú L1–L2–L3 cache-e, utána sorban következhet a szál saját lefoglalt memóriaterülete, a más szálak által lefoglalt és közös memóriaterületek, majd a virtuális, illetve kiswappelt memória, és végül maga a merevlemez. Ha többgépes rendszereket vizsgálunk, akkor még a hálózati sebesség és távolság függvényében a távoli rendszereken található adatok is szóba kerülhetnek, vagy éppen a hálózaton vagy üvegkábelen keresztül bekötött közös tárolóeszközök is.

Alapvető cél, hogy minden adatot, amire szükségünk van, a feldolgozáshoz a lehető legközelebb tároljuk (és olyan formában, ami a feldolgozást minél rövidebbé teszi, transzformációs lépéseket is feltételezve az adatmozgatáson túl). Ezenkívül arra is fel kell készülnünk, hogy frissítsük is ezeket a lokális adatokat, ha szükséges (mivel az eredeti adat már megváltozhat, miközben a gyorsabb memóriaterületen már nem aktuális adatokat tárolunk a feldolgozás számára). Ennek optimális megoldása nem egyszerű feladat, és általában leegyszerűsítve egységesen a „cache-elés" névvel illetjük. Ennek a cache-elésnek hardveres és szoftveres megoldásai is vannak. A szoftveres megoldások többségével a fejlesztőknek folyamatosan foglalkozniuk kell, de felügyelt kódban a Garbage Collector és más háttérszolgáltatásoknak hála lényegesen kevesebbet, mint natív kód esetében. De még felügyelt kódban is oda kell figyelni a lokalitásra, ha a skálázhatóság és a teljesítmény további növelése a cél – mindig meg kell találni ugyanis a számunkra megfelelő egyensúlyt az adatok frissessége, konzisztenciája, az egyszerre fogadható szálak száma és az egyes szálak teljesítménye között.

További információk

Chris Brumme hosting blogja:
http://blogs.msdn.com/cbrumme
Nicholas Blachford érdekes cikkei a témában:
http://www.blachford.info/computer/articles/bigcrunch1.html

Összefoglalva: a webszerverek esetében általában kisebb és egyszerűbb az elvégzendő feladat, kevesebb a közös erőforrás, ami lehetővé teszi a szerverek kényelmes scale out és némileg limitált scale up skálázását is. Azonban nem tud egy adott lekérdezésen gyorsítani, hiába áll több processzormag rendelkezésre, csak a lekérdezések együttes kiszolgálásával ér el teljesítménynövekedést. Az SQL Server ezzel szemben sokkal komplexebb feladatok ellátására is képes, de minden erőforrás gyakorlatilag közös (hacsak nem tudjuk teljesen elszigetelten particionálni az adatokat) és képes akár egyetlen lekérdezést is párhuzamosítani, de csak ha van értelme.

Új utakon

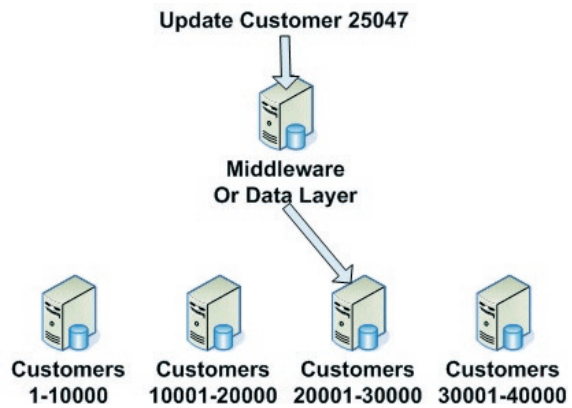
Mint láttuk, van már néhány (bár egymástól nagyban eltérő) lehetőségünk arra, hogy egy többfelhasználós, többszálas rendszert hatékonyan megépítsünk. Viszont felmerül egy másik kérdés: hogyan gyorsíthatók párhuzamosítással az alapvető programozási algoritmusok, amelyek nem SQL-ben vannak, és mondjuk nem is weboldalról beszélünk? Hogyan gyorsíthatók azok a kódok, amelyeket a ma elterjedt programozási nyelvekkel, mondjuk .Net keretrendszerre készítünk el? Hogyan tudjuk kihasználni a párhuzamosságot anélkül, hogy explicit módon törődnénk a szálkezeléssel, és mindent külön szálakra akarnánk szétdobálni? És talán a legfon-

tosabb: hogyan tudjuk mindezt nemcsak a szerverszoftverekkel, hanem akár a kliensalkalmazások esetében is megvalósítani?

A kliensalkalmazások rendkívül kényesek: az operációs rendszer szempontjából szinte minden egyes futó szoftver stateful „lekérdezésekből", vagyis ez esetben parancsokból áll, amelyek többnyire izolált szálakba és folyamatokba vannak szervezve. A kliensszoftver által kiadott parancsok szinte soha nem párhuzamosíthatók, hacsak a fejlesztő nem szervezi őket külön szálakba, vagy nem indítja el aszinkron módon, ami ilyenkor egy, a Thread Pool által létrehozott worker threaden kezd hozzá a feladat végrehajtásához.

Nyúljunk vissza az SQL-hez – elvégre az adatbázisszerverek készítői tudják talán a legjobban, hogyan is lehet növelni egy lekérdezés sebességét, és az SQL Server skálázhatósággal kapcsolatos igényei vannak talán a legközelebb a kliensalkalmazásához, legalábbis ha egy adott parancs vagy lekérdezés gyorsítása a cél.

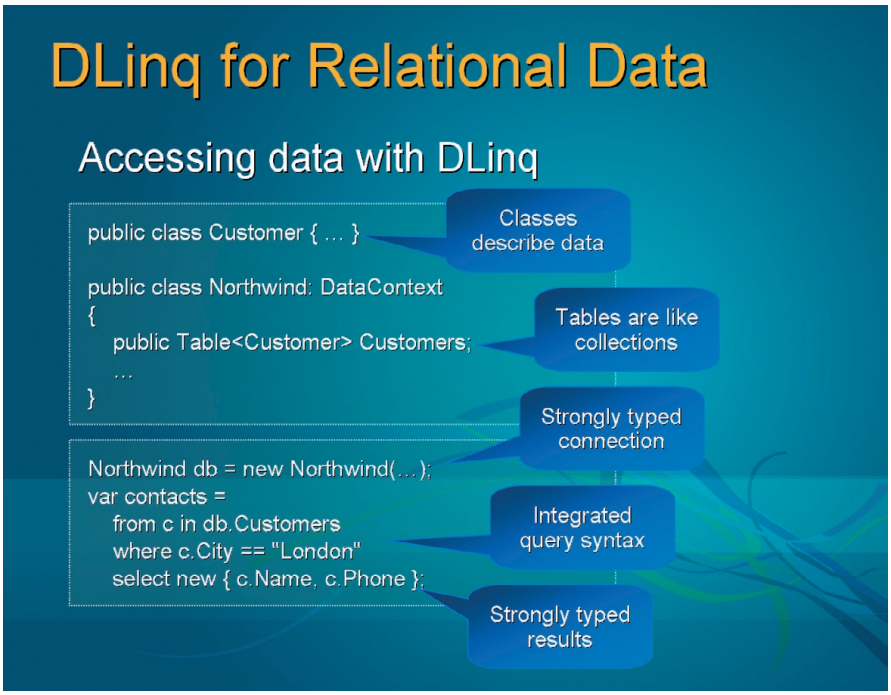
Annak több oka is van, hogy az SQL sok szempontból kiemelkedő teljesítményt tud elérni. Bár nem tértünk ki rá, de belső szál- és memóriakezelése optimálisra sikerült a lekérdezések szempontjából; viszont ami igazán fontos, hogy nekünk, programozóknak szinte csak azt kell megmondanunk, hogy mit szeretnénk a lekérdezés eredményeként



Adatmódosítás particionált adatbázis esetén

kapni, és nem azt, hogy hogyan.

Ez azzal jár, hogy a rendszer motorja képes helyettünk hatékony és akár párhuzamos kódot is készíteni, és nemcsak a kapott adatokat, hanem a lekérdezés optimalizált tervét is cache-eli nekünk, anélkül, hogy ebből bármit is látnánk, így legközelebb mindez még gyorsabban történik.



Adatelérés LINQ-kel

Lássunk néhány érdekes újdonságot, amelyek az előbb részletezett előnyöket próbálják meg kihasználni a kliensoldal, illetve a nem multi-user szoftverek gyorsítására!

PLINQ – párhuzamosítható adatkezelés SQL nélkül

A LINQ (Language Integrated Query) technológia a soron következő ADO.NET részeként jelentkezik, és a C# 3.0, valamint az új Visual Basic nyelv szerves részévé válik majd. A PLINQ (Parallel LINQ) ennek egy speciális alfaja, ami kifejezetten a párhuzamosság kihasználását tűzte ki céljául. Anélkül, hogy elmélyednénk a részletekben, lássuk, miért is érdekes ez.

A LINQ dióhéjban arra való, hogy különféle ciklusok kialakítása és parancsok egymás után rakosgatása helyett lehetőségünk legyen arra, hogy a hagyományos programozási nyelvekkel tetszőleges adatvektorokat, illetve listákat SQL-szerű szintaktikával dolgozzunk fel. Így a rendszer lehetőséget kap arra, hogy kitalálja helyettünk, hogyan lesz a leghatékonyabb az adott lekérdezés lefuttatása, akár több processzormag felhasználásával is (és persze az sem mellékes, hogy a megírandó kód is messze egyszerűbb, áttekinthetőbb, célorientáltabb lesz ezáltal). A lényeg: nem szabad szekvenciálisan gondolkodni. Azt mondjuk meg, mit szeretnénk, a többit bízzuk nálunk okosabbra!

A PLINQ hatékonyságát matematikailag is bizonyították, amit jól szemléltet a következő, bár kevéssé tudományos megállapítássorozat is: minden LINQ-lekérdezésnek van SQL-megfelelője. Minden SQL-lekérdezés relációs algebrára fordítható, és minden relációs algebrával leírt egyenlet párhuzamosítható. Ezt az egészet az is segít alátámasztani, hogy az adatbázisszerverek motorjának fejlődése az elmúlt 20 évben jelentős mértékben fókuszált az SQL-lekérdezések párhuzamos feldolgozására, nem kevés sikerrel – most csak ezt a már meglévő tudást hasznosítjuk újra, más környezetekben. Hamarosan megérkeznek az első publikus teszteredmények is a PLINQ-ről – az viszont már most elmondható, hogy amiket most házon belül látni lehet, nagyon meggyőzőek!

A LINQ arra is képes, hogy ugyanezekkel a nyelvi eszközökkel egy tényleges adatbázison futtassuk le a lekérdezéseinket. Ez azt jelenti, hogy nem kell többé stringekben írogatnunk SQL-parancsokat (csak ha szeretnénk), vagyis végre minden teljesen típusosan kerül feldolgozásra, így jelentősen

csökken a szükséges kód mennyisége, és nem kell az adatkezelési réteggel annyit foglalkoznunk, mint korábban.

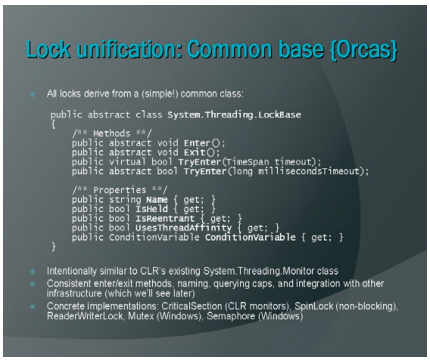
Tranzakciókezelés a memóriában?

A szoftverfejlesztés egy alapvető dolgot gyakran figyelmen kívül hagy (a legnagyobb üzleti rendszerek kivételével): az adatok a kommunikáció során mindig korábbi állapotot tükröznek, vagyis potenciálisan már elavultak (hacsak nem lockoljuk az adott memóriadarabot az adott gépen, de akkor meg nem beszélhetünk nagyon skálázhatóságról). Az esetek többségében azonban nem is célunk az éppen abban a nanoszekundumban aktuális adattal dolgozni, csak szeretnénk kapni egy olyan állapotot, ami konzisztens, és lehetővé teszi számunkra a hibátlan műveletvégzést, reagálást.

Gyakorlatilag ez a lényege az ACID-elven működő tranzakció-kezelésnek az adatbázisszerverekben. Talán kevesen tudják, de a „Longhorn" Serverben már elérhető lesz a tranzakcionális Registry és NTFS, amelyek kezeléséért maga a kernel felel. A Visual Studio „Orcas" és a .Net keretrendszer következő verziójának fejlesztői azonban már a memória tranzakcionális kezelésén is keményen dolgoznak. Az előzetes tervek szerint ez a tranzakcionális memória nem lesz ACID, csak ACI (D, vagyis Durability nélküli lesz, azaz nem lesz célja az adatok garantált és hibamentes tárolása, de ezt maga a memória sem garantálta soha).

Lényegében „atomic", vagyis megbonthatatlan blokkokat tudunk majd definiálni a kódunkban, ezek garantáltan egy konzisztens memóriaállapotban fognak lefutni, amit vagy lockolással, vagy a használt memóriaterületekről készített pillanatkép (snapshot) használatával fognak megvalósítani.

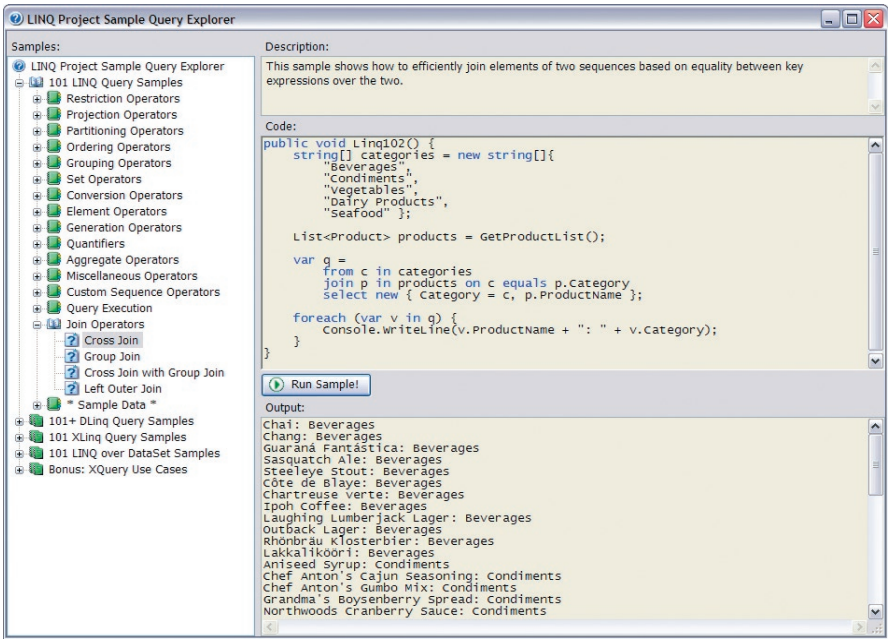
Ez az elképzelés lehetővé teszi majd számunkra, hogy az alkalmazásoknak és a szálaknak kevesebbszer kelljen egymásra várniuk, vagy ha ez mégis kikerülhetetlen, akkor az ehhez szükséges lockolást ne a fejlesztőnek kelljen leprogramoznia, hanem kezelje le a keretrendszer automatikusan. Összességében várhatóan csökkenni fog a szálak közti adatfüggőségek száma és jelentősége, vagyis könnyebben lehet az egyes szálakat külön processzormagokon futtatni.



Az egységesített lock osztályok absztrakt váza

A meglévő megoldások egyszerűsítése

A PLINQ-en és a tranzakcionális memórián kívül kisebb, de hasonlóan hasznos újdonságok is érkeznek a .Net keretrendszer új (harmadikat követő) változatában. Ide sorolható a hagyományos lockolás programozásának egyszerűsítése, amit elsősorban a lock-osztályok egységesítésével és absztrakciójával értek el. A másik terület, ami még kérdéses, hogy mennyire integrálódik az új rendszerbe az aszinkron hívások kényelmesebb kezelése, amire szintén több jó megoldást tesztelnek a Microsoft fejlesztői és kutatói, hogy még egyszerűbbé tegyék számunkra a párhuzamosság problémáinak kezelését.



Egy érdekes LINQ-példa a sok közül

Szál- és processzormag-affinitás

Általánosan érvényes tény, hogy egy adott, többszálú kód akkor tud a leggyorsabban futni, ha minden egyes hardveres szálon (hardveres szál: processzormag vagy HyperThreading esetén logikai processzor) egy időben egyetlen szál fut, egészen addig, míg a szál a feldolgozás végére nem ér. Ha több szál fut egy processzormagon, akkor azok valójában nem is futnak egyszerre, hanem csak felváltva, és a szálváltásokkor a szálak által használt memóriának, illetve a stacknek be kell kerülnie a processzorhoz közeli memóriatárakba, például a lehető legközelebbi cache-szintre. Ha gyakran váltogatjuk, hogy melyik szoftveres szál fut az adott hardveres szálon (ilyen például a hagyományos preemptív multitasking is egy processzoron), akkor a hardveres szál cache-ét és a hozzá tartozó memóriát minden egyes alkalommal teljesen feleslegesen írjuk felül. Ez a teljesítmény csökkenését idézi elő, mivel gyakorlatilag a cache csak az adott szál pillanatnyi futásáig bír tényleges értékkel, és a következő szál már ismét nem talál magának semmi hasznosat a cache-ekben.

Természetesen gyakran nem élhetünk az ideális megoldással, elvégre az esetek többségében nemcsak az számít, hogy egy adott szál mennyire gyorsan végez teendőivel, hanem az is lényeges, hogy hány beérkező kérést tudunk belátható időn belül kiszolgálni. Éppen ezért egy magra egynél több szálat is szokás ráengedni – ilyenkor az egyidejűleg kiszolgálható szálak száma és az egyes szálak lefutási ideje között kell megtalálni az egyensúlyt.

Budai Péter
(i-pbudai@microsoft.com) Microsoft Magyarország

A PARANCSNOKI FÜLKÉBEN ÜLVE

Mi az informatikus feladata egy meglévő rendszer üzemeltetésekor? Ismernie kell azt minden részletében. Az adott rendszer minden felmerülő és meglévő hibájáról tudnia kell. A hibákat a lehető leghamarabb ki kell javítania – System Center Operations Manager 2007.

