# tech.net

WORKING WITH WINDOWS

III. évfolyam
10. szám

Ára: 1344 Ft

A TARTALOMBÓL





ISSN 15865185

9 771586 518005 10

# NetAcademia Mikulás-konferencia
## A biztonság jegyében

**2002. december 6. péntek 10:00-18:00, Andrássy Konferenciaközpont**

**Előadások:**

**Fóti Marcell: Az Active Directory biztonsága**

**Fülöp Miklós: Általános biztonsági teendők a Windows hálózatban**

**Dorner Csilla: Exchange KMS, biztonságos Exchange levelezés**

**Soczó Zsolt:Programozott Security a .NET keretrendszerben**

**Gyakorlatok:**

- Active Directory Authoritative Restore
- IPSec alagút két tartományvezérlő között
- S/MIME levelezés beindítása Exchange környezetben
- Futtatási jogosultságok vezérlése

**Részvételi díj: cégenként az első jelentkező 25.000,-Ft+ÁFA, minden további résztvevő 5.000,-Ft+ÁFA-ért vehet részt.**

**A részvételi díj tartalmazza az ebéd költségét.**

**Helyszín: Budapest 1062, Andrássy út 62.**

**Jelentkezés: http://www.netacademia.net/mikulas**

**Információ: a 06/1/472-1214**-es telefonszámon vagy az **info@netacademia.net** e-mail címen.

# Minőségi ugrás: a Tablet PC



Szerkesztőség
Főszerkesztő: Fóti Marcell
marcellf@netacademia.net
Főszerkesztő-helyettes: Fülöp Miklós
mick@netacademia.net
A szerkesztőség címe:
1062 Budapest, Andrássy út 62.
Tel.: 472-1214
technet@netacademia.net
Nyilvános levelezési lista:
tech.net@technetklub.hu

Kiadja és terjeszti a
NetAcademia Kft.
Terjesztési, előfizetési információ:
Tel.: 472-1214
terjesztes@netacademia.net
Megjelenik havonta, ára 1.344 Ft



Előfizethető megrendelőlevélben a szerkesztőségnél:
1062 Budapest, Andrássy út 62.
Fax.: 472-1215
http://technet.netacademia.net/subs

Hirdetésfelvétel: Szívós Éva
Tel.: 472-1214
Fax.: 472-1215
info@netacademia.net
Grafikai tervezés, kivitelezés:
Gregor László
Nyomdai előkészítés:
NetAcademia Kft.

Nyomda:
Hieron Kft.
2120 Dunakeszi, Tamás A. u. 11/a
Felelős vezető: Török Andrea

ISSN 1586-5185

Az elmúlt hetekben több vidéki városban is felbukkantam a Microsoft szokásos évi előadássorozatának szereplőjeként. Ha már ott voltam, nézőként is „hasznossá tettem magam", vagyis odafigyelve végighallgattam a többi előadást. Az utolsó előadás, melyet az Egyik Nagy Processzorgyártó képviselői abszolváltak, különös érzéseket keltett bennem. Úgy éreztem, ez a nagy cég fél valamitől. Nem a konkurenciától. A recessziótól. De attól nagyon. És ebbéli félelmükben nem átallottak hatalmasabbnál hatalmasabb lózungokat megereszteni a közönségnek, akik szerintem szintén kihallották azt, amit én.

Van-e valaki ebben a szakmában, aki igazat adna annak az érvelésnek, hogy a dolgozók termelékenységét növeli, ha 800 MHz-es asztali gépeiket 1,6 GHz-esre cserélik? Jól hallották, a termelékenységük növekszik. Egy titkárnőé? Ettől?

Mondok én a processzorgyártóknak valamit. Innen, Magyarországról. A mennyiségi növekedés erőltetése helyett valami minőségi ugrást kellene produkálni már. De megfizethető áron. Az árak miatt a 64 bites világ sajnos nincs elég közel hozzánk.
Okosan hangzik, nemde? Egy nagy baj van csupán: a színpadon a Windows áll, az ő produkciójával találkozik a felhasználó. A processzor valahol a háttérben, a zsinórpadláson dolgozik, és rendezgeti a szálakat. A háttérből kellene a nézőtérre kiható, a publikumot felkavaró dolgot produkálnia. Nehéz eset, nem mondom! Erre csak egy esélyt látok: ha előadás közben leengedik a függönyt ;).

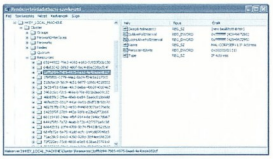
A színpadon azonban most is történik olyan minőségi változás, ami a zsinórpadlás legénységének is hasznára fog válni: jön a Tablet PC!
Ezzel az eszközzel az informatika további helyekre hatol be, és – hál'Istennek - megnövekedett processzorteljesítményt igényel. Amit egyszer Bill elgondolt, az úgy lesz. Information at your fingertip. Anytime, anywhere computing. Én úgy saccolom, a Tablet PC lendületes terjeszkedésbe fog kezdeni. Évek óta van ugyanis egy-két olyan terület, amely hiába várta informatikai ellátottságát:

- Az egyik ilyen terület a monitoron történő olvasás. Ma hiába szeretném a tech.net magazint webre vinni, ez egyenlő lenne az olvasótábor megszűnésével, mert négyoldalas cikkeink legalább 16 képernyőn terjeszkedve olvasási rémálommá válnak. Valódi híveink bizonyára kinyomtatnák a cikkeket. A Tablet PC ClearType technológiája segítségével talán végre képesek leszünk monitoron elolvasni a dokumentumokat.
- Szintén ellátatlan terület a webes mikrofizetés. Webes tartalomért a mai napig nem lehet kérni egy vasat sem. Évek óta várják a tartalom-előállítók, hogy a neten át is érvényesíthessék pénzkereseti igényüket. Most majd kapunk Digital Rights Managementet, talán még digitális pénzt is *(lásd ehavi számunk idevágó cikkét a jogi rovatban).*
- Jegyzetelésre még mindig legjobb a papír és ceruza. Ezt kellene leutánozni. A Tablet PC megteszi.

Persze van, ami még most sem lehet a mienk. Mi a helyzet a magyar beszéd és a magyar kézírás felismerésével? Erre azt hiszem egy keveset várnunk kell még.
A hardvergyártók mindenesetre megnyugodhatnak: pár év múlva már csak a minimum 10 GHz-es processzorok lesznek eladhatók. S gépenként minimum négy kell majd belőlük.

Fóti Marcell
marcellf@netacademia.net

# 2002. 10. szám



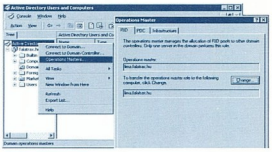
## Farkasokkal táncoló XI.

### *Cluster a gyakorlatban* 4.oldal

A fürtszolgáltatással ismerkedők számára kezdetben nagyon kellemes, hogy a fürt „mindent elintéz". Főleg, ha esetleg a rendszert készen kaptuk, vagyis a szolgáltatások már működtek, amikor az eszköz a kezünkben került. Aztán jön az első saját telepítés, az első erőforráshiba, és a rendszergazdának meg kell tanulnia alámerülni a fürtszolgáltatás mélységeibe.

## Active Directory: FSMO szerepek 8. oldal

Az előző részben végigvettük a tartománytelepítés lépéseit. Most megvizsgáljuk, milyen kiszolgálószerepek születtek. A hagyományos tartományvezérlői szerepeken túl ugyanis lett PDC Emulator, Schema Master, Domain Naming Master, RID Master és Infrastructure Master szerepkör is. Ezeket összefoglaló néven FSMO *(Floating Single Master Operations)* szerepeknek nevezzük.




## Sémamódosítás plusz adatmódosítás 10. oldal

Az Active Directory nemcsak „gyári" adatok tárolására képes, hanem a séma bővítésével tetszőleges további attribútumok adhatók a meglévő objektumokhoz. Felvehető a címtárba például a felhasználók személyi száma, a számítógépek gyártási száma, a felhasználói csoportok gazdasági adatai stb.

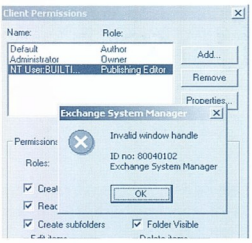
## VPN - Virtuális magánhálózatok

### *2. rész, Az L2TP protokoll* 14. oldal

Az előző részben körüljártuk a VPN hálózatok „lényegét", Windows 2000 kiszolgálónkon varázsoltunk egy VPN kiszolgálót és csatlakoztunk is hozzá – egyelőre csak egy munkaállomással, PPTP kapcsolaton keresztül. Ma terítékre kerül az új protokoll, az L2TP is.