Gyakorlatilag a cél már jól ismert: garantálni kell az informatikai rendszer rendelkezésre állását az azt használók számára. Sokan persze úgy gondolják, hogy egy rendszergazda elsődleges feladata új rendszerek kiépítése, továbbfejlesztése, ami egyes esetekben igaz is lehet, de ezt munkahelye válogatja. Az igazság azonban gyakran az, hogy a vállalatok számára az informatika csupán egy eszköz a sok közül, ami az üzletmenet hatékonyságának növelését szolgálja – épp ezért nem meglepő, hogy általában újabb és újabb informatikai beruházások helyett inkább a meglévő infrastruktúra karbantartását tekintik feladatnak.

Egy kisebb vállalatnál gyakran egyetlen rendszergazda is elegendő ehhez, aki lehet, hogy egyszerre több kisebb cég rendszerét is karbantartja, többnyire manuális munkával. Nem ritka, hogy a rendszergazda a helyszínen próbálja meg nagy küzdelmek árán megoldani a felmerült problémákat, miután azt a cég alkalmazottai jelzik neki. Ha azonban egy nagyobb vállalatról van szó több szerverrel és több száz számítógéppel, esetleg több távoli telephellyel, ott felmerül az igény a karbantartási folyamatok egyszerűsítésére, gyorsítására. Ugyancsak komolyabb technikai segítségre van szüksége annak az informatikusnak vagy szolgáltató cégnek, aki vagy amely egyszerre lényegesen több vállalatnak szeretne informatikai támogatást biztosítani, ráadásul preferáltan távolról, az interneten keresztül.

De igazából bárkinek, aki komolyan veszi egy informatikai rendszer üzemeltetését, el kell gondolkodnia azon, hogyan tehetné elégedettebbé a rendszer felhasználóit azáltal, hogy nagyobb rendelkezésre állást és gyorsabb problémamegoldást garantál a számukra.

A legkézenfekvőbb, ha a karbantartási teendőket és a hozzájuk kapcsolódó folyamatokat tekintjük át, és megnézzük, hogyan segítenek ebben már meglévő módszerek és szoftverek. Természetesen számtalan szoftver áll ehhez rendelkezésre, mi azonban most kiemelten a Microsoft System Center Operations Manager 2007-tel (SCOM), illetve elődjével, a Microsoft Operations Manager 2005-tel fogunk foglalkozni, illetve érintőlegesen a System Center-család többi tagjával.

Miből áll a rendszerem?

Amikor új kolléga érkezik a fedélzetre, természetesen az első feladat, hogy átlássa a rendszert, aminek az üzemeltetéséért felelős lesz. Ha rendelkezésre áll a rendszer pontos felépítésének leírása vagy modellje, az rengeteget segíthet nemcsak a karbantartás, hanem a későbbi rendszerfejlesztések, változtatások során is. Egy ilyen rendszermodellt folyamatosan karbantartani manuális eszközökkel meglehetősen nehéz feladat, különösen egy nagyobb vállalat esetében.

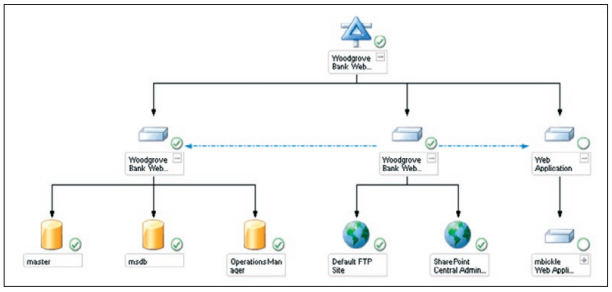
Az Operations Manager képes arra, hogy feltérképezze az adott hálózat számítógépeit, és az azokon található szolgáltatásokat, de ezeket akár kézzel is módosíthatjuk utána, igényeink szerint. Mindez egy olyan élő modellbe kerül bele, aminek segítségével folyamatosan nyomon tudjuk követni ezeknek a komponenseknek a változásait és az állapotát is. A 2007-es Operations Manager a feltérképezést az Active Directory segítségével is el tudja végezni, így akár egyszerűen egy egész OU-t is megadhatunk, amihez ha később új számítógépet rendelünk, az máris bekerül a felügyelt gépek közé, és automatikusan feltelepül rá minden szükséges felügyeleti összetevő.

A SCOM a hozzá kapcsolódó Management Packek segítségével képes adott gyártók szolgáltatásait felismerni, azok jellemzői alapján (például a registry, a nyitott portok, a futó servicek vizsgálatával). Amelyik szoftverhez elérhető ilyen Management Pack, arról az Operations Manager gyakorlatilag mindent tudni fog, beleértve azt is, hogy milyen alkomponensekre bontható tovább, és milyen

más rendszerkomponensekre van szüksége ahhoz, hogy az adott szolgáltatás megfelelően működjön. A Microsoft valamennyi szerverszoftveréhez készít Management Packet (és ezek ingyenesen elérhetőek), így azok automatikus felderítése a hálózaton gyakorlatilag garantált.

Saját alkalmazásaink és szervermegoldásaink felismerése azonban általában nem automatizálható, azokat kézzel kell felvenni a SCOM-ba, vagy gyártanunk kell hozzá egy Management Packet magunknak. Ebben az elosztott rendszerek komplex modelljeinek tervezésére alkalmas felület is segítségünkre siet, ahol pontosan definiálhatjuk, hogy melyik komponensünk mely más komponensektől és milyen módon függ. A SCOM arra is képes, hogy hardverkomponenseket fedezzen fel és felügyeljen, beleértve a hálózati eszközöket, például routereket is, ha rendelkezünk ehhez szükséges Management Packkel.

Az *1. ábrán* látható modell valójában olyan XML állományok halmazaként képzelhető el, amelyek mind-mind egy adott komponenst és annak viszonyát írják le a külvilággal, illetve azt, hogy milyen állapotai lehetnek, és milyen követelményei vannak másokkal szemben. Ezt az XML-állományt az SDM, vagyis a System Definition Modell séma írja le, ami nem csupán a rendszermenedzsment kapcsán lesz érdekes a jövőben. Itt érdemes megemlíteni, hogy már most, a Visual Studio 2005 elosztott rendszerek modellezésére használható felülete is SDM-eket ment el, ami sok mindent előrejelez.

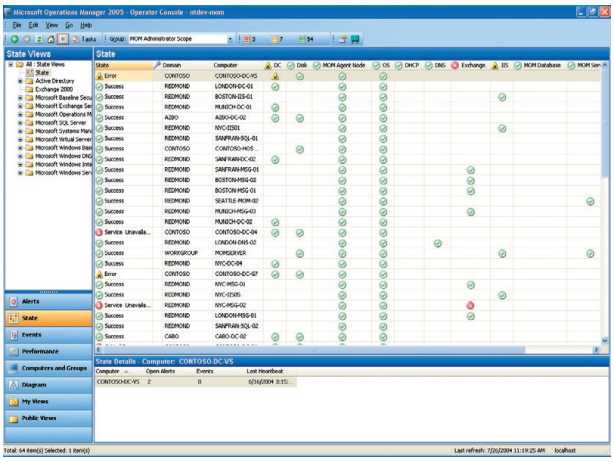


1. ábra. Egy szolgáltatás felépítése és dependenciái

Minden jól működik?

A Management Packeket szinte mindig olyanok készítik, akik a lehető legjobban ismerik azt a komponenst, amit felügyelni szeretnénk: ez általában a komponens saját fejlesztőit jelenti. Elvégre ki tudna többet például az Exchange Server 2007 felügyeletéről, mint az Exchange saját fejlesztőcsapata?

Ahhoz azonban, hogy tudjuk, hogy rendszerünk valóban jól működik, azt megfigyelő „ügynökökre" van szükségünk. Ezek persze lehetnének emberek is, akik percenként rá-



2. ábra. A felügyelt gépek és a rajtuk futó szolgáltatások állapota MOM 2005-tel

néznek az eseménynaplóra, vagy egy teljesítmény-mérőszámra, azonban az Operations Manager ezt képes automatizált szoftverkomponensekkel is kiváltani. Ezek az úgynevezett Agentek. Az Agenteket az informatikai rendszer feltérképezésekor automatikusan telepíteni lehet a megfigyelt számítógépekre, ilyenkor az Agent folyamatosan tájékoztatja a SCOM szervert a felügyelt komponensek állapotáról, teljesítményéről, eseményeiről.

Ezek az adatok bekerülnek a SCOM szerveroldali SQL adatbázisába *(2. ábra)*, és ebből bármely pillanatban nagy biztonsággal meg lehet állapítani, hogy mely pontokon van esetleg probléma a rendszerünkkel. Ebben az állapot, vagy State-nézet van a segítségünkre a SCOM konzoljában. Az, hogy egy adott komponens milyen állapotú, nagyon sok mindentől függhet: például az aktuális terhelésétől, illetve olyan más, hozzá tartozó komponensek állapotától, amelyek befolyással vannak rá. Hogy mikor látszik számunkra a SCOM konzoljában egy komponens hibás állapotúnak, az a lehető legapróbb részletekig beállítható, de ebben természetesen ismét segít a Management Pack előre meghatározott értékekkel.

Az Operations Managerrel lehetőség van a távoli gépek Agentless, vagyis Agentek nélküli felügyeletére is; ilyenkor a szerver adott időközönként „pollozza" – kérdezgeti – az adott gépet állapotáról, azonban ez nagyobb terhet ró a szerverre, és megvalósítani is nehezebb, mint az Agent alapú felügyeletet.

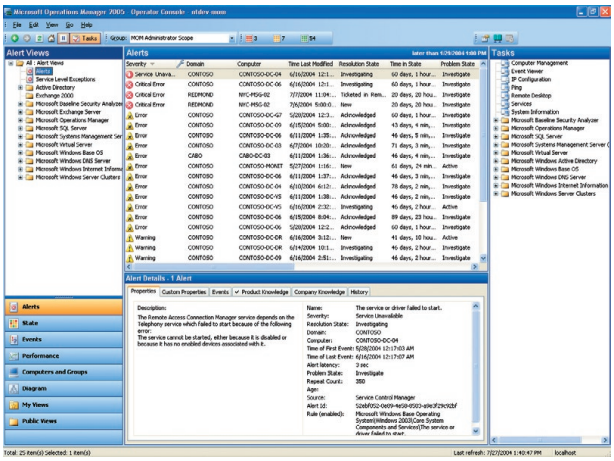
A SCOM arra is alkalmas, hogy egy szolgáltatás elérhetőségét úgynevezett szintetikus tranzakciókat kezdeményezve állapítsa meg – elképzelhető ugyanis olyan eset, hogy egy szolgáltatás fut, és a menedzsmentszerver el is éri azt, de a kliensek nem. Az ilyen esetekről úgy értesülhetünk, ha „figyelőket" helyezünk el egyes kliensgépeken, amelyek bizonyos időközönként megpróbálják működésre bírni az adott szolgáltatást, ennek eredményét pedig továbbítják a SCOM szerver felé.

Mi történt? Mit kell reagálni erre?

A hiba az esetek többségében nem elszigetelt esemény, hanem fennálló állapot, ami addig nem is nagyon változik, amíg el nem hárítjuk. Azonban már a hibák legelső jelei is események formájában érnek el hozzánk. A SCOM esetében a felügyelt szolgáltatások a rajtuk lévő komponensekkel kapcsolatos eseménynapló-bejegyzéseket és teljesítményadatokat is továbbítják a menedzsmentszervernek, így már a legelső figyelmeztetésekről is értesülhetünk, még ha azok nem is idézik elő a komponensek állapotának hibásra változását. Az SCOM a legtöbb adathoz WMI-lekérdezések futtatásával jut hozzá, ami gyakorlatilag teljesen tűzfalbarát módon működik, a WS-Managementen keresztül is.

Amikor egy komponens állapotát vizsgáljuk, rögtön láthatjuk azokat az eseményeket, amelyek vele történtek. Ha például az Exchange szerverünk nem működik megfelelően, akkor azokat a kiváltó eseményeket, amelyek ennek a ténynek a megállapításához voltak szükségesek (nem indult el a szolgáltatás vagy valamiért leállt) máris megtekinthet-

jük. Ilyenkor a Management Packben tárolt rengeteg információ (gyakorlatilag egy tudásbázis) segít abban, hogy mik lehetnek ennek a hibának a legvalószínűbb okai, és megoldást is kínál rá, amennyiben van ötlete.



3. ábra. A beérkezett hibák, figyelmeztetések és ezek részletei MOM 2005-tel

A SCOM 2007-es változata a MOM 2005-tel ellentétben ilyenkor nemcsak scripteket, szöveges ajánlásokat vagy KB-cikkeket ajánl fel *(3. ábra)*, hanem arra is lehetőséget ad, hogy egy adott megoldást (például a szolgáltatás újraindítását vagy egy karbantartó script indítását) egyetlen gombnyomással elindíthassunk. Természetesen ezt a hibákat és eseményeket bemutató tudásbázist mi is bővíthetjük, így a mi környezetünkben gyakran előforduló hibákra akár egy gombnyomással megoldást szolgáltathatunk még akkor is, ha éppen szabadságon vagyunk.

Ha időben látunk minden eseményt, akkor még a kezdeteknél észrevehetünk olyan hibákat, amelyek később más komponensekre is kihathatnak, hiszen általában az első hiba a valódi hibaok. Azonban vannak olyan komplex esetek, amikor több, egymástól teljesen elszigetelt hiba fordul elő, vagy még egy korábbi hiba fennáll, miközben egy újabb érkezik. Mit tehetünk, ha a tényleges hibát valójában más alkomponensek hibái okozzák, amiket szintén alaposabban meg kellene vizsgálnunk a probléma elhárításához, ahelyett, hogy a valódi hiba mellékhatásait gyógyítjuk?

Mi a hiba valódi oka?

A hiba valódi okának megkeresése általában meglehetősen fáradságos tevékenység. A rendszer komponensei ugyanis gyakorlatilag egy gráfot alkotnak, és szinte minden mindenre hatással lehet. Ha például egyszerre három szolgáltatásunk leáll, tudjuk-e, hogy mi a teendő? Egyenként kell-e velük foglalkozni, vagy lehet, hogy mindhárom hiba egy dologra vezethető vissza? Honnan tudjuk, hogy ha nem működik egy telephelyen a levelezés, akkor milyen mélyre kell leásnunk ahhoz, hogy a valódi hibát megtaláljuk? Lehet, hogy az Active Directory lesz a hiba valódi oka, és nem is az Exchange szerver vagy a hálózat?

Ha rendelkezésünkre áll rendszerünk élő modellje, ezekre a kérdésekre szinte azonnal választ kaphatunk. Szerencsére a SCOM pontosan tisztában van az egyes komponensek állapotával. Ha vizuálisan képzeljük el a komponensek viszonyát egymáshoz képest, szinte biztos, hogy ott lesz a probléma, ami a legmélyebben található a rendszerünkben, és bizonyítottan hibás. A többi hiba várhatóan csak ennek következménye lesz. A SCOM segítségével egy hiba okának feltárása sokkal egyszerűbb, mint más módszerekkel, hiszen azonnal látható, melyik az a pont, amelyet elsőként vizsgálnunk kell, ha beérkezik az első panasz, hogy nem működik a levelezés. Sőt, az is lehet, hogy mi vesszük észre a hibát elsőként, és még az előtt el tudjuk hárítani, hogy bárki munkáját komolyabban akadályozta volna.

Hogyan mutassam meg az IT rendelkezésre állását?

A SCOM segítségével jelentéseket is készíthetünk (immár SQL Server 2005 Reporting Services alapokon) az informatikai rendszerünk rendelkezésére állásáról, az egyes teljesítménymutatókról vagy például a szolgáltatások meghibásodásának időpontjairól, arányáról. Azt is pontosan megmutathatjuk, hogy melyek azok a kliensalkalmazások, amelyek gyakran lefagynak, és vélhetően a legtöbb felhasználói panaszhoz vezetnek.

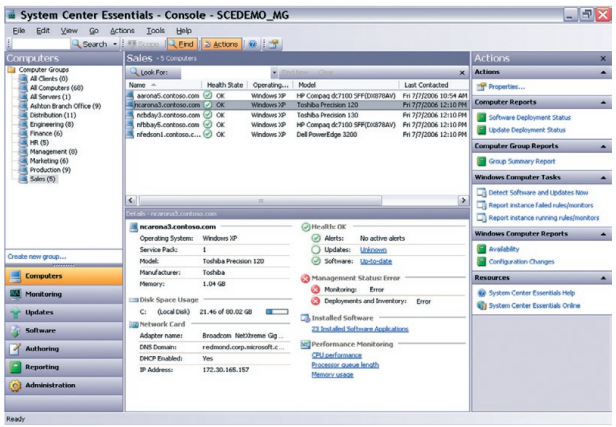
Hogyan tudok több céget/telephelyet távolról felügyelni?

Maga a menedzsmentszerver MMC-s felülettel rendelkezik, ami természetesen akár távolról is elérhető, de mindezek mellett hozzáférhetünk egy webes felügyeleti oldalhoz is.

A SCOM szervereket könnyen össze lehet kötni egymással (vagy más gyártók hasonló megoldásaival), akár HTTPS-en keresztül is a WS-Management protokoll segítségével. Érdemes felügyelt telephelyenként egy-egy SCOM szervert elhelyezni, hogy az lokálisan, hatékonyan tudja gyűjteni az adatokat a telephely gépeiről.

Ezeket a távoli menedzsmentszervereket egy közös szerverről tudjuk a legkényelmesebben figyelni, aggregálva az egyes SCOM szerverek által gyűjtött adatokat.

Ez a felállás *(4. ábra)* nemcsak a nagyobb vállalatoknál lehet jó megoldás arra, hogy teljes hálózatukat központosítva tudják felügyelni, hanem akár arra is alkalmas lehet, hogy egy kisebb cég legyen képes felügyelni más cégeket távolról. A megoldás lényege: mindegyiknél telepítünk egy SCOM-ot (vagy a részben kisebb tudású, de a kisvállalatok számára sokszor hasznosabb System Center Essentialst), valamint a szolgáltató maga rendelkezik egy központi SCOM-mal, ahonnan folyamatosan szemmel tudja tartani, hogy mely felügyelt cégnél milyen az informatikai rendszer aktuális állapota.



4. ábra. A System Center Essentials konzolja

Budai Péter
(i-pbudai@microsoft.com) Microsoft Magyarország

A Windows Vista bevezetése

Az új rendszert támogató új és megújult technológiák.

Ismerjük meg, hogyan válhat gyorsabbá és egyszerűbbé az asztali számítógépek bevezetése/telepítése az új technológiák segítségével! Bár sok hasznos újdonság érkezik a Vistában, azonban a bevezetésért felelős informatikusok csak egy dolgot kérdeznek: az eddigieknél könnyebb lesz-e ezt bevezetni.

A termék bevezetése alapvetően három pilléren nyugszik.