## Digital Money 19. oldal

Az elektronikus kereskedelemről és az elektronikus aláírásról szóló törvények megteremtették Magyarországon is a jogszabályi hátterét annak, hogy az Interneten keresztül áruk, illetőleg szolgáltatások cseréljenek egymással gazdát, vagyis végre meginduljon az elektronikus kereskedelem.



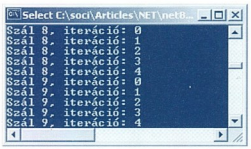

## Microsoft Exchange 2000 Nyilvános mappák 20. oldal

Az Exchange 2000-ben a nyilvános mappák területén fontos változtatások történtek. Az Exchange és az Active Directory összefonódásaként a jogosultságok kiosztása, és használata módosult. Ahogy az adatok elérése is sokféle módon történhet, a jogosultságok állítása is bonyolultabb. Miután megismerkedtünk kicsit a nyilvános mappákkal, belevetjük magunkat a jogosultságok dzsungelébe.

## XMLgessünk 16.

### *SQL XML 3.0 - I. rész* 24. oldal


Az SQL Server 2000 számos beépített XML- szolgáltatással rendelkezik, viszont a termék megjelenése óta több mint három év telt el, és az XML- technológiák pont az utóbbi időben indultak szédületes fejlődésnek. Az SQL Server emiatt újabb és újabb XML- szolgáltatásokat kapott, melyeket külön csomagban lehet letölteni. Ez az SQLXML programcsomag.



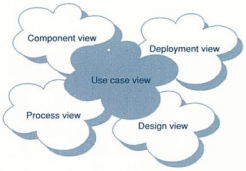

## .NET Akadémia VIII.

### *Többszálú és aszinkron programozás* 28. oldal

A többszálú programozás az egyik legtipikusabb felhasználói módú programozási terület, ahová csak a legelszántabb C++ programozók merészkedtek el. Windows API használatával időnként tényleg nehéz helyzetben találjuk magunkat, de a Common Language Runtime nagyon leegyszerűsíti az életünket, ha menedzselt környezetben kell többszálú programokat fejleszteni.

## Az UML – *1. rész* 34. oldal

A modellezés témát folytatva, pontosabban részletezve UML sorozatom első részében a módszer kialakulásának körülményeit szeretném bemutatni, illetve néhány olyan alapfogalmat, amelyek megértése elengedhetetlen a későbbiek megértéséhez.

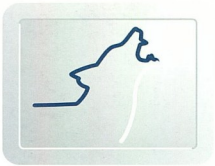



## OfficeXP Resource 39. oldal

Akik kicsit jobban belemélyedtek az Office lelkivilágába, azok már jól ismerik az Office Resource Kit eszközök nyújtotta lehetőségeket. Most egy újabb verzióját tölthetjük le ennek a csomagnak.

## Orregér 40. oldal

Azok, akik hosszú órákat töltenek a számítógép előtt, tudják mit jelent egy nem éppen kifogástalanul működő egér. A mikrokapcsolók – inkább előbb, mint utóbb – már csak eredménytelenül kattognak, a golyóra minden szösz ráragad, a görgőkre kiválóan tekeredik fel a leghosszabb hajszál, és az egéralátét vagy a kábel éppúgy véget ér, mint a türelmünk.



# Farkasokkal táncoló

## *(XI. rész) - Cluster a gyakorlatban*



A fürtszolgáltatással ismerkedők számára kezdetben nagyon kellemes, hogy a fürt „mindent elintéz". Főleg, ha esetleg a rendszert készen kaptuk, vagyis a szolgáltatások már működtek, amikor az eszköz a kezünkben került. Aztán jön az első saját telepítés, az első erőforráshiba, és a rendszergazdának meg kell tanulnia alámerülni a fürtszolgáltatás mélységeibe.

Ezt a mélységet a cluster.log jelenti. Érteni a naplót annyit tesz, tudni, mi történik a háttérben. A háttér pedig feltárja a hibákat, amelyeket kergetünk hét határon át. Ez alkalommal a fürt belső eseménynaplóját elemezzük példákon, helyzeteken keresztül.

### Munkamódszer

A cluster.log elemzésekor konkrét szituációkon keresztül vizsgáljuk meg, hogy mi kerül a naplókba. Terjedelmi okokból nem tudjuk elemezni a bejegyzéseket sorról sorra, ezért a már megismert, ismétlődő, és könnyen értelmezhető eseményeket az írás közben csak említem, de nem idézem. A cikk végén található hivatkozásokról letölthetők a teljes naplók, hogy akik szeretnék még alaposabban boncolgatni a helyzet szülte bejegyzéseket, azok megtehessék. *(Bevallom, e nélkül eléggé bonyodalmas olvasmány lesz a cikk.)* A fürt mindkét állomásának naplóit érdemes végigböngészni, így láthatóvá válik, miként érzékeli ugyanazt az eseményt az egyik, illetve a másik node. A felesleges információk elkerülése érdekében kihagyom a naplóbejegyzések elején található processz- és szálazonosítókat, valamint az időbélyeget is, hacsak nincs különös jelentőségük. A naplók letölthető változatában ezek természetesen jelen vannak. Végezetül feltételezem, hogy az előző hónapban tisztázott fogalmakat az Olvasó elsajátította.

### Egy állomás indulása

A napló egy olyan helyzetben készült, amikor a két állomásból álló fürt első kiszolgálója indult el, a másik még áll. A teljes napló az **[1]** címről tölthető le.

Az első két sort már ismerjük, a naplóbejegyzés a fürtszolgáltatás indulását jelzi. Ezután az esemény-feldolgozó komponens [EP] indul, hogy rögtön elkezdhesse a bejegyzések rögzítését.

```
[EP] Initialization...
[DM]: Initialization
```

Az esemény-feldolgozót az adatbáziskezelő szál követi. Tulajdonképpen minden szálat inicializálnia kell a fürtszolgáltatásnak, a naplóból láthatjuk, hogy ezt meg is teszi.

```
[DM]: Loading cluster database from
C:\WINNT\cluster\CLUSDB
[DM] DmpStartFlusher: Entry
[DM] DmpStartFlusher: thread created
```

Az adatbáziskezelő első dolga, hogy beolvassa az adatbázist. Az előző részben már tárgyaltuk a szolgáltatás indulását, és tudjuk, hogy a fürt ebben az állapotban még nem tudja, hol van a Quorum erőforrás, így a %systemroot%\cluster mappában található clusdb állományt olvassa be, amely a quorumerőforráson található adatbázis egy másolata. Nem biztos, hogy tökéletes másolat, arra azonban elegendő, hogy a szolgáltatás információkat szerezzen a quorum hollétéről és az erőforrásokról. *(Igazság szerint még ezek az információk is elavultak, de létezik egy mechanizmus, amely az ilyen komplikációkat kiküszöböli.)*

A többszálúság előnyeit kihasználva azonnal indul a többi szál is, nem várva meg más szálak munkájának befejezését. Előbb a node manager, aztán a failover manager, az API és a log manager komponensek inicializálódnak. A node manager azonnal munkához lát, és a beolvasott adatbázisból azonosítja az állomást és annak számát.

```
[NM] Local node name = MALSERVER3.
[NM] Local node ID = 1.
[NM] Creating object for node 1 (MALSERVER3)
```

Egy külön komponenshez nem köthető szál megállapítja, hogy milyen fiókkal indult a szolgáltatás, majd az adatbázisból kiolvasott fürtnevet felhasználva megpróbál csatlakozni a másik állomáshoz. A szolgáltatás azt feltételezi, hogy a másik állomás – és így a fürt maga – működik.

```
[CS] Service Domain Account = MAL\clustuser
[CS] Initializing RPC server.
[INIT] Attempting to join cluster MAL-CORPSERV3
```

A napló elemzése közben többször is feltűnik majd, hogy a szoftver minden része „ideges" és „siet", vagyis a feladatát minden lehetséges módon és minél előbb szeretné elvégezni. A következő sorokból látszik, hogy a másik állomáshoz való csatlakozást a fürtszolgáltatás a létező összes hálózati csatoló nevével és IP címével megpróbálja.

```
[INIT] Attempting to join cluster MAL-CORPSERV3
[JOIN] Spawning thread to connect to sponsor
 192.168.112.1
[JOIN] Asking 192.168.112.1 to sponsor us.
[JOIN] Spawning thread to connect to sponsor
 10.0.0.21
[JOIN] Spawning thread to connect to sponsor
 MALSERVER4
[JOIN] Asking 10.0.0.21 to sponsor us.
[JOIN] Asking MALSERVER4 to sponsor us.
```

Érdemes megfigyelni a rendszer állapotváltozásait. Előbb INIT állapotba kerül a szolgáltatás, amit a JOIN követ. Ez utóbbi mindaddig fennáll, amíg a csatlakozási folyamat sikeresen vagy sikertelenül be nem fejeződik.

A leírt szituációból tudjuk, hogy ezúttal a csatlakozás nem sikerülhet, a következő bejegyzés tehát nem váratlan:

```
2002/07/19-19:38:35.654 [JOIN] Sponsor 10.0.0.21
is not available (JoinVersion), status=1753
```

A sikertelenség oka a státuskódban rejlik. A korábban ismertetett módszerrel pontosabb információhoz juthatunk.

```
C:\>net helpmsg 1753
There are no more endpoints available from the
 endpoint mapper.
```

Ez az utasítás még sokszor a segítségünkre lesz, tartsuk szárazon a parancssorunkat.
A próbálkozás egy idő után abbamarad. Figyeljük meg, hogy az utolsó próba épp harminc másodpercig tart.

```
19:38:36.654 [JOIN] Waiting for all connect
 threads to terminate.
19:39:08.669 [JOIN] Sponsor 10.0.0.26 is not
 available (JoinVersion), status=1722.
19:39:08.669 [JOIN] JoinVersion data for sponsor
 10.0.0.26 is invalid, status 1722.
19:39:08.669 [JOIN] All connect threads have
 terminated.
19:39:08.669 [JOIN] Unable to connect to any
 sponsor node.
19:39:08.669 [INIT] Failed to join cluster,
 status 53
19:39:08.669 [INIT] Attempting to form cluster
 MAL-CORPSERV3
```

Az 53-as kód azt jelzi, hogy a hálózaton keresztül a másik állomás nem elérhető. A szolgáltatás visszakerül INIT állapotba, és megpróbál egyedül fürtszolgáltatást kialakítani. Vessünk egy pillantást az időpontra: 19:39:08.669!
A napló végén láthatjuk majd, hogy az üzemszerű működés eléréséhez csupán 12 másodpercre lesz szükség. Az állomás a korábban beolvasott adatbázis alapján felépíti a fürtöt, megkeresi a quorum logot, és a chekpointállományokat, azok segítségével pedig helyreállítja a fürt legutolsó állapotát. Ezt a folyamatot követjük végig.

### A fürt kialakítása a helyi adatbázissal

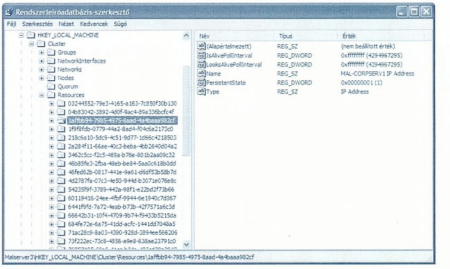
Az első lépés, amelyet meg kell tennie a clusternek, az erőforrás-ellenőr *(resrcmon.exe)* indítása. Az előző cikkből tudjuk, hogy a fürtszolgáltatás az ellenőr segítségével tartja a kapcsolatot az erőforrásokkal. Mivel a végső feladat az erőforrások elindítása, a műveletet meg kell előznie az ellenőr üzembehelyezésének.
A helyi adatbázis – ahogy az már korábban kiderült – nem más, mint egy ág a regisztrációs adatbázisban. A következő bejegyzések ezt jól mutatják. A failover manager [FM] sorra az erőforrásokat reprezentáló regisztrációs értékeket olvas be, majd inicializálja azokat.

```
19:39:08.669 [API] Online read only
[RM] Main: Initializing.
[FM] Creating group 9b26a6cb-a791-
 4807-9c17-bfc83050cd13
[FM] Initializing group 9b26a6cb-a791-4807-9c17-
 bfc83050cd13 from the registry.
[FM] Name for Group 9b26a6cb-a791-4807-9c17-
 bfc83050cd13 is 'MAL-CORPSERV3'.
[FM] Group 9b26a6cb-a791-4807-9c17-bfc83050cd13
 preferred owner 1.
[FM] Group 9b26a6cb-a791-4807-9c17-bfc83050cd13
 contains Resource 54235f9f-3789-442a-98f1-
 e22bd2f73b66.
[FM] Creating resource 54235f9f-3789-442a-98f1-
 e22bd2f73b66
[FM] Initializing resource 54235f9f-3789-442a-
 98f1-e22bd2f73b66 from the registry.
[FM] Name for Resource 54235f9f-3789-442a-98f1-
 e22bd2f73b66 is 'Cluster IP Address'.
 FM] FmpAddPossibleEntry: adding node 1
 as possible host for resource
 54235f9f-3789-442a-98f1-e22bd2f73b66.
[FM] FmpQueryResTypeInfo: Calling
 FmpAddPossibleNodeToList for restype
 IP Address
[FM] FmpAddPossibleNodeToList: adding
 node 1 to resource type's possible
 node list
[FM] FmpAddPossibleNodeToList:Warning,
 node 2 not found
[FM] All dependencies for resource
 54235f9f-3789-442a-98f1-e22bd2f73b66
 created.
```

A koreográfia felismerhető:

1. Az FM létrehozza a memóriában az erőforráscsoportot.
2. Beolvassa a csoporthoz tartozó regisztrációs értékeket.
3. Megállapítja az erőforráscsoport nevét.
4. Meghatározza, hogy a csoportot melyik állomás fogadhatja.
5. Megvizsgálja, hogy a csoport milyen erőforrásokat tartalmaz, és sorra veszi azokat is.
   a. Létrehozza az erőforrást a memóriában.
   b. Beolvassa a regisztrációs értékeket.
   c. Megállapítja az erőforrás nevét.
   d. Meghatározza a lehetséges állomásokat, amelyek birtokolhatják az erőforrást.
   e. Meghatározza az erőforrás függőségeit.
   f. Megnyit minden erőforrást, amelytől az aktuális erőforrás függ, vagy ellenőrzi, hogy már megnyitotta-e azokat.

A látottak alapján néhány fontos felismerést tehetünk. A fürt minden egyes erőforrást egy egyedi azonosítóval jelöl, egy meglehetősen „ronda" jelsorral, mint például 9b26a6cb-a791-4807-9c17-bfc83050cd13. Ezt a jelet GUID-nak hívják, ami a „globális, egyedi azonosító kód" angol rövidítése. A GUID-ok meglehetősen nehézkessé teszik a napló olvasását, mivel hosszúak és nehezen azonosíthatók. Segítségünkre lehet, ha a regedit *(regedt32)* program segítségével előre feljegyezzük az erőforrások nevét és azonosítóját. Ha a fürtnapló másolatával dolgozunk, érdemes írni egy rövid scriptet, amely sorra kicseréli a GUID-okat az erőforrások neveivel.



*A regedit.exe segíthet az erőforrások azonosításában*

Az erőforrások beolvasása mindaddig tart, amíg az FM meg nem találja a quorumot.

```
19:39:08.778 [FMX] Found the quorum resource
 f21ba2a4-62dd-4883-bf69-aa6c69f96320
```

Az FMX a failover manager tranzakciós részét jelöli.
Miután az erőforrás-kiértékelés befejeződött, az FM megkezdi a quorum „birtokba vételét". Az erőforrásellenőr felveszi a quorumot a felügyelt erőforrások listájába. Az FmpRm pontos feloldására nem találtam hiteles információt, de könnyű azt

feltételezni, hogy a „Failover Manager process Resource Monitor" helyett használja a fürt.

```
19:39:08.778 [FM]
 FmpRmCreateResource: creating
 resource f21ba2a4-62dd-4883-bf69-
 aa6c69f96320 in shared resource monitor
19:39:08.857 Physical Disk: PnP widow created
 successfully.
[FM] FmpRmCreateResource: created resource
 f21ba2a4-62dd-4883-bf69-aa6c69f96320, resid 651760
[MM] MmSetQuorumOwner(1,1), old owner 0.
```