- **A platform újításai.** A Windows Vista új image-elő (lemezképkezelő) technológiát, alkalmazásokat és eszközöket tartalmaz az image-ek készítésére, módosítására, testreszabására (az egyéni igényeknek megfelelően) és alkalmazására.
- **Az alkalmazások kompatibilitása.** A Vista a Windows XP-n futó alkalmazások legnagyobb részét probléma nélkül futtatja, valamint a régebbi alkalmazásokat tekintve, a Windows XP-vel összehasonlítva megnövelt kompatibilitással rendelkezik a régi verziókat emuláló réteg továbbfejlesztésével.
- **A bevezetés módja.** A Windows XP és régebbi verzióinak bevezetésekor külső gyártótól származó image-technológiákat kellett használnunk a rendszer bevezetéséhez. Az image-technológiák beépülése révén most már egyszerűbben és külső szoftverek nélkül is megvalósíthatjuk ezt a folyamatot.

A bevezetési folyamatot segítő új technológiák

- **Modularizáció.** A Vista teljesen különálló, csomagszerű részekből épül fel, ezek kihasználásával a legapróbb részletekig testreszabhatjuk az összetevők működését, és igényeiknek megfelelően konfigurálhatjuk őket. A modularizáció ugyancsak segítséget nyújt a külső csomagok (driverek, alkalmazások, nyelvi felületek) és frissítések rendszerbe illesztésében, azok különálló telepítése nélkül. Ez nyújt lehetőséget arra is, hogy a Microsoft az operációs rendszer részeit egyenként frissítse és javítsa, megelőzve az egyéb összetevők nem megfelelő működését. A felhasználói felület megjelenési nyelvét is modulként befolyásolhatjuk a rendszertől függetlenül (vagyis MUI csomagok használata nélkül), egyszerűen a telepítőkészlet testre szabásával.
- **Windows Imaging Format (WIM).** A WIM egy új, fájl alapú image-formátum, amelynek használatával lehetőségünk van egyetlen image-et több, eltérő hardverkonfigurációval rendelkező számítógépre telepíteni, mindezt gépenként egyedi beállítások használatával. A WIM image-fájlok kezelése egyszerű, anélkül telepíthetünk/törölhetünk Windows-összetevőket, drivereket, alkalmazásokat és frissítéseket, hogy az image-et bebootolnánk.
- **Egységes XML-válaszfájl.** Az eddigi, külön-külön előállítandó válaszfájlok helyett most már elegendő egyetlen XML fájl is, ami ráadásul többféle konfiguráció kezelésére is képes, hála a hardverfüggetlenségnek és a modularitásnak.

A bevezetés eszközei (a Windows Automated Installation Toolkit – a WAIK – részei):

- **Application Compatibility Toolkit.** A vállalatnál használt alkalmazások és a Vista kompatibilitásának ellenőrzésére és javítására.
- **User State Migration Tool (USMT).** A felhasználók Windows 2000 vagy Windows XP alatt tárolt beállításainak és személyes fájljainak automatikus migrálása a Windows Vista platformra.
- **ImageX.** WIM image-fájlok létrehozása és fájlszintű szerkesztése.
- **Windows System Image Manager (WSIM).** A Windows Vista WIM image-fájlok komponens alapú módosítása. A rendszerkomponensek beállításainak módosítása és külső összetevők hozzáadása (például nyelvek és driverek) is ezzel lehetséges, mindezt egyetlen unattend.xml létrehozásával tehetjük meg.
- **Windows Preinstallation Environment (WinPE).** Rendszerbeállítások módosítása (például formázás és particionálás) az operációs rendszer elindítása nélkül.

Habár a modularizáció és a WIM-formátum már külön-külön is sokat könnyít a telepítések létrehozásán, a két technológiát együtt alkalmazva még tovább egyszerűsíthetjük a telepítés folyamatát.

Application Compatibility Toolkit (ACT). Az asztali operációs rendszerek cseréje a rajtuk futó alkalmazások miatt alapos tervezést és körültekintést igényel. Különös gondot kell fordítani a már telepített programok Vistával való kompatibilitásának ellenőrzésére. Az ACT az alábbiakban segít nekünk:

- **API-kompatibilitás biztosítása.** A külső szoftvergyártók számára elérhetővé tett információk alapján alkalmazásainkat könnyen alakíthatják Vista-kompatibilissá.
- **Software Inventory Analyzer.** A vállalatnál telepített PC-ken lévő szoftverekről leltárt készít, központi helyen tárolja, ösz-

szesíti és összeveti az ACT webes adatbázisával, megjeleníti az esetleges problémás számítógépeket és alkalmazásokat.

- **Filtering Analyzer.** Egységes lehetőséget nyújt arra, hogy a felhasználók hibát jelezhessenek a rendszergazdának, amennyiben egy alkalmazás Vista alatt nem működik megfelelően. Ugyanekkor automatikus frissítéskeresés is végrehajtódik az adott szoftverre.

WIM. Lássuk az új image-formátumot! A manapság elérhető image-technológiák (például a Ghost) által használt formátumok általában szektor alapúak, ezzel szemben a fájl alapú WIM sokkal több előnyt nyújt számunkra. A WIM-formátum hardvertől független, így tetszőleges számú és fajtájú hardverkonfiguráción is működőképes ugyanaz az image.

	Tervezés	Előkészület	Bevezetés
Teendők	Alkalmazás-kompatibilitás ellenőrzése, migrációs scriptek	Az image összeállítása és testre szabása	Telepítés
Vistás eszközök	ACT USMT	ImageX Windows System Image Manager Sysprep	USMT WDS Image alapú telepítés

Mindezek mellett egy .WIM fájl több image-et is tud tárolni; erre a legjobb példa, hogy a Windows Vista végleges dobozos verziójának DVD-i ugyanazt az image-et fogják tartalmazni, és termékkulcstól függően a fájlból más és más verziójú operációs rendszer csomagolódik ki.

Ugyancsak lehetőségünk lesz például az alkalmazásokat külön image-ben tárolni a WIM fájlon belül, tehát akár gépenként eldönteni, hogy melyekre mely alkalmazáskészlet kerül. A WIM támogatja a single instancinget, ezzel jelentősen csökkentve a fájl méretét. A single instancing használatával a két komponensben megtalálható azonos fájlokat csak egy példányban tároljuk, és a másik fájlnál csak egy hivatkozást hozunk létre (például több Windows-image esetén egy WIM fájlban: az operációs rendszer fájljai szinte teljesen megegyeznek, így akár 30–40 százalékkal is csökkenhet a képfájl mérete).

Megoldható lesz az image offline szervize is. Az operációs rendszer összetevőitől kezdve a patcheken és a drivereken át az alkalmazásokig bármit törölhetünk vagy hozzáadhatunk az image-hez egy új image létrehozása nélkül, értékes órákat spórolva meg. Eddig egy szoftverjavítás image-be illesztése csak az image egy gépre való feltelepítésével, a patch alkalmazásával és az image újragenerálásával történhetett meg; a Vistával azonban nemes egyszerűséggel offline is megtehetjük ugyanezt.

Nagyon fontos a rugalmasság szempontjából is, hogy a WIM nem szektor, hanem fájl alapú, hiszen így tetszőleges méretű partícióra telepíthetjük, nem kell pontos partícióméreteket követni. Egy WIM-image telepítésekor nem kötelező a teljes partíciót formázni, annak tartalma megmaradhat, csak az operációs rendszer cserélődik alatta, gyakorlatilag adatvesztés nélkül.

Fejlesztők, figyelem! Egy új API, a WIMGAPI segítségével tetszőlegesen turkálhatunk az image-fájlokban, erről további információkat hamarosan az MSDN-en találhatnak az érdeklődők a http://msdn.microsoft.com/windowsvista címen.

ImageX

Az ImageX egy igazán lényegre törő program, ezért szeretik már most is sokan. Egy egyszerű, parancssoros eszköz, hasonlóan például az Xcopyhoz. Alaphelyzetben az eszköz arra képes, hogy egy meglévő kötetről készítsen nekünk egy .WIM állományt, vagy hogy egy kötetre egy már korábban létrehozott .WIM-et csomagoljunk ki teljesen vagy igény szerint részlegesen. Előbbit nemes egyszerűséggel az

imagex /capture C: image.wim "Name"

parancs használatával tudjuk elérni, utóbbihoz pedig az

imagex /apply image.wim 1

utasítást kell kiadni. Ugye, milyen egyszerű?

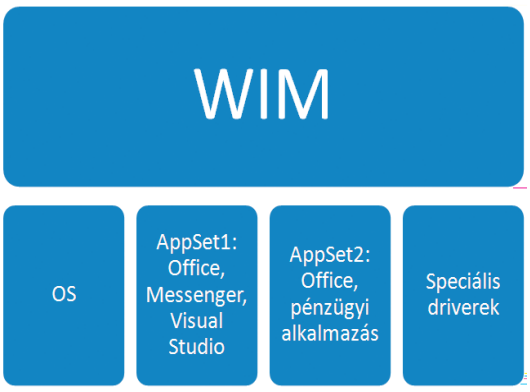
Az ImageX ezenkívül lehetőséget nyújt az image-ek felcsatolására, ami után úgy bütykölhetünk a .WIM fájlon, mintha valóban telepítettük volna az image-et (gyakorlatilag egy betűjelet rendelünk hozzá, ami egy virtuális merevlemezként érhető el ezután).

Az ImageX főbb parancsai:

/append: egy image hozzáadása a WIM-fájlhoz
/apply: egy WIM-ben található image kibontása a megadott meghajtóra
/capture: image létrehozása egy új WIM-fájlba
/commit: a felcsatolt image módosításainak érvényesítése a WIM-ben
/compress: a tömörítés módjának állítása
/config: imagex-beállítások módosítása
/delete: image törlése a .WIM fájlból
/dir: .WIM-ben lévő image tartalmának listázása
/export: image átvitele két WIM között
/info: image-adatok kivitele XML-be
/ref: mélyreferenciák módosítása (csak profiknak)
/split: image felosztása több, kisebb részre
/verify: adatellenőrzés, átviteli hibák kiszűrése
/mount: image csatolása WIM-ből
/mountrw: image csatolása WIM-ből – írási joggal
/unmount: csatolás megszüntetése
/?: súgó

Windows System Image Manager

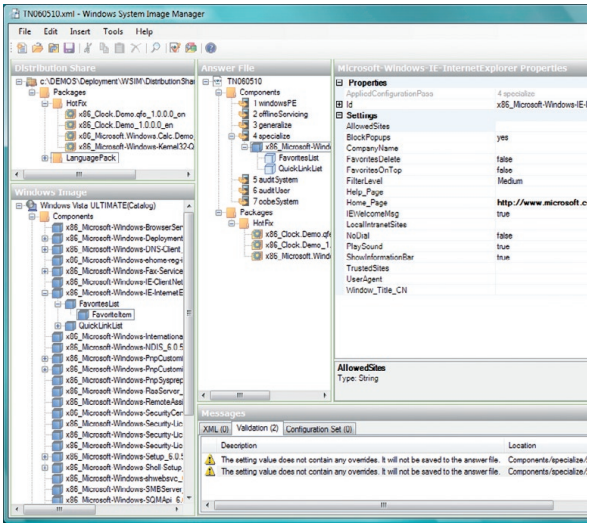
A Windows System Image Manager a Vista telepítésének testreszabásához használt XML-válaszfájlok szerkesztésére használható eszköz. Segítségével a rendszer összetevőit modulonként tudjuk konfigurálni, például megváltoztathatjuk vele az Internet Explorer kezdőlapját, vagy mondjuk letilthatjuk a képernyővédőt.



Egy .WIM szerkezete

A WSIM használatához mindenekelőtt telepítenünk kell azt az internetről letöltve, de a resource kit részeként is elérhetjük. A tele-

pített WSIM-et futtatva létre kell hoznunk egy katalógust. A katalógus voltaképp egy indexfájl, aminek létrehozásakor a program elemzi a WIM-fájlban elérhető beállítási lehetőségeket és tárolja azokat. Ezek után egy új, üres (vagy létező) válaszfájlt kell megnyitnunk, amibe elmentjük a választott módosításokat. Ha ezzel is megvagyunk, meg kell adnunk egy disztribúciós megosztást, ahova a telepítéshez szükséges fájlokat helyezi el a WSIM. Ez a mappa akkor különösen fontos, ha külső programokat is szeretnénk az operációs rendszerrel együtt, hálózatról telepíteni. A Windows telepítése jól definiált fázisokra van bontva, és a különböző beállításokat a telepítés más és más szakaszaiban tudjuk csak módosítani. (Vagyis például az Internet Explorer kezdőlap-beállításait természetesen csak akkor módosíthatjuk, amikor már a szükséges registry hive és -fájlok már a célszámítógépen vannak; előbb semmiképp.)



A Windows System Image Manager felülete

E fázisok közül az első a WindowsPE, amely opcionális, és csak akkor fut le, ha a telepítést WinPE környezetből indítjuk.

Az offlineServicing a telepítő első (vagy második, WinPE esetén) fázisa, mely a WIM-fájl másolásával egy időben hajtódik végre. A további fázisok (generalize, specialize, auditSystem, auditUser) már a WIM kicsomagolása után hívódnak meg.

Az oobeSystem fázis a felhasználói felület (OOBE – Out-Of-The-Box-Experience) első megjelenésekor hajtódik végre.

Ezek testre szabásához nincs más dolgunk, mint a katalógusból kiválasztani a befolyásolandó összetevőt (a WSIM automatikusan a megfelelő fázisba helyezi el a módosításokat), és a jobb oldali panelen kedvünk szerint módosítani az elérhető beállításokat.

A WSIM által kreált válaszfájl használatához vagy a Windows Deployment Services lehetőségeit kell használnunk, vagy nemes egyszerűséggel az XML-válaszfájlt Autoattend.xml-nek elnevezve kell egy pendrive-ra vagy a telepítő médiára másolni. Ezek után a telepítés indítása előtt a pendrive-ot a számítógépre kötve kell hagyni, hogy a telepítő automatikusan megtalálja, és alkalmazza a rajta található beállításokat. A Windows Deployment Services gyakorlatilag az ADS és a RIS utóda, ami elérhető lesz a Windows Server 2003 SP2-ben, és a „Longhorn" Serverben is.

Amennyiben úgy kényelmesebb számunkra – tekintettel a WIM hardverfüggetlenségére –, megtehetjük, hogy készítünk egy referenciatelepítést a válaszfájl használatával, majd az ImageX segítségével rögzítjük a telepített rendszert, és az alkalmazások telepítése után ezt az image-et forgalmazzuk tovább a megfelelő csatornákon keresztül (például WDS segítségével).

Fontos, hogy noha a WIM önmagában valóban nem hardverfüggő, nem szabad elfelejteni: az image létrehozásakor a manuális telepítések során sok egyedi, a számítógépre és a felhasználóra jellemző adat is létrejön. Ezeket a sysprep-pel mindenképpen ki kell pucolni, mielőtt több különböző konfiguráció telepítéséhez kezdenénk hozzá!

A nagytakarításhoz a C:\Windows\System32\Sysprep.exe /oobe /generalize /shutdown parancsot kell futtatni. Ekkor a sysprep az összes egyedi információt törli a számítógépről, újra engedélyezi az OOBE lapokat (amelyek a telepítés után megjelennek – hogy a felhasználó meg tudja adni telepítéskor például személyes adatait: felhasználónevét vagy egyes esetekben magát a termékkulcsot is). Mindezek végeztével pedig leállítja a számítógépet.

Röviden összefoglalva: a WDS–WIM–WSIM az a három eszköz, ami fogja majd a kezünket a bevezetés teljes folyamatán keresztül. A WIM tartalmazza az adatokat, a WSIM által kreált válaszfájl a beállításokat, mindezt pedig a WDS juttatja el a számítógépekre.

A verziófrissítés működése, avagy hogyan húzzuk ki a szőnyeget a Windows XP lába alól

A Windows 2000-ről Windows XP-re történő frissítés esetén az történt, hogy a rendszer a dokumentumainkat és alkalmazásainkat a gépen hagyva, mindössze a \Windows

A TELEPÍTÉS LÉPÉSEI ÉS A TESTRE SZABÁS LEHETŐSÉGEI

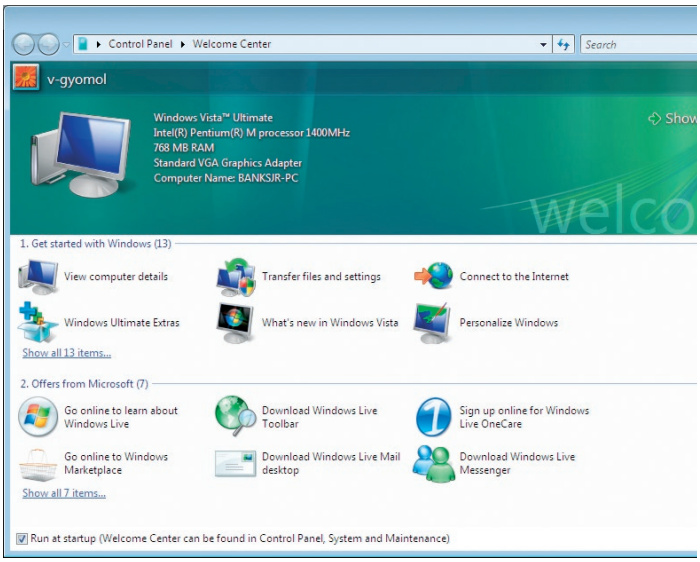
Telepítési fázis	Mi történik?
Downlevel-telepítés (tiszta telepítés/frissítés) telepítési forrásból indítva vagy WinPE-telepítés	Downlevel esetén: a telepítési beállítások (telepítés helye, termékkulcs) megadása: ▪ a telepítővarázslóban manuálisan vagy ▪ Unattend.xml WinPE esetén: a windowsPE beállítási fázisban található válaszfájl feldolgozása
Általános telepítési lépések	1. A lemez beállítása 2. A Windows Image fájl másolása a merevlemezre 3. A rendszerindító adatok előkészítése 4. A válaszfájl offlineServicing beállításainak alkalmazása (külső telepítések)
Online futási beállítások	A Windows egyedi beállításainak létrehozása: ▪ egyedi Security ID (SID) ▪ Plug-and-Play eszközök telepítése ▪ Legacy-Non-PNP eszközök telepítése ▪ a válaszfájl onlineServicingConfigurationPass beállításainak alkalmazása
Windows OOBE (Out-Of-The-Box-Experience)	1. Válaszfájl oobeSystem részének feldolgozása 2. Oobe.xml tartalmi beállítások alkalmazása 3. A Windows Vista első indítása

könyvtár tartalmát felülírva és a regisztrációs adatbázis legnagyobb részének átvételével települt. Ez az esetek nagy részében problémákat okozott a nem tökéletes visszafelé kompatibilitás miatt. A Windows Vista telepítője ezzel szemben a frissítő telepítés esetén is egy tiszta rendszert telepít (tehát nem veszi át a registryt), majd a lementett alkalmazásokat és dokumentumokat egy snapshotból állítja vissza.