A következő bejegyzés azért fontos, mert ez az első, amely nem a fürtkomponenstől, hanem az erőforrás-ellenőrtől származik: a Windows 2000 felkészül egy lemez fogadására.
A sorban olvasható resid tulajdonképpen egy mutató az erőforrás-ellenőr processzben. Az RM-en kívüli világban a resid nem más, mint hivatkozás egy konkrét erőforrásobjektumra.
Végezetül a membership manager beállítja, hogy a quorumnak melyik állomás lesz a tulajdonosa. Persze ezt egyelőre még csak a memóriában található adatbázisváltozatban teheti meg, hiszen a quorumhoz még nem nyúlt hozzá a fürt.
A következő sorok egy tesztről tudósítanak, amelyet az RM végez el, és célja a lemez írhatóságának és olvashatóságának ellenőrzése. A teszt sikere után az ellenőr kiadja a lefoglalási *(reserve)* parancsot, az FM pedig bejegyzi, hogy sikeresen megszerezte az erőforrást.

```
Physical Disk <Disk Q:>: [DiskArb]Arbitrate
 returned status 0.
Physical Disk <Disk Q:>: [DiskArb] *********
 IO_PENDING **********.
[FM] FmGetQuorumResource successful
```

Az ellenőr újabb sorai tanúsítják, hogy a lemezt sikerült működő *(online)* állapotba állítani. Figyelem! Ez még csak a lemezre vonatkozik, nem a lemezt reprezentáló erőforrásra! A különbség mindjárt kiderül.
A most következő pár sorral gyakran lehet találkozni a Cluster.log elemzése közben. A művelet azt jelzi, hogy a Global Updated Manager *(GUM)* változtat a fürtadatbázis tartalmán. Az első ilyen tranzakció a naplóban így néz ki:

```
[GUM] GumSendUpdate: Locker waiting type 0 context 8
[GUM] Thread 0x7e8 UpdateLock wait on Type 0
[GUM] DoLockingUpdate successful, lock granted to 1
[GUM] GumSendUpdate: Locker dispatching seq
 35714364       type 0 context 8
[GUM] GumpDoUnlockingUpdate releasing lock ownership
 Physical Disk <Disk Q:>: [DiskArb]
 DisksOpenResourceFileHandle: Attach successful.
[GUM] GumSendUpdate: completed update seq
 35714364       type 0 context 8
[FM] FmpPropagateResourceState: resource
 f21ba2a4-62dd-4883-bf69-aa6c69f96320 pending event.
[FM] FmpRmOnlineResource: Resource f21ba2a4-
 62dd-4883-bf69-aa6c69f96320 pending
Physical Disk <Disk Q:>: [DiskArb]
 DisksOpenResourceFileHandle: CreateFile successful.
[FM] FmpRmOnlineResource: Returning. Resource
 f21ba2a4-62dd-4883-bf69-aa6c69f96320, state
 129, status 997.
```

A sikeres értelmezéshez ismerni kell a GUM frissítési műveleteinek három típusát, valamint a kontextus kódok jelentését. A táblázat a frissítési típusokat azonosítja.

| | |
|---|---|
| GUM type0 | GumUpdateFailoverManager |
| GUM type1 | GumUpdateRegistry |
| GUM type2 | GumUpdateMembership |

A kontextusszám értéke a frissítés típusától függ. Esetünkben a 8 erőforrásállapot-változást *(ResourceState)* jelent. A **[2]** helyről letölthető egy táblázat, amely a típusokat, a kontextus-számok jelentését és a később tárgyalandó állapotkódokat is tartalmazza.
A fürtszolgáltatásban kialakítottak egy mechanizmust, amely megakadályozza, hogy egy időben több szál végezzen módosítást az adatbázis adatain. A GUM kiad egy zárolási kérelmet, hogy a változtatást elvégezhesse, *(GumSendUpdate: Locker waiting)* megadva azt is, hogy milyen típusú adatmódosítás várható *(type 0 context 8)*. Azt az állomást, amely a módosítást végzi, zárolónak *(locker)* nevezik. A sikeres zárolásról bejegyzés készül *(DoLockingUpdate successful, lock granted to 1)*. Ezt követően végzi el a GUM a tényleges írást, amelyet egy folyton növekvő tranzakciós számmal jelöl meg *(GumSendUpdate: Locker dispatching seq 35714364 type 0 context 8)*, végül feloldja a zárolást *([GUM] GumpDoUnlockingUpdate releasing lock ownership)*. Az FM értesíti a többi állomást az állapotváltozásról *(jelen esetben ennek nincs értelme, hiszen nincs másik állomás)*. A többi bejegyzés érthető, egyedül a „state 129, status 997" kódokat kell feloldani, amit szintén az előbbi Excel táblázattal tehetünk meg. A 129 annyit tesz ClusterResourceOnlinePending, vagyis az erőforrás épp indul, a státus pedig a net helpmsg parancs segítségével lesz érthető: „Overlapped I/O operation is in progress". Most már érthető, hogy mi a különbség a tényleges erőforrás és a hozzá tartozó bejegyzés között. Miközben az ellenőr szerint a lemez már működik, addig a fürtadminisztrátor programban az erőforrás még csak indul!
A következő lépés a partícióhoz tartozó jelölés *(Q:)* egységességének ellenőrzése:

```
Physical Disk <Disk Q:>: Mountie[0]: 1, let=?,
 start=7E00, len=1EE28000.
Physical Disk <Disk Q:>: Online
Physical Disk <Disk Q:>: MountieVerify: Registry-
 System\DISK.GetInfo returned 0 [0:0].
Physical Disk <Disk Q:>: MountieVerify: ClusReg-
 DiskInfo selected.
Physical Disk <Disk Q:>: MountieVerify:
 DriveLetters mask is now 00010000.
Physical Disk <Disk Q:>: MountieVerify: Update
 needed for 08.
Physical Disk <Disk Q:>: FtInfo_Set: Update
 successful.
Physical Disk <Disk Q:>: MountieUpdate: Update
 needed for 00.
```

A hatodik sor jelzi, hogy a HKLM\SYSTEM\Disk bejegyzés nem friss. *(„Update needed for 08". Sajnos, az esetleges további kódszámokról nincs információ. Itt valószínűleg arról van szó, hogy a lemez becsatolása után a regisztrációs adatbázist a Windows 2000 még nem frissítette.)* A változtatást az erőforrás-ellenőr elvégzi, hogy aztán már azt írhassa ki „Update needed for 00". Újabb lemezellenőrzéseket végez az ellenőr, végül jelenti a fürtszolgáltatásnak, hogy az erőforrás állapota „működő"-re *(online)* állítható.

```
[RM] RmpSetResourceStatus, Posting state 2
 notification for resource <Disk Q:>
```

Egy újabb érdekességre figyelhetünk fel. Ahelyett, hogy a GUM módosítaná az erőforrás állapotát „indul"-ról „működő"-re, valami más kezdődik. A fürtadminisztrátorban még

nem történt állapotváltozás. A helyi adatbázis feladata lassan véget ér. Mindent előkészített a valódi quorum beolvasására. Ez az a hiányzó művelet, ami a GUM munkáját késlelteti.

### A valódi quorum betöltése

```
[FM] NotifyCallBackRoutine: enqueuing event
[FM] FmpCreateResStateChangeHandler: Entry
[FM] FmpCreateResStateChangeHandler: Exit, status 0
[FM] FmpHandleResStateChangeProc: Entry...
[DM] DmpQuoObjNotifyCb: Quorum resource is online
[DM] DmpQuoObjNotifyCb: Own quorum resource, try
 open the quorum log
[DM] DmpQuoObjNotifyCb: the name of the quorum
 file is Q:\MSCS\quolog.log
[LM] LogCreate : Entry FileName=Q:\MSCS\quolog.log
 MaxFileSize=0x00010000
[LM] LogpCreate : Entry
[LM] LogpMountLog : Entry
 pLog=0x00098498
[LM] LogpMountLog::Quorumlog Fil
 e size=0x00008000
[LM] LogpMountLog::reading 1024
 bytes at offset 0x00000400
[LM] LogpMountLog::checking LSN
 0x00000408
```

A quorumot a DM és az LM együttes munkával nyitja meg. Sajnos, a naplónak ez a szakasza nem pontosan dokumentált, a folyamat csak hozzávetőlegesen követhető. Nem ismert az „Entry pLog=0x00098498" jelentése, sem az, hogy miért épp 1024 bájtonként történik a beolvasás, mit rejt az LSN rövidítés *(logical sequence number?)*. Az alábbi pár sor is teljes rejtély:

```
[LM] AddTimerActivity:: hTimer = 0x000003b8
 pfnTimerCb=0x0107fb50 dwInterval(in
 msec)=120000
[LM] AddTimerActivity:: Interval(high)=0xffffffff
 Interval(low)=0xb8797400
[LM] AddTimerActivity:: returns 0x00000000
```