A Business Desktop Deployment Toolkit (BDDT)

A BDDT eszköztára egy csomagban elérhetővé teszi a Windows Vista és az Office 2007 bevezetéséhez szükséges összes eszközt, technológiát, tudást és segédletet, ami szükséges lehet a vállalatok szakemberei számára. Ami ugyancsak újdonság, hogy a Microsoft a BDDT-t hivatalosan támogatni is fogja a megfelelő csatornákon keresztül, tehát probléma esetén végre lesz kihez fordulni. A BDDT ennek a cikknek a megjelenésekor már elérhető is lesz a http://connect.microsoft.com oldalon.

A végére egy kis finomságot hagyva, ejtsünk pár szót a Welcome Centerről. A Welcome Center egy olyan alkalmazás, amelyik a számítógép minden egyes elindításakor megjelenik, egészen addig, amíg a felhasználó ki nem kapcsolja. A Welcome Center egyszerű, és központi helyen nyújt módot arra, hogy a rendszergazda által kialakított lehetőségeket (programok telepítése, belső weblapok stb.) a felhasználók számára a rendszer automatikusan felkínálja.



A Welcome Center

Mélyebben a portálon

A bevezetésről és az azt segítő technológiákról további mély, nagyon részletes információkat találunk a TechNet portálon, a http://www.microsoft.com/technet cím windowswista pontjánál és a hamarosan megjelenő Resource Kitben is.

Moldova György
MCSE+I, MVP, MSS (v-gyomol@microsoft.com)
Microsoft Magyarország

WSUS 3.0 Beta 2

Megérkezett a Microsoft központi frissítéskezelő és telepítő alkalmazásának következő változata, egyelőre csak tesztverzióban.

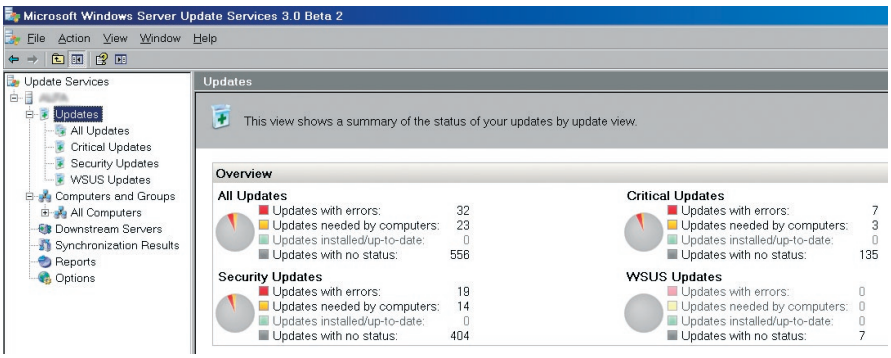
Majdnem egy éve írtunk utoljára ezeken a hasábokon a WSUS-ról (Windows Server Update Services), a Microsoft kis- és középvállalatoknak szánt, központi frissítéskezelő és telepítő alkalmazásáról. A termék második változata akkor már hónapok óta bárki számára ingyenesen letölthető volt, és nagyjából a cikk írása idején érkeztek az első hírek arról, hogy – a szokásosnál zártabb körben – elindul a következő verzió tesztelése. Ez év augusztusának közepén viszont „kitárták a kapukat", így aztán a Microsoft Connect oldalon keresztül, megkötések nélkül lehetett jelentkezni az új WSUS tesztelésére. Mindez közvetlenül a Beta 2-es verzió megjelenése előtt történt, amelyet aztán nyomban ki is próbálhattunk, és immár másfél hónapos tapasztalat alapján be is tudunk számolni a változásokról.

A rögtön látható különbségek

Az eltérések a telepítéssel kezdődnek, de ez csak féligazság: a folyamat sokkal több lépésből áll ugyan, viszont e lépések jelentős része már ismerős lesz minden WSUS-üzemeltetőnek. Azaz nem történt más, mint hogy a régebbi verziók telepítését követő 5-6 konfigurációs lépést (proxybeállítás, frissítésszinkronizálás, időzítés, termék és kategória, illetve nyelv kiválasztása stb.) integrálták egy új, Windows Server 2003 R2 típusú varázslóba. A telepítési előfeltételeket tekintve már több a változás, íme a lista:

1. Windows Server 2003 SP1 + IIS6;
2. .Net Framework Version 2.0;
3. Microsoft Report Viewer Redistributable 2005;
4. MMC 3.0;
5. BITS 2.0- és WinHTTP 5.1-frissítések.

A kakukktojás a 3. pont, ugyanis a többi már integrált (például MMC 3.0 > R2), vagy valószínűleg korábban frissített, ellenben a Report Viewert külön le kell tölteni és telepíteni, bár ez abszolút egyszerű művelet. Sajnos, a listából az is kiderül, hogy a Windows 2000-kiszolgálókra a WSUS 3.0 már nem telepíthető, viszont a következő kiszolgálóverzióra (a Windows „Longhorn" Serverre) igen.



A WSUS MMC nyitóoldala

A telepítést nem számítva az első igazi különbség, amivel szembesülünk, a kezelőfelület alapjaiban történt váltás. A HTTP/HTTPS-ről az MMC 3.0-ra térünk át ugyanis az új WSUS-szal. E cserének számos előnye van, gondoljunk csak bele például a több szerver együttes kezelésének lehetőségébe. Ezt enyhén szólva körülményesen lehet csak megoldani a korábbi verzionál, most viszont az MMC-ben már megszokott módon (Connect to...) tetszőleges számú WSUS-szervert szervezhetünk be egy konzolba.

A megjelenítésben, a szűrésben és rendezésben is több változást hoz a kezelőfelület cseréje. A WSUS 2.0-ban 4 oszlopba szervezhettük például a frissítéseket (Title, Classification, Release Date, Approval), az MMC 3.0-ban viszont 15 különböző oszlopunk is lehet, azaz például olyan extra információkat is szerepeltethetünk minden sorban, mint a vonatkozó KB-cikk száma vagy az MSRC-kategória és -név. Hasznos dolog az „Apply to All Views" parancs is, amellyel az egyszer alaposan kigondolt oszlopkiválasztást rögzíthetjük az összes létező nézetre.

A frissítések rendezésénél van még további újítás is: bevezették a például az Outlookban is látható vízszintes csoportosítást, a frissítések besorolását alapul véve.

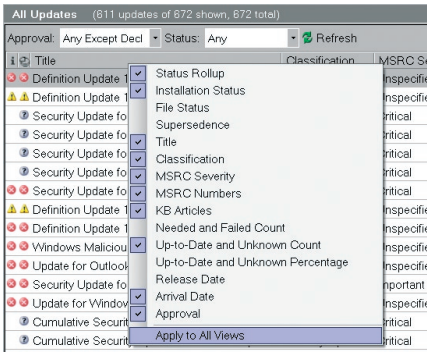
Újdonságok a működésben

Kicsit talán nehézkesebb lett az admin gépéről elérni a WSUS-szervert, mivel a HTTP-vel működő verziónál nem kellett semmit sem telepítenünk. Most viszont – miután gondoskodtunk a korábban felsorolt előfeltételek 2–4. pontjának meglétéről – a telepítő mappájában adjuk ki a következő parancsot a konzol telepítéséhez:

```
wsussetup.exe Console_install=1
```

A korábbi cikkekben beszámoltunk azok-

ról a nagyon munkaigényes praktikákról is, amelyek szükségesek voltak ahhoz, hogy alacsonyabb jogosultsággal is lehessen használni a WSUS-konzolt, például a jelentések megtekintése miatt. A Microsoft viszont „kapcsolt", és lehetővé tette ezt az új WSUS-ban mindenféle extra teendő nélkül. A megoldás elegáns, a címtárban vagy a helyi csoportok között észrevehetünk a telepítés után egy WSUS Administrators, illetve egy WSUS Reporters biztonsági csoportot, ezekbe tetszés szerint pakolhatunk felhasználókat. Pillanatnyilag csak egy probléma látszik ezzel kapcsolatban, nevezetesen, hogy a címtárban létrejövő csoportokba hiába raktuk be a korlátozott felhasználót, a kliensen csak akkor működött, ha a helyi Reporters csoport-



Oszlopmegjelenítési opciók

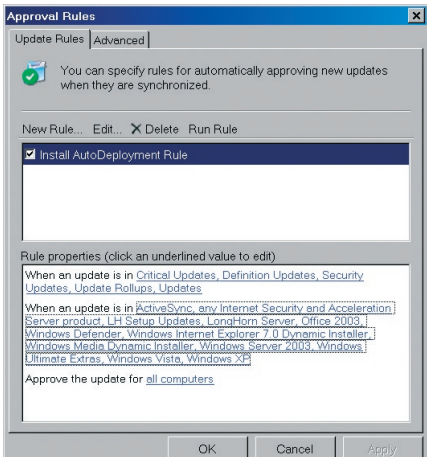
ban szerepelt a tartományi fiók. Valószínűleg erre születik javítás a későbbiekben.

Az előző verzióban a WSUS-kliensek csoportosítása (targeting) újdonságnak számított, ennek megfelelően bizonyos korlátokkal valósult csak meg. Egy gép maximum egy csoport tagja lehetett, és a csoportok egymásba ágyazása sem jöhetett szóba. Ez nyilvánvalóan az áttekinthetőség rovására ment, hiszen csak olyan rendszert hozhattunk létre, amelyik sok csoporttal és lapos hierarchiával rendelkezett. A WSUS 3.0 viszont változtat ezen a helyzeten, mindkét korlátot sikerült feloldani, így nagyobb vagy bonyolult hálózatok esetén is átlátható hierarchiát hozhatunk létre.

A megerősítések területén is van fontos változás, bővült az automatikus megerősítések szabályozhatósága. Készíthetünk szabályokat (szintén az Outlookhoz hasonlítható módon) a különböző termékek, illetve a frissítések típusának felhasználásával, ami azért jó, mert könnyedén engedélyezhetjük az automatikus telepítést a kevésbé kritikus komponenseknél (például Office-alkalmazások), és óvatoskodhatunk manuális megerősítéssel a fontosabb szoftvereknél.

A jelentésekkel kapcsolatban nem beszélhetünk apró változásokról, csakis a teljes átalakulás kifejezés jöhet szóba, ugyanis tényleg gyökeresen más lett ez az alrendszer – nem véletlenül előfeltétel a telepítésnél említett plusz komponens. Egyrészt jelentős számú előzetes beállítási, szűrési lehetőségünk van, másrészt amellett, hogy a végeredmény az összes kategóriában igazi vizuális élmény, nagyon részletes is, harmadrészt a jelentéseket elmenthetjük jól felépített Excel-és pdf-formátumban, negyedrészt a nyomtatással kapott végeredmény is megfelel az elvárásoknak.

A termék frissítésével kapcsolatosan – a tapasztalatok szerint – sok probléma nincs, a Beta 2-vel helyben végzett (in-place) frissítés teljesen simán, mindenféle probléma nélkül lezajlott éles környezetben is. Tegyük hozzá, hogy egy viszonylag kis hálózatban működő WSUS-t frissítettünk (amelyiknél viszont az WSUS 2.0 SP1 frissítésekor voltak azért komoly problémák!) gyakorlatilag csont nélkül, minden beállítás megmaradt, az adat-



Szabályokat hozhatunk létre a megerősítésekhez

bázis szintjén az átállás az SQL Server 2005 Express Editionre szintén nem okozott problémát (egyébként a rendszer az SQL Server 2005 teljes változataival is kompatibilis), sőt az egész folyamat mindennel együtt nagyjából félórát vett igénybe. Ha a hálózatunkban több WSUS-szerver is van, akkor nem árt viszont tudni, hogy a WSUS 2.0 képes szinkronizálni az új verzióval, de a fordított eset nem lehetséges.

A szerveroldali frissítés után a klienseket gyorsan megtalálta a WSUS 3.0, majd a frissítésük is megtörtént (a wuauclt.exe verziószáma jelenleg: 7.0.5451.90). Ha már a kliensnél tartunk, meg kell említenünk azt is, hogy az augusztus közepén kiadott Beta 2 nem támogatta még sem a Windows Vistát, sem a Windows „Longhorn" Servert (akkor sem, ha az előző a frissítési kategóriákban már választható volt).

Szeptember közepén azonban kaptunk hozzá pótlásképp egy frissítést, amelynek telepítése után ez a helyzet megváltozott – igaz, egyelőre csak a Vista RC1 tekintetében –, de várhatóan a Vista-támogatás alapértelmezett lesz majd a jövőben.

Kiegészítő újdonságok

A WSUS 2.0 alapos megismerése után sokunkban valószínűleg támadt némi hiányérzet a kiegészítő szolgáltatások iránt. Aztán az interneten sorra jelentek meg azok a plusz szkriptek, alkalmazások (egy-kettő még a Microsofttól is), amelyek praktikus kiegészítőként sokat segíthettek az üzemeltetésben. Nos, a WSUS köré tartozó eszközökből egy adag immár bekerült magába a termékbe is. Ilyen újdonság a Server Cleanup Wizard, amellyel a lejárt frissítések, a frissítések régebbi – már érvénytelen – verziói, az elutasított vagy nem használt frissítések, az ezekhez passzoló metaadatok és a 90 napnál régebben „látott" számítógépek is törölhetők a rendszerből. A varázslás végén egy kategóriák szerinti összegzést is kapunk a „tisztogatásról". A különböző források szerint egyelőre több WSUS-szerver egy konzolon való használata esetén ezzel a komponenssel lehetnek még problémák.

Végül essen szó még egy új és fontos opcióról, a különböző eseményekhez kapcsolódó értesítési lehetőségekről, azaz például e-mailben kaphatunk információt a frissítések szinkronizálásáról, illetve kérhetünk napi/heti jelentéseket, időzítve is.

Ebben a cikkben – többek között a termék bétaállapota miatt – nem említhetjük meg az összes apró vagy jelentősebb változást, illetve újdonságot, viszont a Microsoft magyar nyelvű Technet-weboldalán folyamatos tájékoztatást adunk, többek között a WSUS 3.0-val kapcsolatban is.

Gál Tamás
(gtamas@tjszki.hu) MCSE, MCSA, MCT, MVP

A PKI-INFRASTRUKTÚRA KIÉPÍTÉSE

Fokozott biztonságú hitelesítő központ kialakítása elektronikus levélaláíráshoz, Windows PKI-infrastruktúra alapokon.

Hogyan fogjunk hozzá egy nagyléptékű Windows alapú PKI-rendszer kivitelezésének? Cikkünk egy közelmúltban befejezett projekt fontosabb koncepcionális követelményeit, a felmerült problémákat és azok kezelését, valamint a kivitelezés fázisában választott megoldásokat ismerteti. Nem tekintjük ugyan feladatunknak a PKI és az intelligens kártyával kapcsolatos alapfogalmak tisztázását, de ha azzal kapcsolatos kérdés merülne fel az olvasóban, arra e-mailben szívesen válaszolunk.

A követelmények

Az elvárás legalább 5000 különféle, egymástól független, munkacsoportban vagy különböző tartományi tagként üzemelő munkaállomáson digitális aláírás megvalósítása elektronikus levélben, több ezer felhasználó számára. A feladat magában foglalta a központi szerverinfrastruktúra tervezését, implementációját, valamint a kártyákkal kapcsolatos műveleteket, egészen a helyszíni telepítésig, a kártyaátadásig.

A digitális aláíráshoz szükséges tanúsítványt és a kulcsokat is intelligens kártyán (smartcardon) kellett tárolnunk. Az intelligens kártyával kapcsolatos követelmények szerint a kiadott tanúsítványhoz tartozó kulcsméretnek legalább 1024 bitesnek kellett lennie; a kártya erőforrásait a felhasználó munkaállomásának oldaláról szabványos interfészeken (CSP, CAPI/CryptoAPI) keresztül kellett elérhetővé tenni. Követelmény volt, hogy a kriptográfiai műveleteket szabványos APDU-üzenetekkel kell elvégezni, valamint a kártyák védelmi funkciója legalább Common Criteria szerint EAT. 3-as vagy ITSEC E2-es, illetve ezeknél magasabb garanciális szinteken legyen.

Ezeket a követelményeket a kártyagyártók korszerű kártyái teljesítik. A mi választásunk az Axalto legkorszerűbb kártyájára esett. Fontos követelmény volt ezen kívül, hogy a kártyák legyenek fokozottan védettek a tanúsítvány és a titkosító kulcsok első használatáig. Ez utóbbi a gyakorlatban a kártyák telephelyekre történő leszállításáig és átadásáig jelent fizikai és adminisztrációs védelmet.

Fokozott biztonságú rendszerről lévén szó nem volt szabad megfeledkezni – fizikai tárolás tekintetében – a PKI-infrastruktúra szervereiről sem. Ezeket olyan speciálisan elzárt helyen kell tárolni és üzemeltetni, amely megfelelő garanciát ad a fokozott biztonságú minősítéshez. Erre a leghatékonyabban a szerverszobában elhelyezett, de zárható rackszekrény kínálkozott megfelelő megoldásnak. A fizikai védelmet természetesen adminisztratív védelemmel kellett kiegészíteni, hogy ne tudja akárki kinyitni a szekrényt. További biztonságfokozási célzattal a CA- szerver privát kulcsát HSM modullal kellett védeni.

A szerverek rendelkezésre állásának kívánt mértékét 99,5 százalékban adta meg a megrendelő.

A technikai kialakítás mellett fontos szerepet kapott a szabályozás. Az üzemeltetés, a tanúsítványkiadási, visszavonási folyamatok szabályozása, megoldása is a feladatunk része volt. Az üzemeltetést kiegészíti egy auditálási rend, amely biztosítja a megfeleltetést a törvényi követelményeknek. A törvényi előírásokról és egyéb rendelkezésekről részletesen olvashatunk például a http://www.bhsz-m.gov.hu/ weblapon barangolva.

A megvalósítás

Mint látható, a feladat egyáltalán nem volt egyszerű. A megrendelő kritériumrendszere szerint a teljes szerverrendszert 4 + 1 különálló biztonsági zónában kellett kiépíteni. A zónákat nemcsak fizikai szempontból, hanem adminisztratív, szabályozási oldalról is védetten alakítottuk ki.