Mindeközben egyéb – szintén nem ismert folyamatok is zajlanak. A 0x768 szál a sorok közt elszórva a következő bejegyzéseket írta:

```
00000768: [LM] :ReSyncTimerHandles Entry.
00000768: [LM] :ReSyncTimerHandles Exit
 gdwNumHandles=2
00000768: [LM] :ReSyncTimerHandles Entry.
00000768: [LM] :ReSyncTimerHandles Exit
 gdwNumHandles=3
```

A gdwNumHandles egy változó, akárcsak a gdwQuoBlockingResources, de a szerepe, ellentétben az utóbbival, egyelőre nem dokumentált. A sok ismeretlen jelentésű sor között van néhány, amely fontos a számunkra:

```
[FM] HandleResourceTransition: Resource Name =
 f21ba2a4-62dd-4883-bf69-aa6c69f96320 old
 state=129 new state=2
```

A GUID, az állapot- és a státuskódok megfejtése után látható, hogy a „Disk Q:" erőforrás „indul" állapota „működik"-re váltott. Ezt egy adatbázis frissítésnek kell követnie, ami pár sorral lejjebb be is következik. Egy másik fontos bejegyzés az alábbi:

```
0000096c.000007e8::2002/07/19-19:39:09.185 [DM]
 DmpChkQuoTombStone - Entry
…
0000096c.000007e8::2002/07/19-19:39:09.185 [DM]
 DmpChkQuoTombStone: Exit, return
 ing 0x00000000
[FM] FmFormNewClusterPhasel, Entry.
 Quorum quorum will be deleted
```

Arról értesít minket a napló, hogy nem talált „quorum sírkövet". Előfordulhatott volna ugyanis, hogy miközben az épp most induló állomás állt, a másik állomás megváltoztatta a quorum helyét. Ekkor a mi állomásunk csak a hűlt helyét, vagyis egy „sírkőállományt" talált volna. Mivel azonban a sírkő nem mutat a quorum új helyére, az állomásunk nem tudta volna betölteni azt. Ebből az következik, hogy az állomás nem lett volna képes egyedül fürtszolgáltatást nyújtani, csupán csatlakozhatott volna egy másik, már működő node-hoz. A sírkő megléte esetén a szolgáltatásunknak haladéktalanul le kellett volna állnia. A bejegyzés szerint azonban ez nem következett be.

```
[DM] DmpApplyChanges: The current registry
 sequence number 35714363
[LM] LogGetLastChkPoint:: Entry
[LM] LogGetLastChkPoint: ChkPt File
 Q:\MSCS\chkF432.tmp ChkPtSeq=35714098
 ChkPtLsn=0x00000708 Checksum=176054
[LM] LogGetLastChkPoint exit, returning
 0x00000000
[DM] DmpLogFindStartLsn: LogGetLastChkPt rets,
 Seq#=35714098 ChkPtLsn=0x00000708
[DM] DmpLogFindStartLsn: ChkPt not applied,
 search for next seq
[LM]:LogScan::Entry Lsn=0x00000708 ScanForward=1
 CallbackRoutine=0x0106c056
…
[LM]:LogScan::Calling the scancb for
 Lsn=0x000010e0 Trid=35714231 RecordSize=216
[LM]:LogScan::Exit - Returning 0x00000000
[DM] DmpLogFindStartLsn: LSN=0x00000000, returning
 0x00000000
[DM] DmpApplyChanges: Exit, returning 0x00000000
[FM] FmFormNewClusterPhasel, Entry. Quorum quorum
 will be deleted
```

A napló szerint az LM megkeresi az utolsó checkpoint állományt, majd egy rutin segítségével ellenőrzi, hogy az adott tranzakciók bekerültek-e az adatbázisba. A bejegyzés kicsit csalóka, az állományt ténylegesen nem kell még egyszer alkalmazni, mert különben olyasmit is látnunk kellene, hogy „[DM] DmpLogFindStartLsn: chkpt uploaded from quorum log".
Az utolsó sor arról tájékoztat, hogy egy új fürtszolgáltatás kialakításának első fázisa úgy kezdődik, hogy a quroumot rögtön töröljük. A magyarázatot azonban már csak a következő számban adjuk meg…

Lepenye Tamás, MCSE 2000
lepenyet@mal.hu

**A cikkben szereplő URL-ek:**

[1] Eseménynapló
http://technet.netacademia.net/download/cluster/nodestart.log
[2] Excel tábla
http://technet.netacademia.net/download/cluster/farkasokkal XI.xls

# Active Directory: FSMO szerepek

Az előző részben végigvettük a tartománytelepítés lépéseit. Most megvizsgáljuk, milyen kiszolgálószerepek születtek. A hagyományos tartományvezérlői szerepeken túl ugyanis lett PDC Emulator, Schema Master, Domain Naming Master, RID Master és Infrastructure Master szerepkör is. Ezeket összefoglaló néven FSMO *(Floating Single Master Operations)* szerepeknek nevezzük.

### A többforrású *(multimaster)* replikáció

Az Active Directory mindegyik tartományvezérlője alkalmas arra, hogy ott felhasználókat hozzunk létre. Úgy gondolhatják, ez nem nagy csoda, de valójában mégis az. Gondoljunk csak bele: két távoli gépen egymástól függetlenül változik valami a címtárban *(itt létrejön egy objektum, amott átköltözik vagy törlődik egy másik)*, majd a replikáció során a két különböző címtártartalmat megpróbáljuk egységesíteni. Alapvetően három eset fordulhat elő:

- Az esetek elsöprő többségében semmi gond sincs a tartalomegyesítéssel, a módosítások nem zavarják egymást. A két, egymástól távoli gépen olyan módosítások keletkeztek, melyek teljesen függetlenek egymástól. Az egyik gépen mondjuk létrejött egy új felhasználó, a másikon pedig törölték X. Y. telefonszámát.
- Néhány esetben a módosítások egyazon objektum különböző tulajdonságaira irányulnak. Az egyik gépen kitiltjuk Samukát, a másik gépen kivesszük őt egy csoportból. Az Active Directory jól átgondolt replikációs stratégiája miatt ezekben az esetekben sincs gond, ugyanis az átvitt adatok legkisebb egysége nem egy komplett objektum, hanem annak egyetlen piciny attribútuma.
- Lényegesen ritkábban ugyan, de előfordul, hogy a módosítások ütik egymást: egyszerre két helyen ugyanazon az objektumon ugyanazokat az adatokat módosítják. Ilyenkor nincs „jobb" megoldás, az Active Directory automatikusan a későbbi módosítást érvényesíti a címtárban.
- Vannak olyan esetek, amikor a két módosítás teljesen ellentmondásos helyzetre vezetne. Ilyen eset, ha az egyik gépen új felhasználót teszünk egy olyan szervezeti egységbe, melyet a másik gépen gyorsan letörlünk. A replikációs konfliktus feloldásának egyik lehetséges megoldása az lenne, hogy az új objektum a „levegőben" jön létre, hisz kirántották alóla a szervezeti egységet. Vagy egyáltalán ne jöjjön létre? Ezt az esetet korábban részletesen is elemeztem, így csak utalok a megoldásra: az objektum a Lost and Found *(talált tárgyak osztálya)* tárolóba kerül.
- És végül, de utolsósorban vannak olyan adatok, ahol másodpercekig sem engedhető meg az ellentmondás. Ezeket az adatokat az AD nem engedi egynél több tartományvezérlőn módosítani. Ilyen például a gyermektartományok bejegyzése: egy pillanatig sem fordulhat elő, hogy két azonos nevű gyermektartomány létezzen, mert ezt a konfliktust a későbbiekben képtelenség feloldani. Az ilyen adatok számára az AD-ban egyetlen kijelölt módosítási pontot találunk.

A többforrású módosítás úgynevezett laza konzisztenciát eredményez, ami azt jelenti, hogy a címtár minden adott pillanatban tartalmazhat olyan adatot, melyről majd később derül ki, vajon megfelel-e a tárolás szabályainak. De vannak adatok, amelyeknél nem engedhető meg egyetlen másodpercnyi bizonytalanság sem, nem „ér rá" percekkel később megtudni, hogy befogadhatók-e. Ezeket az adatokat az arra kijelölt pontokon módosíthatjuk: a FSMO-gépeken.