Az egyes zónák:

1. Internetzóna – (+1) az internetes forgalom, külső tűzfallal védett.

2. DMZ webszerverzóna – két tűzfallal határolt, a külső tűzfalról csak a webszerverek

érhetők el, a webszerverek viszont elérhetők a belső szerverzónából.

3. Szerverzóna – a címtár és a CA-szerver védett zónája.

4. Ügyfélszolgálat – az ügyfélszolgálat munkatársainak speciális szegmense.

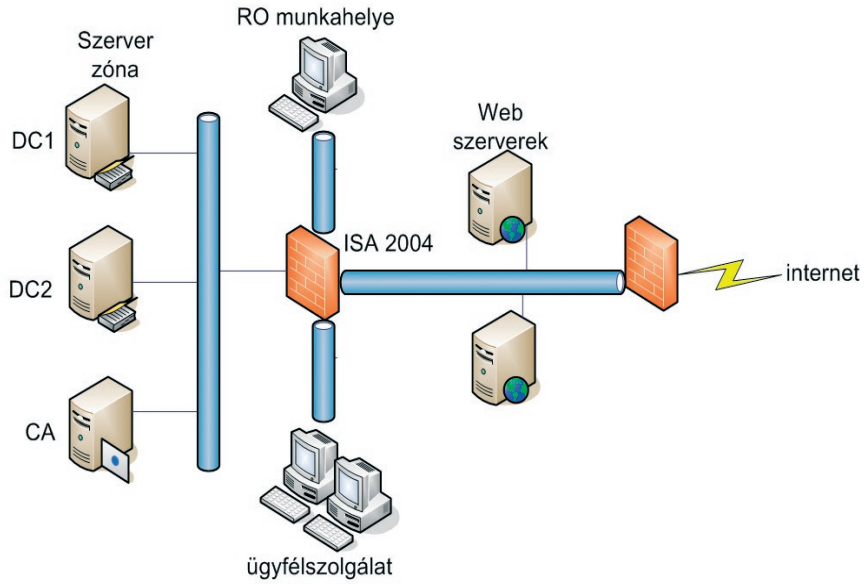
5. Az RO munkahelye – az RO számára (a szerepkör magyarázata később) kialakított speciális biztonsági zóna.

Szerverzóna

A szerverzóna tartalmazza a megfelelő biztonság és a választott redundáns megoldások szerint a szervereket.

A tartományvezérlő szerverek a leendő felhasználói adatbázis címtárban történő tárolását, valamint a felhasználókhoz tartozó tanúsítványok tárolását hivatottak megoldani. Mivel adatbázisról van szó, nem elég redundáns diszken tárolni az adatbázist, annak replikált példányát is üzemeltetni kell a megfelelő rendelkezésre állás érdekében.

CA-szerver esetében – hasonló vizsgálatok eredményeképp – a diszk-alrendszer kialakítása szintén RAID1 tükrözött tömbre került. A CA-szerver esetében a kialakítandó infrastruktúra megkövetelte a V2 tanúsítványok használatát; ez csak és kizárólag AD integrált CA-szerveren – pontosabban Windows Server 2003 Enterprise Editionön – lehetséges. A redundancia kritériumának egyedül a CA-szerver nem tud maradéktalanul eleget tenni, mert a beépített HSM modul oly mértékben érzékeny a fizikai behatásokra, hogy egy elhibázott mozdulatból eredő koccanás is tönkreteheti.



Speciális zónák

Tanúsítvány-adminisztrátorok esetében (ügyfélszolgálat és RO) – biztonsági és kompatibilitási megfontolások miatt – a kialakított webfelületre intelligenskártya-hitelesítéssel lehet csak belépni. Ez a tanúsítvány csak és kizárólag a hitelesítést szolgálja, a digitális aláírást nem.

A CA-szerver kialakításakor kihasználtuk a Microsoft Windows Server 2003 Advanced Edition által elérhetővé tett jogok szétválasztását, amely szerint ha a CA-szerveren az adott jogkörhöz tartozó adminisztrációs csoportba sorolt személyek más jogkörhöz tartozó műveletet szeretnének végezni, nem elég, hogy a rendszer nem engedi azt, de még a felhasználói fiókot is zárolja.

Belső tűzfal

A szerverzónát védő tűzfal esetünkben ISA 2004 lett, mivel ez volt az egyedüli olyan termék, amelyik képes volt a tervezés pillanatában a hálózat szegmenseit egymástól függetlenül kezelni, valamint a szegmensekhez tartozó SHTML-forgalomban tartalmi szűrést végezni, garantálva a szerverzóna sérthetetlenségét. Az ISA 2004 szerver kialakítása során figyelembe kellett vennünk azt a tényt, hogy a felhasználók és adminmunkakörök hitelesítése ezen a ponton is megtörténik – ráadásul intelligenskártya-eszközzel –, és ennek megfelelően el kell tudnia érni az AD címtárat. Ez csak akkor lehetséges, amennyiben az ISA szerver tagja a tartománynak, ahol a felhasználók megtalálhatók. A RADIUS rendszerrel kapcsolt (és általánosságokban is javasolt) hitelesítési módozatokat, a bonyolultabb hitelesítési eljárásokat a rendszer képtelen kezelni. Hosszas vizsgálódást követően arra a következtetésre jutottunk, hogy nem jelent többletkockázatot az ISA 2004 szerver behelyezése a belső tartományba, mivel a rendszer felhasználóinak köre ismert és korlátozott. Mindezek mellett az ISA szerverben levő naplóállományokból egyértelműen kiderülhet, hogy ki, mikor és honnan próbált megkérdőjelezhető módszerekkel bejutni a rendszerbe.

Webszerverzóna

A CA-szervernek további funkciójaként a kiadandó tanúsítványok érdekében rendelkeznie kell egy tanúsítványkiosztó webfelülettel. A megszokott felületet módosítanunk kellett, ennek oka egyrészt a megrendelő által támasztott grafikai arculat, valamint a megváltozott funkcionalitás. Ennek megfelelően négyféle felületet valósítottunk meg.

Végfelhasználói felület. A felhasználó a tanúsítvány kérését, a tanúsítvány meghosszabbítását végzi. Ennek egyszerűnek és egyértelműnek kell lennie: végfelhasználói ismeretekkel is, magyar nyelven tudjon dolgozni a felhasználó.

Ügyfélszolgálati felület. Az ügyfélszolgálati munkatárs a felhasználó kérésére felfüggesztheti a tanúsítványt, vagy felfüggesztésből képes visszaállítani az eredeti állapotra.

Registration Authority (RA) felület. Az RO, az RO munkaállomáson dolgozó adminisztrátor – az ügyfélszolgálattal együttműködve – képes visszavonni a véglegesen kompromittálódott tanúsítványokat, illetve bizonyos speciális esetekben tanúsítványkiadást végez.

A tanúsítvány státuszát lekérdező felület. Bármely felhasználó ellenőrizni tudja, hogy egy rendszerben kiadott tanúsítvány az adott pillanatban milyen állapotban van.

A webszerverek funkciói a CRL és Delta CRL publikálása, valamint a CA-szervertanúsítvány közzététele. Minden PKI-infrastruktúra lényeges eleme, hogy mennyire biztos a kliensek számára oly fontos CRL és

Delta CRL elérhetősége. Hasonlóan az AD címtárakhoz, itt is törekedni kell a folyamatos működést biztosító állandó elérhetőségre. A lehetséges megoldásokat megvizsgálva a választás a Microsoft Windows Server 2003 Web Editionre esett, amely képes NLBS szolgáltatáson keresztül a magas rendelkezésre állásra, ezen kívül a megoldásnak hatalmas előnye a terheléselosztás is.

A szerverméretezések során itt is elégségesnek bizonyult a Microsoft ajánlása, mivel dinamikus oldalt ez a webrendszer nem tartalmaz. A CRL- és Delta CRL-állományok alapértelmezetten a CA-szerveren keletkeznek, és ott is tárolódnak. Esetünkben ezeknek az állományoknak a mozgatása a CA-szerveren implementált, és megfelelő jogosultsággal, illetve a jogszeparációt követően megfelelő csoporttagsággal rendelkező felhasználóhoz rendelt, időzítetten futó scripttel oldottuk meg. A futó script az elkészült CRL- és Delta CRL-állományokat a CA-szerverről feltölti a webfürtökre.

A tesztek során két problémába ütköztünk. Az első, jelentősnek nevezhető gond az volt, hogy a webszerver bizonyos esetekben zárolja a CRL- és Delta CRL-állományokat, ezáltal az FTP-felöltést végző script nem képes ellátni a feladatát. A megoldást a script időzítésének gyakorisága jelentette, amely kompromisszum ugyan, és nem teljesen up-to-date állapotot jelent a CA-szerver és a webfürtök esetében, de bőven belefért abba a 10 perces tűréshatárba, amit a PKI-infrastruktúra ilyen esetekben képes elviselni.

A pontos méréseink kevesebb, mint 120 másodperces időt állapítottak meg, és ezzel a megoldással a megrendelő is maximálisan elégedett volt. A másik fő probléma biztonsági kérdéseket vetett fel, ugyanis a webszerverek a megrendelő többi szervere által is használt DMZ zónában voltak kihelyezve az ISA 2004 szerver egyik publikus lábára kötve. Az általunk alkalmazott megoldás kiterjedt az FTP-forgalom védelmére, ezért a teljes FTP-forgalmat IPSec-csatornába ágyaztuk be, preshared kulcsok használata mellett. Végezetül erre a webfürtre helyeztük ki az üzemeltetési szabályzatot is.

Szerepkörök a tanúsítványkiosztóban

Rendszergazda. Feladata a rendszer üzemeltetése, a rendszergazdai feladatok ellátása mellett a napi mentések elkészítése.

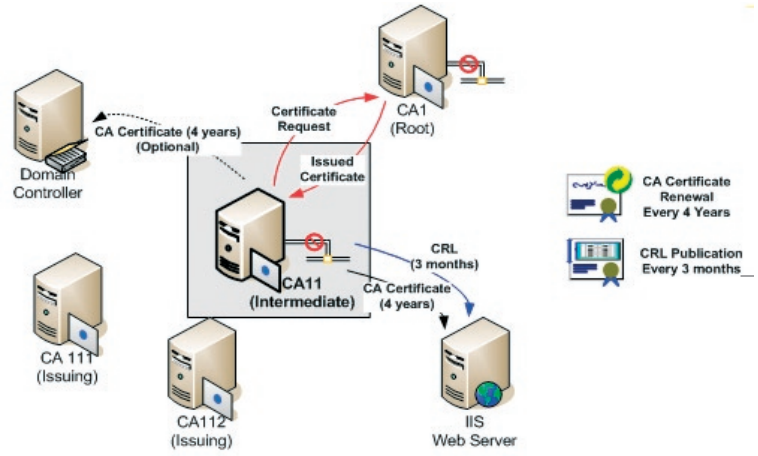
CA-szerveradminisztrátor. Kizárólagosan végzi a CA-szerver tanúsítványsablonjainak módosítását, valamint CRL- és Delta CRL-állományok manuális kibocsátását.

Biztonsági felügyelő – Security Officer (SO). Elsősorban adminisztrációs védelemmel és a fizikai hozzáférés korlátozásával védi a rendszert. Esetünkben a szervereknek helyet adó rackszekrény kulcsához csak az ebbe a csoportba tartozó felhasználók férhetnek hozzá, portaszolgálaton keresztül. A másik alkalmazott védelem a rackszekrénybe szerelt KVM-kapcsolóhoz egyedi jelszóval történő belépés. Enélkül a rendszeradminisztrátor képtelen használni a szerverekhez fizikailag csatlakozó billentyűzetet.

Regisztrációs felügyelő – Registration Officer (RO). A dedikált munkaállomásán tanúsítványok visszavonásával és kiadásával foglalkozik. A tanúsítványkiadás magában foglalja a rendszerüzemeltető személyzet által használt és alkalmazott hitelesítési célt szolgáló tanúsítványokat is. Ez utóbbi tanúsítványokat csak akkor képes kiadni az RO, amennyiben előtte az SO a saját kártyáján tárolt tanúsítvány segítségével az RO-munkaállomás és az ISA szerver közötti csatornát ki nem nyitotta.

Ügyfélszolgálat. A helpdesk nem más, mint valamennyire „butított" RO-funkció, ahol az adminisztrátori személyzet csupán tanúsítványfelfüggesztést és felfüggesztésből történő helyreállítást képes kivitelezni. Ezzel a megoldással tehermentesítettük az RO-t, és még egy ellenőrzési lépcsőt iktattunk be a végérvényes tanúsítvány-visszavonás előtt.

Auditor. A szerepkör elkülönül az IT-üzemeltetéstől, és az ide sorolt munkatársak semmiféle kapcsolatban nem állnak a rendszerüzemeltetést végző csapattal. Feladatuk a CA-szerveren keletkező naplóállományok részletes elemzése, valamint azok összevetése az üzemeltetési naplókkal. Napi riportokat készítenek a vezetés számára, amelyben jelzik mind a normális, mind az attól eltérő működést.



Egy tanúsítvány megújításának folyamata

Tanulságok

Ez a cikk jól szemlélteti, hogy egy PKI alapú, fokozott biztonságú rendszer kialakítása egyáltalán nem egyszerű, és nagyon komoly körültekintést igényel még úgy is (vagy pedig pont emiatt még fokozottabban!), hogy az operációs rendszerként használt Microsoft Windows Server 2003 már pár kattintással rengeteg funkciót tesz elérhetővé. Az alapos tervezés és az implementáció minőségének biztosítása amiatt is elengedhetetlen, hogy a kész rendszer valóban megfeleljen az eredeti követelményeknek, és adott esetben egy egyszerű szerverleállás vagy valamely kritikus lemez megsérülése ne tegye az egész informatikai rendszert teljesen használhatatlanná pillanatok alatt.

A technikai megoldások mellett a projektben a fő hangsúly a folyamatok, a törvényi megfelelőségek biztosításán volt, ami egyre több cég esetében jelent egyre nagyobb kihívást. Habár a projektben csak levélaláírásra használtuk a tanúsítványokat, érdemes meggondolni hasonló tanúsítványkiadó kialakítását, ha a különböző tanúsítványokat központosítva akarjuk kezelni.

Urbanovits György
(Urbanovits.gyorgy@euroone.hu), MCSE, MCT
Kiss Mihály
(kiss.mihaly@euroone.hu) MCSE
Babócsy László
(Babocsy.laszlo@euroone.hu), MCSE

SQL Server 2005 Reporting Services

Ha valaki saját alkalmazásához jelentéskészítő és megjelenítő eszközt szeretne illeszteni, vagy jelentésplatformot kell választania, akkor jól teszi, ha megismerkedik az SQL Server 2005 Reporting Services által nyújtott alkalmazásintegrációs lehetőségekkel.

Fejlesztők, alkalmazástervezők számos alkalommal kerülnek szembe azzal a kérdéssel, hogy milyen eszközt, platformot használjanak saját alkalmazásaikhoz jelentéskészítésre. Az SQL Server Reporting Services (SSRS) megjelenéséig legtöbbször nem a Microsoft, hanem valamilyen más gyártó termékét kellett választaniuk, mivel a Microsoftnak egyszerűen nem volt jelentésplatformja. Az egyetlen valamirevaló eszköz az Access volt, ami ügyfél–kiszolgáló környezetben kiváló, de többrétegű vagy webalkalmazásokhoz nem való.

Mit használjunk jelentéskészítésre?

A Reporting Services viszont olyan jól sikerült, és olyan sokrétűen használható, illeszthető be a saját alkalmazásainkba, rendszereinkbe hogy ma már az esetek többségében az SSRS a legjobb választás. Következzen itt egy gyors – és „szubjektíven szelektív" – összefoglalás arról, hogy mi mindenre használható, és miért is ennyire jó az SSRS.

Önálló jelentéskészítő, publikáló megoldásként is megáll: viszonylag egyszerű telepítés, konfigurálás, aztán hajrá, tolhatjuk alá a Visual Studio alapú Report Designerrel vagy a felhasználók által egyszerűen használható Report Builderrel összerakott jelentéseket.

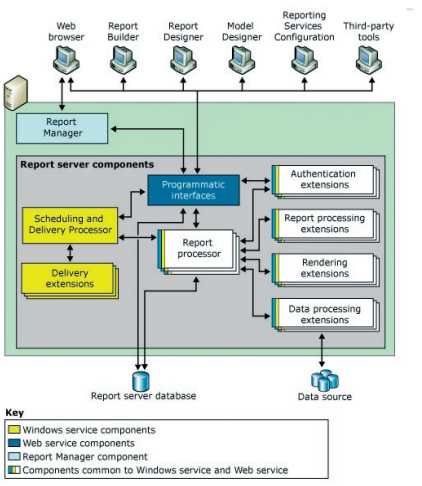
A felhasználók kapnak egy portált, amelyen egyszerű eligazodni, és alig egy óra alatt meg lehet tanulni a használatát. A jelentéseket PDF-ben, Excelben, HTML-ben, XML-ben, CSV-ben, TIFF-ben lehet nézegetni, kinyomtatni, lementeni. Meg lehet őket rendelni e-mailben, vagy ha úgy akarjuk, akkor minden reggel készen várhatnak minket a hálózaton lévő könyvtárunkban. A rendszergazdák ugyanezt a portált használhatják publikálásra, a hozzáférés szabályozására, paraméterek definiálására és még számos feladat megoldására.

Többféle módon illeszthető saját alkalmazásainkhoz, portálmegoldásainkhoz. Az egyszerű URL-en keresztüli eléréstől egészen az általunk programozottan generált jelentések saját formátumban történő megjelenítéséig, a lehetőségek tárházát kínálja az együttműködésre.

Tökéletesen illeszkedik a cégeknél már eddig is használt Microsoft-technológiákhoz: AD integrált felhasználó- és jogosultságkezeléssel rendelkezik; webszolgáltatás, .Net, WMI-interfészeken keresztül programozható; szkriptelhető; skálázható; igen jól konfigurálható gyorsítótárral rendelkezik; webfarmot építhetünk belőle – és még sorolhatnánk. Ehelyett nézzük meg néhány példán keresztül, hogyan használhatjuk ki legjobban a Reporting Services szolgáltatásait, és hogyan integrálhatjuk azt saját alkalmazásainkkal.

Architektúra

Ahhoz, hogy az SSRS által nyújtott lehetőségeket lássuk, és tisztában legyünk a korlátok-



1. ábra. Az SSRS architektúrája

kal is, vessünk egy pillantást az SSRS architektúrájára *(1. ábra)*!

Az ábráról leolvasható, hogy a jelentések és a konfigurációs információk tárolása adatbázisban (SQL Server) valósul meg. A jelentések előállítása („rendering") a felhasználói kérést követően a Report Serveren történik, nem az ügyfélalkalmazásban, és maga a jelentés különböző közvetítőkön – riportportál, e-mail, fájlszerver, programozható interfészek – keresztül jut el az ügyfélhez.

Mivel az SSRS szervermegoldás, azaz a jelentések előállítása nem az ügyfélen, hanem a szerveren történik, önálló, egygépes jelentéskészítésre csak akkor alkalmas, ha minimálisan az SQL Server Express változatát és az Internet Information Servert is telepítjük az ügyfélgépre.

(Megjegyzés: a Visual Studio 2005-ben webes és Windows-változatban is használható ReportViewer vezérlőelem támogatja a jelentések ügyféloldali előállítását is.)

Az SSRS-t tehát akkor tudjuk a legjobban kihasználni, ha a felhasználók online elérik a Report Servert. A beépített (SMTP és fájlrendszer alapú) és bővíthető riporttovábbítási mechanizmusokkal azonban a közvetlen kapcsolat nélküli és kötegelt jelentéskészítést könnyedén meg tudjuk valósítani.

Alkalmazásintegráció

Alapkérdés: ha van egy webes vagy Windows-alkalmazásunk, hogyan használhatjuk az SSRS-t jelentéskészítésre, továbbá a jelentések kezelésére?

Természetesen többféle módon, attól függően, hogy mit akarunk megvalósítani. Tekintsük át először nagy vonalakban, aztán részletesebben a lehetőségeket.

URL-elérés. Egyszerű megoldás, amelyet webes és Windows-alkalmazásból is használhatunk. Gyorsan implementálható, viszont kevésbé alakítható. Lényege, hogy HTTP protokollon keresztül GET/POST kéréssel érjük el a Report Servert, amelyen különböző műveleteket hajthatunk végre.

ReportViewer-vezérlőelemek. Visual Studio 2005 webes és Windows-alkalmazásokban használható vezérlőelemek. Akkor érdemes használni őket, ha űrlapokon beágyazva szeretnénk jelentéseket megjeleníteni, és kontrollálni akarjuk a lekérdezési, megjelenítési folyamatot.

Report Server Web Service. Lehetővé teszi, hogy a Reporting Services minden szolgáltatásához hozzáférjünk az alkalmazásainkból. Mivel a jelentések megjelenítését általában sokkal egyszerűbben is megoldhatjuk, leginkább a jelentések feltöltésére, módosítására, törlésére, megrendelésére, jogosultságok kezelésére és tulajdonságok beállítására használjuk.

Report Definiton Language (RDL). Az RDL a Reporting Services jelentésdefiníciós nyelve. A nyelv XML alapú, nagyon jól dokumentált, és kiválóan alkalmas arra, hogy saját alkalmazásunkból, jelentésgeneráló eszközünkből Reporting Services alatt futtatható jelentéseket gyártsunk. Egy másik fontos alkalmazási terület lehet a jelentések kötegelt módosítása, például a jelentések adatforrásául szolgáló adatbázis-struktúra megváltozása miatt.

Reporting Services Extensions. A Reporting Services moduláris felépítéséből adódóan többféleképpen bővíthető. Bizonyos esetekben a bővítés jobb megoldás lehet, mint a saját alkalmazásunk „reszelése", így például saját adatforrásokat, jelentéspublikáló kiterjesztéseket és jelentésmegjelenítési formátumokat hozhatunk létre többkevesebb programozás árán.

URL-elérés

A legegyszerűbb módja a webalkalmazás-integrációnak az URL-en keresztüli elérés. Ez akkor előnyös, ha a webalkalmazásunkból egyszerűen akarunk jelentéseket futtatni, és nem akarjuk különösebben felügyelni a felhasználó és a Report Server interakcióját. Ebben az esetben az SSRS saját jelentésmegjelenítő felületén keresztül férünk hozzá a jelentésekhez, vagy a Report Server által támogatott formátumok valamelyikében kérhetjük le a riportokat.

Nézzünk erre egy egyszerű példát, amely egy webalkalmazásban http GET-et használ egy jelentés megjelenítésére:

```
<a href = „http://server/reportserver
?/AdventureWorks Sample Reports/Sales Order Detail
&rs:Command=Render&rs:Format=HTML4.0&rc:Link
Target=main&SalesOrderNumber=SO50750" >Sales
Order SO50750 details</a>
```

A hivatkozás az „AdventureWorks Sample Reports" mappa „Sales Orders Details" riportját hívja meg paraméterként átadva a megrendelés azonosítóját. Ahogy a példa is mutatja, az URL-elérés nagyon egyszerűen használható.

Az URL-t négy részre bonthatjuk:

1. A Report Server elérési útvonala

A Report Server alkalmazás URL-jét tartalmazza.

Példánkban **http://server/reportserver**

2. A jelentés elérési útvonala

Az URL első paramétere, egy jelentés, adatforrás vagy egy mappa teljes elérési útvonalát tartalmazza a szerveren belül.

Példánkban: **?/AdventureWorks Sample Reports/Sales Order Detail**

3. Report Server-parancsok, -paraméterek

A Report Servernek szóló utasítások és paraméterek, amelyek parancsokat, a jelentés formátumát, a bejelentkezési információt és egyéb feldolgozási opciókat határoznak meg.

```
private void FrmMain_Load(object sender, EventArgs e)
{
    List<ReportParameter> reportParameters  =
        new List<ReportParameter>();

    this.repViewer.ProcessingMode = ProcessingMode.Remote;
    this.repViewer.ServerReport.ReportServerUrl =
        new Uri("http://server/reportserver");
    this.repViewer.ServerReport.ReportPath =
        "/AdventureWorks Sample Reports/Sales Order Detail";

    reportParameters.Add(
        new Microsoft.Reporting.WinForms.ReportParameter(
        "SalesOrderNumber","SO50750"));
    this.repViewer.ServerReport.SetParameters(reportParameters);

    this.repViewer.RefreshReport();

}
```

2. ábra. A példa, a Microsoft.Reporting.WinForms.ReportViewer vezérlőelemmel megvalósítva

A Report Servernek szóló instrukciókat paraméter-előtagokkal látjuk el azért, hogy meg tudjuk őket különböztetni a jelentésparaméterektől.

Példánkban: **&rs:Command=Render** – azaz a jelentést meg akarjuk jeleníteni.

&rs:Format=HTML4.0 – a megjelenítési formátum legyen HTML 4.0.

&rc:LinkTarget=main – a jelentést a „main" keretbe kérjük.

4. Jelentésparaméterek

A jelentés paraméterei, úgy, ahogy azok a riportdefinícióban szerepelnek, egyszerűen a *paraméternév=érték* szintaxist használhatjuk. Ha null értéket akarunk átadni, akkor azt a *paraméternév:**isnull=true*** formában kell megadnunk. Példánkban: **&SalesOrder-**

Number=SO50750 – a SalesOrderNumber paraméter értéke legyen S050750.

A fenti példa használható Windows-alkalmazásban is: indíthatunk egy Internet Explorert, használhatjuk a *System.Windows.Forms.WebBrowser* vezérlőelemet, ha űrlapban beágyazottan akarjuk megjeleníteni a jelentést, vagy nagyobb beavatkozási lehetőséget szeretnénk, illetve ha teljesen kézben akarjuk tartani a folyamatot, a *System.Web.HttpRequest* osztállyal állíthatjuk össze a kérést, míg a választ a *System.Web.HttpResponse* osztállyal dolgozhatjuk fel.

ReportViewer-vezérlőelemek

A Visual Studio 2005-ben webes és Windows-vezérlőelemekkel űrlapokon mutathatjuk meg a jelentéseket .Net-alkalmazásainkban. Használatuk egyszerű, néhány paraméter beállításával már láthatjuk is az eredményt. A vezérlőelemek eseményein keresztül valamelyest ellenőrzésünk alatt tudjuk tartani a felhasználó–jelentés–Report Server interakciót, azonban magába a jelentés-előállítási folyamatba nem tudunk közvetlenül beavatkozni. A jelentés paramétereinek módosításával viszont, közvetetten, még ezt is meg tudjuk tenni. Természetesen ilyenkor a jelentést kell ennek megfelelően elkészítenünk.

A Report Viewer-vezérlőelemekkel lokális jelentések futtatására is van lehetőségünk, ilyenkor a jelentés definícióját vagy lokális fájlban, vagy beágyazott erőforrásként kezeljük, és a jelentést lokálisan futtathatjuk.

Az előzőleg említett példát a *Microsoft.Reporting.WinForms.ReportViewer* vezérlőelemmel megvalósítva a *2. ábra* szemlélteti. Mint láthattuk a ReportViewer-vezérlőelemek egyszerűen használhatók, de azért nem árt az óvatosság: ha a jelentés nagyon sok adatot jelenít meg, akkor bizony meg tudja terhelni a memóriát, mivel az egész jelentés letöltődik – a lapozás nem dinamikusan tölti a lekérdezés egyes részeit. Ilyen esetekben célszerű lehet saját lapozó mechanizmus készítése.

A Report Server-webszolgáltatások képezik a Reporting Services legáltalánosabban használható programozási interfészét. Az SQL Server 2005 Reporting Services két webszolgáltatás-belépési pontot tartalmaz.

Report Server Web Service

ReportService2005. Adminisztratív szolgáltatásokat tartalmaz, a Report Server beállítására, jogosultságkezelésre, a jelentések feltöltésére és módosítására stb. A végpontot a *http://server/reportserver/ReportService2005.asmx* hivatkozáson keresztül érhetjük el.

ReportExecution2005. A jelentések futtatására szolgáló webszolgáltatás-interfész. Támogatja jelentések tetszőleges formátumban való lekérését, a paraméterek magadását, a jelentéseken belüli navigációt könyvjelzők és dokumentumtérképek alapján, az egyes jelentéselemek láthatóságának módosítását, a keresést és a rendezést.

A végpontot a *http://server/reportserver/ReportExecution2005.asmx* hivatkozáson keresztül érhetjük el.

A *3. ábrán* látható kódrészlet az első példa webszolgáltatás alapú megvalósítását mutatja meg: a *ReportExecutionService* webszolgáltatás-osztály *Render* metódusát használjuk a jelentés futtatására, majd a jelentést egy fájlba mentjük, és egy WebBrowser vezérlőben jelenítjük meg.

```
<?xml version="1.0" encoding="utf-8" ?>
- <Report xmlns="http://schemas.microsoft.com/sqlserver/reporting/2
    xmlns:rd="http://schemas.microsoft.com/SQLServer/reporting/rep
  - <DataSources>
    + <DataSource Name="DSAdventureWorks">
    </DataSources>
    <BottomMargin>2.5cm</BottomMargin>
    <RightMargin>2.5cm</RightMargin>
    <PageWidth>21cm</PageWidth>
    <InteractiveWidth>8.5in</InteractiveWidth>
  - <Body>
      <ColumnSpacing>1cm</ColumnSpacing>
    - <ReportItems>
      + <Textbox Name="textbox1">
      + <Table Name="table1">
      </ReportItems>
      <Height>2.00635cm</Height>
    </Body>
    <rd:ReportID>8f5b1911-69ff-4237-8a55-59f538f07f19</rd:ReportID>
    <LeftMargin>2.5cm</LeftMargin>
  + <DataSets>
    <Code />
    <Width>12.69841cm</Width>
    <InteractiveHeight>11in</InteractiveHeight>
    <Language>en-US</Language>
    <TopMargin>2.5cm</TopMargin>
    <PageHeight>29.7cm</PageHeight>
  </Report>
```

4. ábra. Egy jelentés RDL-szerkezete

Reporting Services Extensions

A Reporting Services ötféle bővítést támogat, amelyeket különböző feladatok megoldására használhatunk. Mivel a bővítések programozása nem egyszerű feladat, általában érdemes végiggondolni, hogy nincs-e egyszerűbb út egy-egy probléma kezelésére.

Data Processing Extension. Lehetővé teszi, hogy új adatforrásokkal bővítsük a Reporting Servicest. Írhatunk például olyan kiterjesztést, amelyik bináris vagy szövegfájlokból képes adatot beolvasni. Mielőtt adatforrás implementálásába kezdenénk, fontoljuk meg, hogy vajon egyszerűbb-e az adatainkat a Reporting Services által már kezelhető formába (például XML-re) konvertálni vagy betölteni az SQL Serverbe.

Delivery Extension. A jelentések publikálását és megrendelését kezelhetjük vele tetszőleges saját megoldáson keresztül. Készíthetünk például olyan bővítést, amelyik egy webszolgáltatásnak küldi el a jelentést. Mivel a Reporting Services beépítetten a fájlrendszeren keresztüli és SMTP publikálást tesz lehetővé, alternatívaként az ezeket használó szolgáltatások (például SQL Server Integration Services, BizTalk) is alkalmasak az integrációra.

Security Extension. Saját felhasználóazonosító és jogosultságkezelő rendszer implementálását teszi lehetővé. Ez nagyon hasznos

```
private void FrmWSForm_Load(object sender, EventArgs e)
{
    Local variables
    string reportPath =
        "/AdventureWorks Sample Reports/Sales Order Detail";

    ReportExecutionService rs = new ReportExecutionService();
    rs.Credentials = System.Net.CredentialCache.DefaultCredentials;
    rs.Url =
        "http://server/reportserver/ReportExecution2005.asmx";

    ParameterValue[] parameters = new ParameterValue[1];
    parameters[0] = new ParameterValue();
    parameters[0].Name = "SalesOrderNumber";
    parameters[0].Value = "SO50750";

    ExecutionInfo execInfo = rs.LoadReport(reportPath, historyID);
    rs.SetExecutionParameters(parameters, "en-us");
    result = rs.Render(format, deviceInfo, out extension,
        out encoding, out mimeType, out warnings, out streamIDs);

    FileStream stream = File.Create(
        @"c:\OrderDetail.htm", result.Length);
    stream.Write(result, 0, result.Length);
    stream.Close();

    this.webBrowser1.Url = new Uri(@"file://c:\OrderDetail.htm");
}
```

3. ábra. Webszolgáltatás alapú megvalósítás

lehet olyankor, amikor a felhasználók kezelése nem AD-ben, hanem saját adatbázisban történik, mint például a nyilvánosan elérhető, bejelentkezést igénylő webalkalmazások esetében.

Rendering Extension. Saját megjelenítési formátumok definiálását támogatja. Ilyen lehet például a vállalatirányítási rendszer export/import formátuma, képformátumok, RSS vagy speciális szövegformátumok. Rendering Extensiont nem könnyű írni, akár több száz osztályt, metódust is kell implementálnunk, ezért célszerűbb inkább a beépített formátumokban készített jelentéseket konvertálni.

Report Processing Extension. A jelentésfeldolgozó bővítések lehetővé teszik, hogy a jelentésdefinícióban elhelyezett saját elemeket is feldolgozzuk, így saját jelentéselemeket definiálhatunk és jeleníthetünk meg. Készíthetünk például olyan bővítést, amely egy új grafikon típus (például tölcsérdiagram) előállítását támogatja.

Az SQL Server 2005 Books Online minden típusú bővítésre tartalmaz megfelelően beszédes példákat.

A Reporting Services jelentésdefiníciós nyelve, az RDL egy XML alapú nyelv, amely igen jól dokumentált, és elég egyszerű ahhoz, hogy különösebb nehézség nélkül állhassunk neki saját jelentésgenerátorunk megírásához, vagy a meglévő jelentések alkalmazásból történő módosításához.

Egy jelentés előállításához minimálisan a következőket kell tennünk:

1. Definiálnunk kell egy adatforrást a *DataSource* elem segítségével.

2. Meg kell adnunk egy adathalmazt a *DataSet* elemmel.

3. Létre kell hoznunk a jelentéselemeket, és el kell helyeznünk a jelentésen a *Body*/*ReportItems* elemben.

A *4. ábrán* látható lista egy jelentés RDL-szerkezetét szemlélteti, kihagyva az egyes elemek részleteit.

Az SQL Server 2005 Books Online egy részletes útmutatót (Tutorial) tartalmaz, amely Visual Studio 2005 használatával lépésről lépésre mutatja be az RDL programozott előállítását, és körülbelül egy óra alatt elvégezhető. A Books Online-ban az RDL részletes leírását is megtaláljuk, sajnos nem túl sok példával, úgyhogy mielőtt nekiállnánk RDL-t szerkeszteni, tanulmányozzuk át néhány Report Designerrel készített jelentés forrásállományát (.rdl fájlok).

Összefoglalás

A fentebb kifejtetteken kívül még számos egyéb eszköz áll rendelkezésünkre, hogy a Reporting Servicest saját környezetünkbe illesszük. Ilyen lehetőség például a saját assemblykben implementált függvények használata, az XML webszolgáltatások adatforrásként történő elérése, telepítő, konfigurációs szkriptek írása az *rs* segédprogrammal.

Az SSRS tehát nem „csak” egy jelentésszerver és a hozzá kapcsolódó portál, hanem ennél sokkal több: olyan jelentésplatform, amelyet szinte minden alkalmazáshoz lehet illeszteni.

Végezetül még egy érv az SSRS mellett: ingyenes kiadásban is rendelkezésre áll, mivel az SQL Server Expresszel együtt letölthető – így az olcsóbb alkalmazásokhoz, portálmegoldásokhoz is nyugodtan használhatjuk.

Kovács Zoltán
(kovacsz@szamalk.hu) vezető oktató, Számalk

SZOFTVERGYÁRAK

Mi köze van egymáshoz a kézművességnek, az autónak és a szoftvernek?

A szoftverek készítése ma sok esetben olyan, mint a kézművesség. A művész vázlatokat és tanulmányokat készít, aztán vesz egy darab friss agyagot (File – New – Project...), és elkezdi megtölteni tartalommal. Közben kísérletezik, visszalép, hozzátesz, elvesz, egyszóval teljesen egyedi műveket alkot újra és újra. Azok, akiknek rendszeresen kell éles környezetben használt szoftvert üzleti alapon előállítaniuk (és még nem mentek csődbe), már kidolgoztak rendszeresebb módszereket. Már van szereposztás, folyamatrendszer, és akár specializált eszközök is rendelkezésre állnak – ez már amolyan manufaktúra. Ennél persze lehet még messzebb menni, vannak, akik meg is megteszik, és például fokozottabban odafigyelnek a már meglévő, kipróbált és bevált építőkockák újrahasznosítására is.