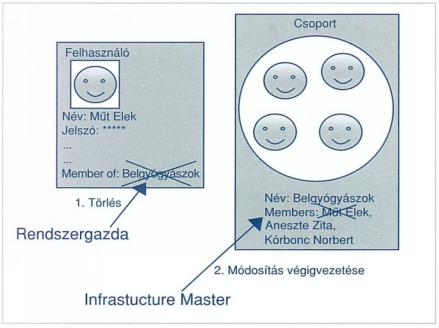
### FSMO szerepek

A bevezetőben felsorolt öt FSMO-szerep közül kettő erdőhárom pedig tartományszintű. Az erdőszintűekből az egész AD-erdőben egyetlenegy van, a tartományszintűekből viszont tartományonként egy-egy. *(A tartományszintűek replikációs határa egy adott tartományra korlátozódik.)* Röviden vegyük őket sorra:

- **PDC Emulator.** A leggyakrabban hivatkozott, tartományszintű FSMO-szerep, pedig ennek a kis kakukktojásnak nem is adatmódosítási, hanem kompatibilitási szempontból kell egyedül állnia. A régi ügyfelek *(Win95, Me, NT4)* ugyanis azt hiszik, minden tartományban van Primary Domain Controller, és csak ezt bízzák meg bizonyos módosításokkal, például jelszóváltoztatással. Ha nekik ez kell, hát „van" PDC. Emellett a szerep hordozója lesz a Domain Master Browser, a GPO Master és az időmester is. Négyet egy csapásra!
  *A PDC Emulator tartományszintű szerep.*
- **RID Master.** Az NT4-es címtárban *(SAM)* nem okozott gondot egyedi azonosítók *(SID-ek)* létrehozása, mivel csak egyetlen helyen, a PDC-n jöhetett létre objektum. Elosztott módosítási környezetben viszont már jelentős kihívás az itt-ott létrejövő objektumok azonosítóinak egyediségét biztosítani. Ezt az AD-ban úgy oldották meg, hogy továbbra is központi helyen találjuk a SID-gyárat, s a többi tartományvezérlő kétszázas csomagokban szállítja el a kész azonosítókat. Amikor új objektumot hozunk létre, az úgynevezett RID *(Relative Id)* poolból kapunk egyedi azonosítót.
  *A RID Master tartományszintű szerep.*
- **Infrastructure Master.** Ez a jószág a kócos tartományi adatok kifésüléséért felelős. Mitől válhat, sőt válik kócossá az AD tartalma? Attól, hogy az LDAP-szabványnak megfelelően hierarchikus felépítésű, de valójában adathálót tárol! Gondoljunk csak a felhasználók csoporttagságára: egy júzer több csoport tagja lehet, s egy csoportban számtalan tag szerepelhet. Ez tipikus many-to-many reláció, amit az AD úgy tárol, hogy mindkét fél „tudja", kikkel áll kapcsolatban. Ráadásul - a rendszergazdák életének megkönnyítésére – az összerendelési

adatok mindkét helyen módosíthatók. Ha egy felhasználón átállítom a csoporttagságot, a csoporton „után kell húzni" a megfelelő *(Member)* attribútumot, hogy a rend helyreálljon. Ez fordítva is igaz: ha egy csoportból kihajítunk valakit, az eltávolított objektum MemberOf attribútumát illik korrigálni.

*Az Infrastructure Master tartományszintű szerep.*




***Az Infrastructure Master elvégzi helyettünk a módosítások átvezetését, a konzisztencia megőrzését***

- **Schema Master.** Az objektumok tulajdonságkészletét a séma írja le. Ez egyetlen másodpercig sem lehet zavaros állapotban, mert akkor zavaros, önellentmondást hordozó objektumok születhetnének a címtárban. Nem lehet lebegtetni az a kérdést, hogy vajon a személyi szám attribútum numerikus vagy string típusú-e? A séma bővítését egyébként eléggé megnehezítették számunkra: a megfelelő MMC modul *(schmmgmt.dll)* települ ugyan, de nem regisztrálódik, csak a Schema Admins csoport tagjai jogosultak a módosítására stb.
  *A Schema Master erdőszintű szerep.*
- **Domain Naming Master.** Gyermektartomány vagy új fa telepítésekor az AD igen egyszerű módon gondoskodik a nevek egyediségéről: a Domain Naming Master dönti el, hogy az új jövevény beléphet-e. Ha ezt tudjuk, nem meglepő, hogy a DCPROMO.EXE erőteljesen támaszkodik erre a szerepre, s ha nem találja az erdőben, nem tud telepíteni. Számos Knowledge Base cikk szól erről. .NET Server 2003 újdonság: a Domain Naming Master lehetővé fogja tenni tartományok és DC-k átnevezését! *(Hogy ehhez mit fog szólni egy Exchange Server, azt egyelőre ne firtassuk.)*
  *A Domain Naming Master erdőszintű szerep.*

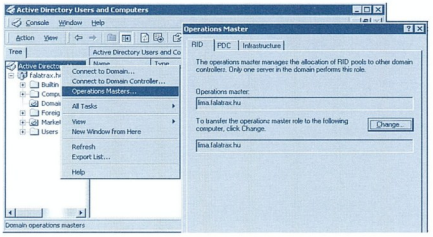
### Mit kell tenni az FSMO kiszolgálókkal?

Normális üzemeltetési körülmények között az égvilágon semmit. Ha azonban valamelyik FSMO-gép elromlik, vagy le kell állítanunk, záros határidőn belül pótolnunk kell a kiesett szerepeket. A gondos gazda például úgy vonja ki a hálózatból a Schema Mastert, hogy előzőleg ezt a FSMO-szerepet átviszi egy másik tartományvezérlőre. A tervezett átvitelt **FSMO-transfernek** nevezik. Ebben az esetben a jelenlegi FSMO-tulaj szabályos átadás-átvétel keretében vonul nyugállományba.

Nem várt gépkiesés esetén a szerep szabályos átvitelére nincs mód. Ha biztosak vagyunk abban, hogy a hiányzó gép sohasem kerül vissza a tartományba *(mert ellopták, kigyulladt stb.)*, az előző, eltűnt „tulaj" engedélye nélkül is átvihetjük a szerepet egy másik gépre. Ez a **seize** *(hatalomátvétel)* módszer. Ha lehet, kerüljük ez utóbbi megoldást, mert ha a halott gép mégiscsak feltámad, két masina is azt fogja hinni hogy ő az X-Master. Ezt a hibát csak az élőhalott újbóli megölésével korrigálhatjuk. A szerepek átvitele *(transfer)* grafikus, míg a hatalomátvétel *(seize)* parancssori eszközzel végezhető el. Ebből is látszik, hogy ez utóbbi módszer nem annyira támogatott.

### Transfer

A múltkor felvillantottam, hogy az ADUC-ban *(Active Directory Users and Computers)* hol találjuk az FSMO-szerepeket: jobbklikk a tartományon, Operations Masters.




***FSMO-szerepek megtekintése, átállítása ADUC-cal***

Ugyanitt nemcsak megtekinteni, hanem – megfelelő jogosultságok birtokában – átállítani is lehet, melyik szerepet melyik tartományvezérlő lássa el.

A Domain Naming Master az AD Domains and Trusts eszközzel, a Schema Master pedig a Schema MMC snap-innel költöztethető. Ez utóbbihoz be kell regisztrálni a fentebb említett, megfelelő DLL-t:

```
Regsvr32 schmmgmt.dll
```

### Seize

Erőszakos hatalomátvétel. Ehhez az NTDSUTIL nevű parancssori eszközt használhatjuk. Ennek parancskészlete, használata - talán szándékosan - annyira bonyolult, hogy a kisebb hitűek szakembert hívnak a feladat elvégzésére.

Emlékszem, pár hónapja a levelezési listákon közvetítette valaki hősies küzdelmét, s a többiek segítségével végülis két nap alatt megoldotta a problémát. Egy egyszerű PDC Emulator átvitel így nézne ki:

Legközelebb a replikációról mesélek.


**Fóti Marcell**
marcellf@netacademia.net
MZ/X

# Sémamódosítás plusz adatmódosítás

Az Active Directory nemcsak „gyári" adatok tárolására képes, hanem a séma bővítésével tetszőleges további attribútumok adhatók a meglévő objektumokhoz. Felvehető a címtárba például a felhasználók személyi száma, a számítógépek gyártási száma, a felhasználói csoportok gazdasági adatai stb.