A szoftvergyártás Ford T-Modellje a sablon. Egyszerű, általános céloknak megfelelő példányokat könnyen generálhatunk, ilyenek a Visual Studio sablonjai, de még inkább az egyes Starter Kitek, amelyek már önmagukban is kész alkalmazások, vagy elővehetünk olyan kész építőkockákat, blokkokat is, mint amilyenek például a Patterns & Practices mintái. Természetesen ezek vagy nagyon partikuláris igényeket elégítenek ki, és ezért ritkán igazán hasznosak, vagy túl általánosak, és ezért még sok munka van velük.

Az autógyártás mai modellje a tömeges testreszabás. Kihasználják a tömeggyártás előnyeit – a méretgazdaságosságot, a futószalagot stb., de mégis minden megrendelőnek képesek az egyéni igényeihez alakítani az autóját. Előre kiválasztott kárpit, hangrendszer és motor kerül bele abba a karosszériába, ami csak színében különbözik a futószalagon előtte haladótól. Ezt a működésmódot szeretnénk megvalósítani a szoftverek készítése során is, ez a szoftvergyár-koncepció.

Mi kell egy szoftvergyárhoz?

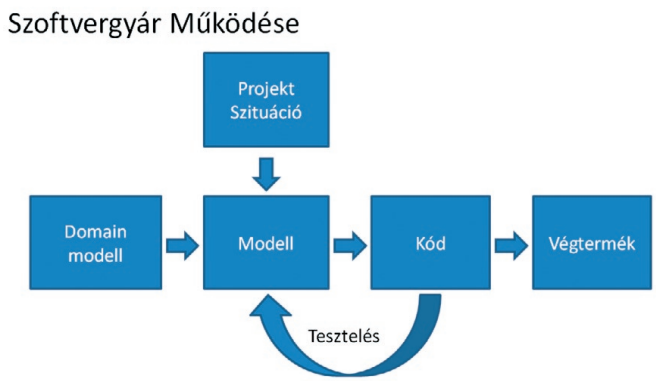
Ahhoz, hogy tömeges egyedi gyártást tudjunk véghez vinni, szükségünk van szerszámokra és gépekre. A szerszámokhoz és a végtermékhez is szükség van tervrajzokra. A tervrajz szerepe, hogy a végső termékhez képest egyszerűsített vázzal, egy modellel tudjunk dolgozni.

A modell pedig nem más, mint egy valódi, létező dolog egyszerűsített leírása. Modellek segítségével könnyebben láthatunk át komplex rendszereket. Még ha veszítünk is az általánosítás miatt a pontosságból, ezáltal tudunk a nagy egészre koncentrálni. Ahogy egyre nő a szoftverek mérete és komplexitása, úgy van egyre nagyobb szükség hatékony modellezési eszközökre, hogy a készülő rendszer biztosan helyes alapokon nyugodjon.

Erre találták ki a modell alapú szoftverfejlesztést, amiben modelleket alkotunk, a fejlesztési munkát a modell megfelelő kialakításába fektetjük, és ezután a végterméket ez alapján adja ki a futószalag. A végtermék azonban gyakran még mindig csak egy váz, amelyen még jelentős mennyiségű utómunkára van szükség, de így is hatalmas munkát spórol meg nekünk az automatizálás, hiszen kapunk egy bevált, tesztelt módszerek alapján készült, stabil alapot további fejlesztéseinkhez. Természetesen ez a futószalagos megoldás egyszerűen hangzik, de valakinek a modell modelljét és a futószalagot is ki kell fejlesztenie.

A szoftverfejlesztés mindeddig a lehető legáltalánosabb megoldásra törekedett, ilyen például az UML és a köré épített eszközrendszerek. Maga az UML – ami egyébként kiváló eszköz a szoftver egyes vetületeinek modellezésére – általánossága miatt nagyon bonyolult, és nem mindig kellően „mozgékony". Az UML-diagramok alapján generált vázosztályok és kódalapok nem mindig segítenek kellőképpen a tényleges fejlesztési munka során, emiatt még nemigen nevezhető futószalagnak a szoftverfejlesztés szempontjából.

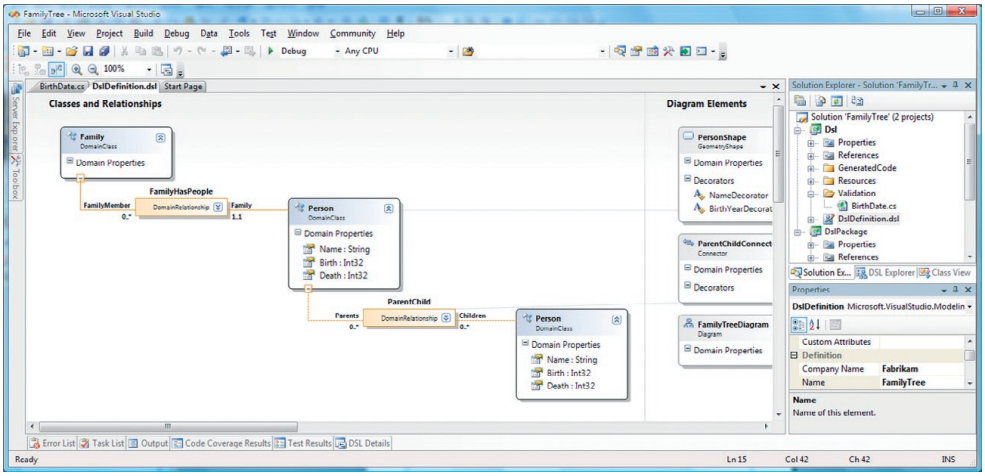
A másik irány a RAD- (Rapid Application Development) eszközök esetében figyelhető meg; ezek a minél gyorsabb és egyszerűbb kódgenerálást tekintik gyakorlati célnak, és ehhez legtöbbször szintén modelleket használnak. Ilyen modell például a Visual Studio Form Designere is, amivel Windows ablakok kódjának vázát készíthetjük el néhány kattintással. Ez már sokkal jobban megközelíti a futószalag elvét.



A szoftvergyár belső folyamatrendszere

A szoftvergyár-koncepció igyekszik e két megoldás egyensúlyát keresni, és a fejlesztő kezébe adni a választás lehetőségét, hogy mindenki felépíthesse magának azt a speciá-

lis modellezési környezetet, ami a legtöbbet segít neki. A cél szinte mindig a meglévő és bevált módszer újrahasznosítása, a modell alapú, egyszerűsített és áttekinthető tervezés, valamint a hatékony kódgenerálás ötvözése.



Egy egyszerű domain-modell a tervezőben...

A folyamat

De nézzük, hogyan építhetjük meg magát a futószalagot, vagyis a modellépítő modellt. Ez a domain-modell felállításával kezdődik. A domain-modell írja le, hogy szoftverünk modell alapú tervezésekor milyen elemekből és hogyan építhetjük fel majdani modelljeinket, és az egyes modellelemek milyen kódot fognak készíteni. A domain-modell tartalmazza az igazi know-how-t, ami idővel egyre csak gyarapodik, és akár egy gép vagy szerszámkészlet, önálló értéket nyer. Bár nem a DSL technológiát használják, de ilyen domain-modellnek lehet tekinteni az imént említett Form Designert vagy akár a Visual Studio 2005 UML-szerű Class Designerét is. Ezeket a domain-modelleket az adott szoftverfejlesztési terület legjobb ismerői hozzák létre, hogy segítsék mások munkáját.

Egy ilyen alapokon nyugvó szoftver fejlesztésekor valójában egy hatalmas, több futószalaggal rendelkező gyárnak aktiváljuk bizonyos részeit. Természetesen a gyárból több, különböző típusú futószalagot is igénybe vehetünk egyszerre; ezek mind-mind más alkotóelemek elkészítéséért felelnek majd.

Amikor a tényleges szoftvertervezésre és fejlesztésre kerül sor, akkor az adott domain-modell, vagyis futószalag kiválasztása után elkészíthetjük azt az egyszerűsített modellt, ami a létrehozandó kódot szimbolizálja. Ilyen modell lesz például egy valódi osztályhierarchia osztálydiagramja a Class Designerrel elkészítve vagy éppen egy ablak felülete Form Designerrel készítve. (Bár ezek nem a DSL-t használják, de jól szemléltetik a struktúrát. Aki igazi DSL-t szeretne látni működés közben, az vessen egy pillantást a Distributed Systems Designerre!) Ez a modell messze könnyebben módosítható a fejlesztés során, mint maga a forráskód, viszont egyetlen kattintással, bármikor legenerálhatjuk belőle a számunkra szükséges kódot, amit utána már csak minimálisan kell testre szabnunk saját igényeinknek megfelelően.

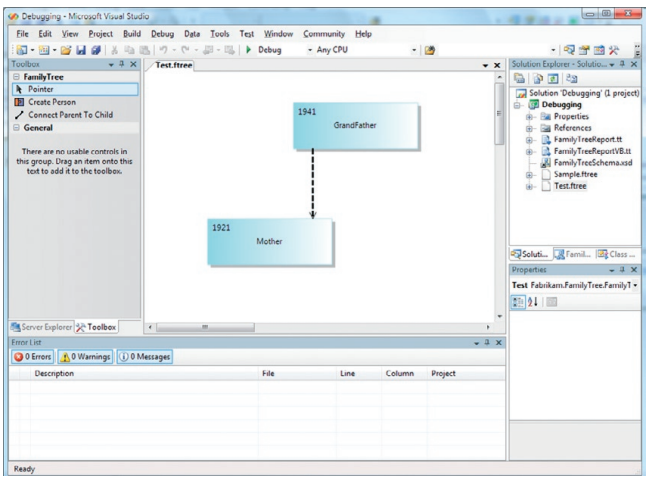
Hogyan támogatja ezt a DSL Toolkit?

A Visual Studio SDK része a Domain Specific Languages Toolkit, ami a fenti folyamatot támogatja, illetve beilleszti a Visual Studióba. A DSL Designer projekt segítségével meghatározhatjuk azt a domain-modellt, ami egyszerre definiál egy XML alapú nyelvet és egy vizuális tervezőt. A modell használata során ezt a vizuális tervezőt tudjuk majd arra használni, hogy látványosan és közérthetően készítsünk egy modellt, az XML nyelv pedig azt biztosítja, hogy az emellett egzakt és szabályos is legyen. A projekt részeként szöveges sablonokat (text template-eket) készíthetünk, amelyek a modell egyes elemeit az általunk megkívánt végeredményre fordítják le, például osztálydefiníciókat készítenek a modell tulajdonságai alapján.

Az elkészített domain-modellt mindjárt tesztelhetjük is, ekkor a Visual Studio publikálja a szükséges elemeket, és már tervezhetjük is a tényleges modellünket az általunk megadott szabályrendszer, vagyis a domain-modell felépítése szerint. (Ennél a lépésnél fontos, hogy a Visual Studio rendszergazdai jogokkal fusson, mert a domain-modell szerelvényét a GAC-ba kell publikálni.)

Konklúzió

A szoftvergyár-megközelítés és a DSL toolkit lehetővé teszi, hogy sokkal profibb és hatékonyabb legyen a fejlesztési folyamat. Saját magunk jelölhetjük ki a középutat az absztrakció és a hatékonyság között, és ehhez látványos eszközöket kapunk. Fontos még, hogy a modell megalkotásába fektetett energia nem hiábavaló, hiszen a következő iterációban a finomításnál vagy a következő projektben busásan megtérül az erőfeszítés. Akinek sikerült felkelteni az érdeklődését, az töltse le az SDK-t, és próbálja ki a helpben mellékelt oktatóanyagot, azzal sokkal kézzelfoghatóbb lesz az egész folyamat. Aki pedig további cikkeket szeretne a témában olvasni, annak javasoljuk az MSDN oldalait.



.. és tesztelés közben

Nagy Levente
(nagy.levente@microsoft.com) Microsoft Magyarország

A SHAREPOINT KERESŐ-SZOLGÁLTATÁSA

A Sharepoint-webhelyek keresői működésének és a hibakeresés, finomhangolás eszközeinek áttekintése.

Az irodai munkában manapság jellemző, hogy nagy mennyiségű, és különböző helyeken szétszórtan található információt kell folyamatosan használnia az embernek. Alapvetően fontos, hogy a tartalom valamilyen logikailag áttekinthető rendszerben legyen elrendezve. De még így is rendkívüli könnyebbséget nyújt egy olyan kereső, amelyet gyorsan segítségül lehet hívni, és egy-egy megfelelően kiválasztott kulcsszóval azonnal a kívánt tartalomra lehet lépni. A Sharepoint-családban két kereső is van, de jó tudni a különbséget a Windows Sharepoint Services v2 (WSS) és a Sharepoint Portal Server 2003 (SPS) lehetőségei között. A WSS egyszerű keresőjének rövid tárgyalása után cikkünk az SPS jóval nagyobb tudású és összetettebb indexelőjének működését vázolja fel, és igyekszik támpontokat adni a szolgáltatás mozzanatainak tetten éréséhez, ahogy arra a hibakereséshez, finomhangoláshoz szükség lehet. A cikk fejtegetéseinél egy teljesen alapértelmezett beállításokkal telepített rendszert tekintünk hivatkozási alapnak, feltételezve, hogy aki valami eltérő konfigurációt alkalmaz, az ismeri a módosításait (amelyek egyébként remélhetőleg le is vannak dokumentálva).

Windows Sharepoint Services v2

A Sharepoint rendszer alap-infrastruktúrája, a WSS az SQL Server teljesszöveg-indexeit használja. Gyakorlatilag semmit nem kell és nem lehet rajta konfigurálni: vagy be van kapcsolva, vagy nincs (Központi Felügyelet – A teljes szövegben történő keresés beállítása). Ennek a keresőnek az eredményhalmaza mindig arra a webhelyre korlátozódik, ahol a kulcsszó beírásával elindítottuk a keresést.

Ha az eredményhalmazban nem jelennek meg az elvárt elemek, azaz hibás működést sejtünk, akkor az SQL Serverbe kell ellátogatni, és első lépésként a tartalom-adatbázishoz tartozó teljesszöveg-indexen kell egy „repopulate” vagy „rebuild” műveletet végrehajtani. (Amennyiben ez nem segít, érdemes

A WSS és az SPS keresőjének összehasonlítása – 1. táblázat

Keresés célja	WSS-kereső	SPS-kereső
Listaelemek	Igen	Igen
Dokumentumok	Igen	Igen
Listák	Igen	Igen
Logikai operátorok (AND, OR, Near, NOT) használata	Nem	Igen, külön programozással (a NOT nem támogatott).
Nem .doc, .xls, .ppt, .txt, and .htm típusú fájlok	Alapértelmezésben nem, de lehet külső iFiltereket telepíteni.	Igen. Bővebben az SPS2003 „Administrator's Guide”-ban.
Alwebhelyek tartalma	Nem. A keresést az adott alwebhelyről kell indítani.	Igen
Nem szöveg típusú mezők (például pénz, szám, lookup, Igen/Nem)	Nem	Igen
Listaelemek csatolmányai	Nem	Igen
MS Office System 2003 dokumentumok tulajdonságai (Szerző, Cég)	Nem	Igen
Felmérés típusú listák	Nem	Igen
Rejtett listák	Nem („by design”)	Nem („by design”)
Külső webhelyek, megosztott mappák, dokumentumok	Nem	Igen
Keresőszóra kapott összes találat számának jelzése	Nem	Igen. Egyéni keresési lekérdezést kell hozzá használni.

megnézni a 817301 és 317746 KB-cikkeket, illetve lehet, hogy a probléma mélyebb elemzést kíván). Általában ha baj van a WSS keresőjével (azaz a tartalom-adatbázis teljesszöveg-indexével), akkor semmilyen eredményt nem kapunk. (Ilyenkor a teljesszöveg-indexek tisztába tételével megtalálhatjuk a megoldást; nagy adatbázis esetén azonban óvatosnak kell lenniük: amikor beindul az indexelés, hosszabb időre is lehúzhatja a szerver teljesítményét.) Azokon a Sharepoint-helyeken, amelyek nem teljes SQL Serverre, hanem a csak a Sharepoint-telepítőben található ingyenes MSDE-re épülnek, a WSS-lapokon nem lehet bekapcsolni a keresőt, mert az MSDE-ben nincs teljesszöveg-indexelés!

1. ábra. Így néz ki az SPS keresője

Sharepoint Portal Server 2003

A Sharepoint Portal keresője sokkal többet nyújt, mint a WSS *(1. táblázat)*; persze ezzel együtt a beállítás és a hibák elhárítása is nagyobb ráfordítást igényel. Az SPS szerves részét képező keresőmotor az egyik, ha nem a legfőbb szolgáltatás, ami a portálrendszert a WSS-től megkülönbözteti. Ezzel az indexelővel a Sharepoint-lapokon kívül egyéb tetszőleges webes források, megosztott mappák és Exchange-tárak (nyilvános mappák) tartalmát is bevonhatjuk a keresési eredmények körébe. Szabályokkal határozhatjuk meg, hogy pontosan mit veszünk bele az indexelésbe, és mit zárunk ki. A Sharepoint-tartalomban rákereshetünk konkrétan egyes metaadatokra (például a dokumentum szerzőjére, a munkatársunk tulajdonságaira stb.). Megadhatunk keresési tartományokat, amelyek az igényeink szerinti halmazra szűkítik a keresés területét, illetve némi különmunkával elérhető, hogy logikai műveleteket használhassunk a kulcsszavakkal („vele VAGY nélküle").

SPS rendszereknél felmerülhet a kérdés, hogy ha az SPS a WSS-re épül, illetve a WSS-t egészíti további szolgáltatásokkal, akkor mi lesz a WSS keresőjével egy portálon. A válasz nem kézenfekvő: sajátos módon együtt él a kettő. A (gyökérszintű) portálwebhelyen (http://portal/) és annak területein (pl. http://portal/News) a portál keresőjét érhetjük el, míg az az alá rendelt WSS webhelyeken (például http://portal/sites/webhely) a WSS keresőjét. A WSS-webhelyeket a portál területeitől egyrészt az URL alapján különböztethetjük meg (a WSS-lapok a „.../sites/..." virtuális könyvtár alatt találhatók), másrészt a keresődoboz is másként néz ki. A portál keresőjénél a kulcsszavak bevitelére szolgáló mező mellett módunkban áll kiválasztani a keresési tartományt, illetve az ezektől balra lévő nagyítóikon egy link, amely a részletes keresés lapjára mutat. Ez utóbbi két funkció a WSS keresőjénél nincs meg (nagyítóikon van, de az nem link) *(1. és 2. ábra)*. Ami mindezt igazán érdekessé teszi, az az, hogy a WSS-kereső továbbra is csak arról a webhelyről ad eredményeket, ahonnan a keresést indítjuk, míg az SPS-kereső a tetszés szerint kiválasztott tartományról, azaz akár az egész portálról annak minden területével és alsóbb (WSS) webhelyével együtt. Függetlenül tehát a WSS keresőjétől, amely továbbra is az SQL teljesszöveg-indexét használja, az SPS bejárja a portált is, az alá csatlakozó WSS-webhelyeket, és minden egyéb általunk hozzáadott tartalomforrást is! Az SPS keresőjének mindegy, hogy épp melyik portálterületen állunk, amikor megadjuk a keresőszavakat, (ha a „Minden forrás" tartományt választjuk) az összes előbb felsorolt helyről kaphatunk találatokat.