A séma bővítése a megfelelő jogosultságok birtokában egyszerű feladat. A probléma ott jelentkezik, hogy ezt követően a felhasználói felület elemei *(pl. Active Directory Users and Computers, ADUC)* nem „veszik észre" a változást, azaz a friss attribútumok meg sem jelennek a felhasználói felületen – nemhogy módosítani lehetne őket. További munkabefektetésre van szükség ahhoz, hogy az új adatok hozzáférhetőkké váljanak.
Egyik lehetőség az ADUC tuningolása. Ehhez sajnos COM-objektumot kell faragni, amivel két baj is van:

- Valódi programozói munkát követel
- Csak az ADUC felhasználói, vagyis tipikusan a rendszergazdák élvezik az előnyeit

Ezt a lehetőséget tehát elvethetjük *(úgysem sikerülne az a COM-objektum ;)*, más megoldás után kell néznünk. Sajnos nincs olyan varázsló a Windowsban, amelyik ismerné a kibővített sémaelemek megjelenítésének búbáját, így a „programozó rendszergazda" vízió jegyében scriptírásra fogunk fanyalodni.
S ezzel két legyet ütünk egy csapásra. Miért? Mert a jó öreg VBScript-kód némi módosítással ASP-kóddá alakítható, így a kikísérletezett kód minimális változtatással weblapba ültethető! Az ASP-technológia ráadásul megadja az adatok módosításának kulcsát is: webes űrlapokat faraghatunk, melyen át könnyűszerrel töltögethető az AD.
Ebben a cikkben tehát teljes körű megoldást mutatok a séma kreatív felhasználására. Teljes körű alatt azt értem, hogy teljesen működni fog, de tudom, hogy más módszerek hasonlóképpen célravezetőek lehetnek. Célom, hogy olyan sablonokat adjak a Kedves Olvasók kezébe, amellyel könnyen el tudnak indulni saját, nagyszabású címtáralkalmazásaik megírása felé.
A lépések a következők:

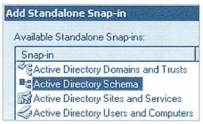
1. Sémabővítési előkészületek. A Schema Management snap-in regisztrálása.
2. OID generálása. A sémabejegyzések egyedi azonosítóval rendelkeznek. Honnan szedjünk azonosítót?
3. Új attribútum felvitele a sémába. Az attribútumok független elemként jönnek létre.
4. A User objektum kiegészítése. A függetlenek függővé tétele.
5. Az új attribútum kilistázása VBScripttel. Ötsoros ADSI-script a címtár tartalmának kilistázására.
6. Az új attribútum kilistázása ASP-lapon. Az előző ADSI-script web-lapba gyömöszölve.
7. ASP adatbeviteli űrlap. Egyszerű űrlapot készítünk a felhasználók számára.
8. ASP-kód a címtár módosítására. Az űrlap által összegyűjtött adatok bevitele a címtárba.
9. A megfelelő jogosultságok beállítása. NTFS-jogok? Címtárjogok?

### Sémabővítési előkészületek

A séma bővítéséhez eszköz kell. A Schema Management nevű MMC snap-in megteszi, csak éppen nem működik, amíg be nem regisztráljuk:

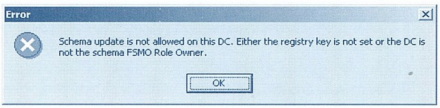
```
regsvr32 schmmgmt.dll
```

Ezután a parancs után megjelenik az MMC-modulok között a sémabővítési eszköz.




*A schmmgmt.dll regisztrációja után megjelenik az MMC-modulok közt a sémaszerkesztő*

Csak a Schema Admins csoport tagjai jogosultak a séma bővítésére. Még egy buktató: ha nem a Schema FSMO-szerepet futtató gépen próbálkozunk, hibaüzenetet kapunk:




*A séma csak a Schema Masteren bővíthető.*

Ez az üzenet akkor is felbukkanhat, ha a Schema Managerben nincs bepipálva az Operations Master menüpont mögötti ablakban a „The Schema may be modified..." pipa. Alapértelmezésben nincs bejelölve. Körkörös védelem a sémabővítés ellen.

### OID generálása

Új attribútum felvitelekor a séma bővítése két lépésből áll. Először az attribútumot, mint önálló, független bejegyzést kell megalkotni, s ezt követheti az objektumhoz csatolási lépés. A független, ide-oda csatolható attribútumok előnye, hogy több objektum is rendelkezhet azonos attribútummal anélkül, hogy többször létre kellene azt hozni. Többszörösen, sőt minden objektumon felhasznált attribútum például az X.500-útvonal *(DN)* illetve a relatív név *(CN)*.
A séma módosításához egy-két dolgot előre el kell döntenünk: mi legyen az új attribútum neve? Mi legyen az adattípusa

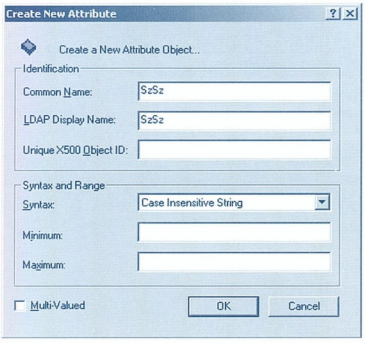
*(Distinguished Name, Case Sensitive String, Case Insensitive String, Boolean, Integer, SID)*? Ezek még a könnyebben eldönthető kérdések.
Nehezebb válasz adni erre a kérdésre: mi legyen az objektum OID-je?
Mi is ez az OID? Egy, az ISO-testület által kiosztott iparági azonosító. Az attribútum létrehozásakor az OID-mező kitöltése kötelező. Ha belekukkantunk a sémába, ilyen OID-ket láthatunk a gyári attribútumokon:

```
1.2.840.113556.1.5.4.1307
```

Ennek értelmezése balról jobbra a következő:
1=ISO
2=ANSI
840=USA
113556=Microsoft
1=Active Directory
5=Classes
4=Builtin-Domain
továbbá egy attribútumsorszám. Ez eddig szép, de honnan akasszunk le magunknak ilyen azonosítót? Ajánlott a Resource Kit megfelelő eszközével *(OIDGEN.EXE)* generálni egyet, vagy, ha megígérjük magunknak, hogy soha nem veszünk címtáralapú alkalmazást, akár hasraütéses módszerrel is dolgozhatunk. Például a 1.2.840.111111.1.5.x *(objektumosztályok)*, illetve a 1.2.840.111111.1.4.x *(tulajdonságosztályok)* tartományokat használhatjuk kiindulási alapként. Ezzel összeütközésbe kerülünk az 111111 azonosítójú céggel, de ha az nem az informatika területén működik *(az ISO azonosítók az összes iparágra kiterjednek)*, akkor kicsi az esélye, hogy bármikor belénk botoljanak a saját címtárunkban.

### Új attribútum felvitele a sémába

A Schema snap-inben jobbkattintva az Attributes csomóponton válasszuk a Create New Attribute menüpontot. Az alábbi látványban lesz részünk:

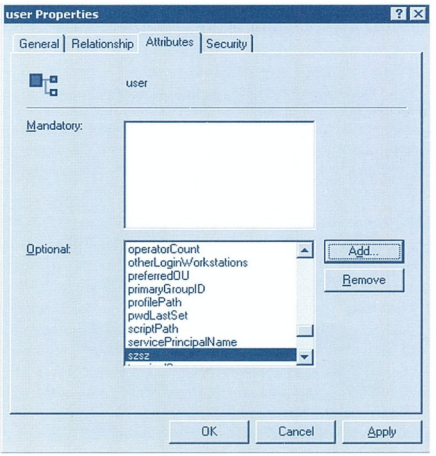



*A séma bővítése „személyi szám" (szsz) mezővel. A harmadik sor az ominózus OID-érték*

Töltsük ki a Common Name *(CN)* és Display Name mezőket az új attribútum nevével, az OID legyen az, ami, az adattípus *(személyi szám esetén)* legyen Case Insensitive String, minimum- és maximumértékeket ebben az esetben nem tudunk megadni. Kattintsunk az OK gombra, és az attribútum létrejön. Hol? A levegőben. Vagyis egyelőre nem kapcsolódik semelyik objektumhoz.