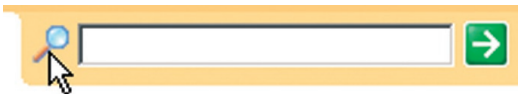
Most pedig egy kicsit tekintsünk bele a gépezet működésébe. A Sharepoint Portal keresőmotorja rokona az SQL Serverének, a közös alapból kiindulva más-más irányba fejlesztették őket. Egyforma a működésükben, hogy a bejárandó tartalomból bizonyos szűrőkkel (iFilter) kivonják a szövegtartalmat. A szavaknak megkeresik a szótövét (így a ragozástól független keresési eredményeket tudnak visszaadni). Az összetett szavakat összetevőikre bontják, valamint figyelmen kívül hagyják az érdemi jelentés nélküli szavakat („a", „az" stb.).

Az SPS indexelője két folyamatban testesül meg: az mssearch.exe-ben, amelyből mindig egy van, illetve az mssdmn.exe-ben, amelyből a tartalom bejárása alatt több is dolgozik egyszerre. Az mssearch.exe szolgáltatásként fut, és a bejárást vezényli, illetve a tiszta szöveg anyagból az indexeket létrehozza, kezeli, majd a portálra bejövő keresési lekérdezéseket kiszolgálja. Az mssdmn.exe egy-egy szála az mssearch.exe-től mindig egy konkrét tartalomelem, dokumentum feldolgozására kap megbízást, azt tiszta szöveggé alakítja, feldolgozza, és az eredmény visszaadása után lezárul.

A szövegtartalom alapján előállított indexeket az SPS a „c:\Program Files\Sharepoint Portal Server\Data\<GUID>\sps.edb" fájlban tárolja, ami nem más, mint egy JET adatbázis, akárcsak az Exchange-szerverek adattárai. A hibakeresés során ez jellemzően „fekete doboz", mert nem olyan „felhasználóbarát" a kezelése, mint egy SQL adatbázisé. Ezen kívül a legtöbb esetben, amikor okunk van feltételezni, hogy megromlott a tartalma, eldobhatjuk, és újragenerálhatjuk: ha az indexelőt csak a portál (esetleg néhány további, a vállalati hálózaton belül nagy sávszélességgel elérhető tartalomforrás) bejárására használjuk, akkor az indexek néhány óra vagy egy éjszaka alatt újragenerálhatók.

Ebből következik, hogy nagyon mély, nagy ráfordítással járó hibakeresésnek ebben az irányban nincs is értelme.

Természetesen vannak olyan példák (és a Sharepoint portál részben pont erre lett kitalálva), ahol az SPS a vállalati információgazdálkodás csomópontja, és a sok, Sharepointon kívüli forrás (archivált fájlok, Exchange: nyilvános mappák) teljes bejárása több napon át elnyúlik. Bár az ekkora indexek ritkák, természetesen ilyenkor óvatosabban kell őket kezelni. Néhány általános ellenőrzést és javítást el lehet végezni a Sharepoint telepítőlemezén található eseutil.exe-vel.



2. ábra. A WSS-ben pedig ezt láthatjuk

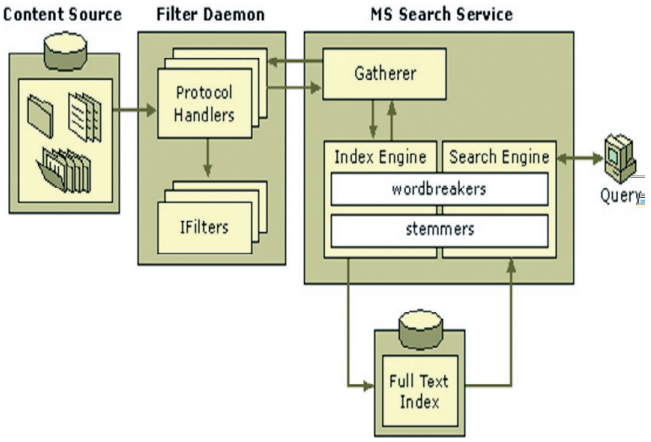
Van még egy szolgáltatás, ami az előbbiekkel ellentétben a WSS és az SPS keresőjébe egyaránt be van építve. A keresési eredmények megjelenítése előtti utolsó lépés a jogosultságok ellenőrzése („Security trimming"): a keresést indító felhasználó a Sharepoint-webhelyekről kapott találatok közül csak azokat az elemeket láthatja, amelyekhez számára engedélyezve van a hozzáférés. Ez azt jelenti,

hogy azoknak a webhelyeknek, területeknek a tartalma, amelyekhez az illető a böngészés során nem férhet hozzá, az eredményhalmazban sem jelenik meg.

Ha nem jól működik

Milyen eszközeink vannak rá, hogy belelássunk a kereső magánéletébe, ha egyáltalán nem működik, lassú, vagy nem hozza a kívánt eredményeket?

Mindig az eseménynaplóban kezdjük a kutatást; a hiba kódján és szövegén (ezekre érdemes rákeresni az interneten) kívül az üzenetek sok esetben támpontot adnak teendőkkel kapcsolatban is. Az SPS képes részletesen naplózni működésének lépéseit a „c:\Program Files\Sharepoint Portal Server\Logs" könyvtárban lévő állományokba. A részletesség mértékét a „Központi felügyelet – Diagnosztikai beállítások" lapról kiindulva tudjuk megváltoztatni; a kereső körüli hibákról értékes információkhoz juthatunk, ha bővebbre állítjuk a „Search Service" és az „Administrative Service" naplózását. A hibanaplókban az „exception", „error" és „fail" szavak környezetében találhatunk adatokat a rendellenességekkel kapcsolatban.



A SharePoint Portal Server keresőjének felépítése

Amikor az indexelés nagyon lassan halad vagy nem is fejeződik be, illetve túlzottan erőforrás-igényes, arra vagyunk kíváncsiak, hogy milyen tartalmat dolgoz éppen fel, amikor így viselkedik, és hol van a rendszerben a szűk keresztmetszet. Az előbbihez a Tartalomgyűjtő napló (Gathererlog) nyújt hatékony segítséget: ebben a „Webhely beállításai – Keresés és indexelés konfigurálása – Hibák és figyelmeztetések megjelenítése: Portáltartalom/külső tartalom" lapon elérhető logban másodpercről másodpercre követhetjük, hogy milyen hiba adódott a tartalom egyes elemeinek feldolgozásakor. Szoktak lenni olyan bejegyzések is, amelyekből nem olvasható ki konkrét fájlnév. Ezekből kiemelhetjük a „listid" és az „itemid" (= doclibrowid) paramétereket, és az alábbi lekérdezésbe behelyettesítve őket, a portál tartalom-adatbázisából (<portálnév>_SITE) megtudhatjuk, hogy miről van szó:

```
select dirname, leafname, size, metainfosize from docs
where listid='3f574d9b-4d11-49e2-9ea2-a125a36a-
ba4f' and doclibrowid=186
```

Amikor például külső gyártó által készített iFiltert használunk (pdf, zip, vektorgrafikus vagy egyéb fajta fájlokhoz), és az nem tudja megfelelően kezelni a hozzá tartozó állományokat, innen világosan kiderül a hiba.

A Tartalomgyűjtő naplónál van mód arra is, hogy ne csak a hibákat jegyezze be, hanem a sikeresen feldolgozott vagy szándékosan kizárt elemek bejárását is említse (Webhely beállításai – Keresés és indexelés konfigurálása – Naplófájl beállításai). A hibakereséshez nagyon hasznos lehet ez a szolgáltatás, de ügyeljünk rá, hogy amikor már nincs rá szükségünk, akkor legyen kikapcsolva, különben veszélyesen nagy mennyiségű adat halmozódhat fel a <portálnév>_SERV adatbázisban.

Ha elveszett erőforrásaink után nyomozunk, akkor sokat segít a Teljesítménymonitor. Érdemes egyrészt általános számlálók beállításával meggyőződni róla, hogy a gépen a processzor, memória és merevlemez mennyire van igénybe véve *(2. táblázat)*; másrészt, ha minden folyamatnak figyeljük a memória- és processzorfogyasztását (%Processor Time; Working set), könnyű észrevenni, hogy melyeknek vannak túlzott igényeik. Végül, de nem utolsósorban vegyük észre, hogy milyen sokféle számlálóval rendelkezik maga a Sharepoint-kereső.

Némi kísérletezéssel és intuitív használattal ez utóbbiak jól kiegészítik a rálátásunkat a rendszerre.

Azokban az esetekben, amikor a kereső működik, de nem adja vissza az elvárt eredményhalmazt, a Tartalomgyűjtő napló a legfontosabb forrás, illetve azt kell megfigyelni, hogy a hiányzó találatoknak milyen közös tulajdonságaik vannak. Egyes weblapokról, tartalomforrásokból nincsenek eredmények? Valamilyen dokumentumok teljesen hiányoznak?

Hiányos eredményhalmaznál az egyik leggyakoribb ok az, hogy nincs a tartalomhozzáférési fióknak (Központi felügyelet – Kiszolgálófarm fiókbeállításainak konfigurálása) elég jogosultsága: ellenőrizni kell, hogy az említett fiók tagja-e az SPS-felügyeleti csoportnak (Központi felügyelet – Sharepoint felügyeleti csoportfiók beállítása), illetve az ő nevében megnyitott (Futtatás mint...) böngészőből el lehet-e érni az adott tartalmat.

Általános teljesítménymutatók – 2. táblázat

Objektum	Számláló
Processor	Percent processor time_total
Network interface	Bytes total per second_network interface
Logical disk	Percent idle time
Paging file	Percent usage
Memory	Page faults per second
System	Processor queue length
ASP.NET applications	Requests per second_total
Web service	Get requests per second_total

Ezzel be is fejeződik a diagnosztikai körutazás, sikerült áttekinteni az eszköztárat, ami minden Sharepoint-tulajdonosnak rendelkezésére áll. Talán érdemes ezek megismerésére némi időt fordítani még a „boldog békeidőkben" is, mert általuk betekintést nyerhetünk a kereső működésébe, és felkészültebben állhatunk neki a szerver beállítások finomhangolásának vagy az esetleges bajok leküzdésének. Azt pedig soha nem árt hangsúlyozni, hogy a szervizcsomagok (tesztrendszeren történő sikeres próbák utáni) telepítésével sok felesleges fejfájást takaríthatunk meg.

Pölös Máté
(matepol@microsoft.com) Microsoft Magyarország

NEM SZABAD LEMARADNI

Inkább lendületbe jövünk!

Az előző TechNet Magazin igencsak pozitív fogadtatása után újult erővel láttunk hozzá a legfrissebb szám elkészítéséhez. Most nem egy konkrét szoftverre fókuszáltunk címlapsztoriként, hanem az informatikai infrastruktúra legáltalánosabb tervezési és implementációs kérdéseire koncentráltunk; megnéztük, egyáltalán milyen építőelemek állnak a rendelkezésünkre. Mindezt olyan érdekességek bemutatásával fűszereztük meg, amelyek egészen újszerű, korábban igen nehezen megvalósítható megoldásokat is elérhető közelségbe hoznak.

Ezt követően lényegesen mélyebbre ásunk a mostani áttekintő cikkek folytatásaként. Elsőként az Exchange 2007-nek szentelünk csaknem egy teljes lapszámot, majd a System Center termékcsaládot vesézzük ki. Mire pedig végzünk ezekkel, már az ajtón kopogtat a „Longhorn" Server is, amelyet szintén igyekszünk a megfelelő alapossággal körbejárni.

A TechNet-modulok

Mivel rengeteg az új, megismerésre váró technológia, ezért a TechNet programot is átalakítottuk, hogy könnyebben, kevesebb idő és energia ráfordításával lehessen megismerni a most érkező szoftvereket. A megszerezhető tudást igyekeztünk időben és tartalmilag is rendszerezni, így születtek meg a TechNet-modulok.

A modulok a webcastokat (online előadásokat), szemináriumokat, és a TechNet Magazinokat fogják össze egy egységes információfolyammá, így hetente pár órát kell csak rászánni arra, hogy mindenki maga ismerhesse meg a legújabb megoldásokat.

MVP-k Kelet-Európából

Ebben a félévben két modult indítottunk útjára: az egyik a Windows Vista és a 2007-es Office képességeit tekinti át, a másik pedig az IT-rendszerek infrastrukturális kérdéseivel foglalkozik mélyebben.

Némi extra motivációt biztosítva – és a tanulásra szánt időt honorálva – meglehetősen értékes nyereményeket sorsolunk ki azok között, akik a legjobb eredménnyel végeznek a modulokat záró, január végi TechNet-kvízeken. Ezeken bárki részt vehet, és tesztelheti, mennyit sikerült elsajátítania az eltelt félév alatt a modulok tematikájából.

Az első helyezettek egy-, illetve másfél millió forint értékben jutnak hozzá a Számalk és a NetAcademia által felajánlott rendszermérnöki képzéscsomagokhoz, de a további helyezettek is értékes könyvcsomagokat, illetve dobozos Windows Vista Ultimate és Office 2007 Professional szoftvereket nyerhetnek.

A TechNet-modulokról minden információ közvetlenül elérhető a TechNet Portálról, a www.microsoft.hu/technet címen, beleértve a már lezajlott valamennyi esemény és webcast felvételeit és prezentációit is.

A legkiemelkedőbb szakemberek

Szeptember végén került sor arra a találkozóra, amelyik a kelet-európai MVP-ket (Most Valuable Professional) hozta össze, hogy közösen képezzék magukat tovább a Microsoft előadóinak segítségével, és egyúttal jobban megismerjék egymást. Az MVP-k idei nyílt napjának Moszkva adott otthont, és Magyarországot is képviselte két kiváló MVP: *Lénárd Gábor* és *Petrényi József*.

A nemzetközi és hazai nyílt napokon a Microsoft igyekszik minden segítséget megadni, hogy az MVP-k a lehető leghamarabb juthassanak információkhoz az újdonságokkal kapcsolatban. Emellett akár közvetlenül a szoftvereket fejlesztő mérnökökkel is megoszthatják véleményüket, illetve kikérhetik segítségüket napi problémáik megoldásához. Az MVP-k lehetőséget kapnak arra is, hogy hozzáférjenek és tanulmányozhassák a számukra fontos Microsoft-szoftverek forráskódját.

Nyakunkon a Vista!

Végezetül pedig felhívjuk mindenki figyelmét: a legnagyobb valószínűség szerint a Windows Vista, az Office 2007, továbbá az Exchange Server 2007 végleges változata már október végére elkészül, és ezt erősen alátámasztja, hogy már a Magazin készítésekor elérhető a Vista RC2-es verziója is. Lehet, hogy mire a lap az olvasók kezébe kerül, már lezárul az a hosszú és küzdelmes fejlesztési folyamat, ami végül a Windows és az Office új generációját eredményezi.

Ez lesz az a pont, amikor a legtöbb szakember elsőként próbálja majd ki ezeket az új szoftvereket, és elkezdenek gondolkodni az új képességek felhasználási lehetőségein és a bevezetés lépésein. Reméljük, hogy ebben a nem egyszerű feladatban továbbra is hasznos segítőtársak lehetünk a TechNettel!

Budai Péter
(i-pbudai@microsoft.com) Microsoft Magyarország

VEZETŐKÉPZŐ - A NETACADEMIA ÚJ TANFOLYAMSOROZATA

VEZETŐKÉPZŐ

Híd az üzlet és az informatika között

Ha

- fontos Önnek az informatikai biztonság,
- nyomasztja a szabályozatlanság,
- el akarja kerülni az adatvesztéseket, IT katasztrófákat,
- tudni szeretné mi az ITIL, a BS 7799, az ISO, a COBIT,
- meg akarja óvni a cégadatokat,
- ismerős Önnek a „Kulcsember” fogalma.

Akkor a megoldás:
NetAcademia Vezetőképző!

Tanfolyamsorozatunkon újfajta rálátást nyújtunk szakmájának legkritikusabb területeire!

Jelentkezzen most a tanfolyamsorozatunkra!

További információk:
HTTPS://WWW.NETACADEMIA.NET/TRAINING.ASPX?CTYPE=66

IDÉN FIZET, JÖVŐRE KAP!

Vásárolja meg idei áron a jövő év szolgáltatásait!

Ne hagyja elveszni 2006-os oktatási keretét csak azért, mert az év végi hajrá miatt nincs ideje a képzésekre!

Biztosítsa be magát az „Idén fizet, jövőre kap!” akciónk keretében.

A NetAcademia Oktatóközpont közel 100 tanfolyama, 700 vizsgája és rendszerintegrátori szolgáltatása most még idei áron érhető el.

Ne szalassza el a lehetőséget!

Rendelje meg már most a képzéseket, vagy egyeztessen munkatársainkkal!

További információk:
HTTPS://WWW.NETACADEMIA.NET/ACTION.ASPX

netACADEMIA
A LEGJOBBAKAT TANÍTJUK.

CÍM: 1062 BUDAPEST, ANDRÁSSY ÚT 62.

TELEFON: (06 1) 472-1214
IRODAI MOBIL: (06 20) 369-6947
FAX: (06 1) 472-1215
INTERNET: WWW.NETACADEMIA.NET
E-MAIL: INFO@NETACADEMIA.NET

Egy értéktőzsde, amely naponta 300 millió tranzakciót dolgoz fel. Mégpedig Microsoft SQL Server 2005-tel.

A NASDAQ, az USA legnagyobb elektronikus értéktőzsdéje, a világ 37 országának részvényeivel kereskedik. Létfontosságú kereskedői és üzenetközvetítő rendszere az SQL Server 2005-re épül, hogy képes legyen másodpercenként akár 64 000 tranzakció kiszolgálására is – mindezt 99,999%-os rendelkezésre állással.* Nézze meg, hogyan! www.microsoft.com/bigdata

*Ezek az eredmények nem általános érvényűek, és a Windows Server™ 2003 Enterprise Edition környezetben születtek. A rendelkezésre állás számtalan tényezőtől függ, beleértve a hardver- és szoftver-megoldásokat, folyamatokat, támogatási szolgáltatásokat is.