### A User objektum kiegészítése

A vadonatúj „szsz" attribútumot kapcsoljuk most hozzá a User objektumhoz! Válasszuk ki a Classes csomópontot a baloldali fában, s a jobboldalon felsorakozó kismillió objektum között keressük meg a Usert! Kettő kattanás, és megnyílik. Az Attributes fülön a felső kockában sorakoznak a User objektum kötelezően kitöltendő attribútumai *(lám-lám, egy sincs ilyen, mindegyik máshonnan öröklődik!)*, az alsóban pedig az opcionálisak. Ide kell felvennünk a „szsz" mezőt. Az alábbi ábrán látható, hol kell lennie, ha ügyesek voltunk:




*A User objektum opcionális mezői közé felvettük a „szsz", azaz személyi szám mezőt*

### Az új attribútum kilistázása VBScripttel

Most jön a neheze. Az új attribútum ugyanis sehol sem jelenik meg a felhasználói felületen. Ha nem teszünk semmit, ez így is marad. Azonban az alábbi kóddal könnyedén kilistázhatjuk a teljes címtárból a „szsz"-attribútumokat:

```
Dim ember, emberek
set emberek=
  GetObject("LDAP://cn=users,dc=falatrax,dc=hu")
emberek.filter=Array("user")
for each ember in emberek
  wscript.echo ember.name, ember.szsz
next
```

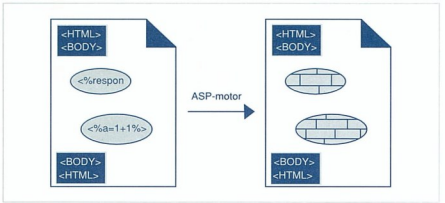
Ez a kód az **[1]** címről tölthető le *(szszlist.vbs)*, és a tartomány LDAP-útvonalának aktualizálása után vidáman listázza a személyi számokat – feltéve, hogy a Kedves Olvasó is elvégezte a címtármódosítást. Ha ez nem történt meg, ismeretlen attribútumra való utalással a programocska leáll.
Némi magyarázatot adok kódról. Ebben a VBScript-kódban az Active Directory Services Interface *(ADSI)* COM-objektumkupac segítségével tevékenykedünk a címtárban. Az ADSI a maga hierarchikus formájában teszi elérhetővé számunkra a címtárat, s ehhez csak a kezelni kívánt objektum LDAP-útvonalát kell ismerni.
A GetObject() függvény egy objektumkezelőt *(handle)* ad vissza a paraméterként megadott LDAP-útvonalon található objektumhoz, legyen az felhasználó, csoport vagy szervezeti egység.

A mi esetünkben ez a kezelő a Users tárolóra mutat. A második sorban a kezelő által elért objektumokat leszűkítjük a felhasználóobjektumokra *(filter)*. Ennek köszönhetően a harmadik sorban kezdeményezett ciklus csak User típusú objektumokat fog érinteni. Meglepő módon Computer objektumokat is ki fog listázni, ami bizonyos összefüggésre utal e két objektumtípus között. A for each – next ciklus végigfut az emberek-handle gyermekobjektumain, s a wscript.echo sorban kiírja a kimenetre a megtalált objektumok nevét és szsz attribútumát.

Remélem nem meglepő, hogy a listában a szsz mező nem szerepel, hisz mindenütt üres. Nem töltöttük még fel, mivel nincs mivel! De egyelőre örüljünk ennek a kis sikernek, és a listázási funkciót vigyük át a webre!

### Az új attribútum kilistázása ASP-lapon

Aki még sohasem alkotott ASP-weblapot, azok kedvéért mondom el, hogy itt egy olyan HTML-lapról van szó, amelyik VBScript (vagy más scriptnyelvi kód) szigeteket tartalmaz <% és %> „zárójelek" között, ahogy azt az alábbi ábra bal oldalán láthatjuk:



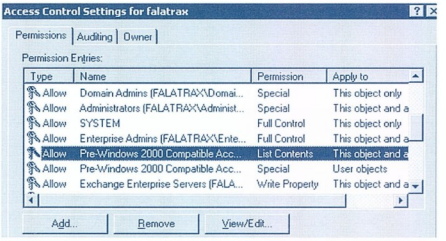

*Script-szigetek a HTML-tengerben: ez az ASP!*

A csoda akkor következik be, amikor ezt a lapot átszivattyúzzuk az IIS-webszerver ASP-motorján, vagyis ráböngészünk egy URL segítségével. Ekkor ugyanis a kódszigetek lefutnak, és HTML-lel tapasztják be azt a helyet, ahol korábban ők voltak. A kimenet pedig tiszta HTML-kód lesz. *(Ha csak egyszerűen megnyitjuk a böngészővel a fájlrendszerből az ASP-lapot, a csoda nem következik be, mert ha a lap nem IIS-től kerül hozzánk, nem jut szóhoz az ASP-motor, s a szigetek szigetek maradnak.)*

Ennyi ismétlés után lássuk azt az ASP-lapot! Ehhez körülbelül annyit kell tennünk, hogy az előző kódot <% és %> jelek közé zárva behányjuk egy .ASP-kiterjesztésű weblapba, illetve a kiíratást kicseréljük olyan megoldásra *(response.write)*, mely a weblapba ontja adatait. Az alábbi ASP-lap szintén az [1] címen található, neve: szszlist.asp.

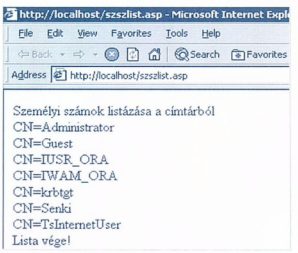
```
<HTML>
<BODY>
Személyi számok listázása a címtárból
<%
Dim ember, emberek
set emberek=
GetObject("LDAP://cn=users,dc=falatrax,
 dc=hu")
emberek.filter=Array("user")
for each ember in emberek
  response.write ember.name
  response.write " "
  response.write ember.szsz
  response.write "<br>"
next
%>
Lista vége!
</BODY>
</HTML>
```

Ezt a kódot mentsük le az InetPub\wwwroot könyvtár alá, és hívjuk meg böngészőnkből a http://gepem/szszlist.asp címen! Lefut? Ez az Active Directory biztonsági beállításaitól függ. Ugyanis az IIS egy speciális felhasználó, az IUSR_gepnev nevében futtatja az azonosítatlan *(anonymous)* felhasználók által letöltött kódot. Ennek a furcsa nevű felhasználónak csak abban az esetben van legalább listázási joga az Active Directoryn, ha mindenki másnak is Read joga van – azaz az AD telepítésekor NT4-biztonságra voksoltunk. Ekkor ugyanis a tartomány gyökerén Read joggal megtalálható a „Pre-Windows 2000 Access" nevűcsoport, melynek tagja az Everyone, melynek tagja az IUSR_gepnev fiók. Az alábbi ábra a Falatrax tartomány AD-jának gyökerén mutatja a jogosultságokat:




*Miért képes az anonim felhasználó a címtár listázására weben át? Mert így telepítettük az AD-t!*

Ha nem fut le a kód, hanem Access Denied hibaüzenetet kapunk, vegyük el a szszlist.ASP fálj NTFS-joglistájából az Everyone csoportot, és tegyükbe helyette például a Domain Adminst, vagy bármelyik másik csoportot, melynek nem tagja az IUSR_gepnev. Ez kikényszeríti az autentikációt, ezáltal megszünteti az anonim böngészést, és rendszergazdaként már bármit lekérhetünk a címtárból. Az eredmény ehhez hasonló:




*A szszlist.asp futásának eredménye*

Ha már működik, vizsgáljuk meg a böngészőben megjelenő lap forrását! Tiszta HTML-t látunk, mivel az összes kód kifejtődött!

Természetesen ez a megoldás épphogy csak működik, modellként, csontvázként használható. Komoly felhasználás esetén én tennék az elejére egy <%OPTION EXPLICIT%> megszorítást, amely megvéd a nemlétező változók használatától, illetve a címtárelérést is körül kellene bástyázni hibakezelő kóddal *(ON ERROR)*.

A személyi számok listája még mindig üres. Most már itt az ideje, hogy a módosítást is lehetővé tegyük.

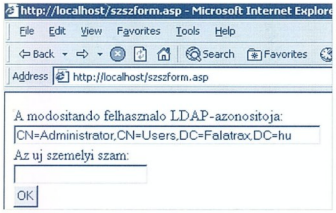
### ASP adatbeviteli űrlap

A webes adatfelvitel legalább két ASP-lapot igényel. Az egyik-

ből űrlapot készítünk, amely lehetővé teszi, hogy a felhasználók adatokat gépeljenek be, a másikat pedig az űrlap fogja meghívni a módosítás kivitelezése céljából. Kezdjük az űrlappal:

```
<HTML>
<BODY>
<FORM NAME="szszform" METHOD="GET"
 ACTION="szszmod.asp">
A modositando felhasznalo LDAP-azonositoja: <br>
<INPUT TYPE="text" NAME="FormObj" VALUE=
  "CN=Administrator,CN=Users,DC=Falatrax,DC=hu"
  size="50"><BR>
Az uj szemelyi szam: <br>
<INPUT TYPE="text" NAME="FormSzsz"><BR>
<INPUT TYPE="submit" VALUE="OK">
</FORM>
</BODY>
</HTML>
```

És kész. Letölthető az [1] címről, a fájl neve: szszform.asp, és így fest a böngészőben:




### *A szszform.asp űrlap működés közben*

A <FORM> HTML-tag alakítja ki az űrlapot, melyen két mezőnk van. Az elsőbe a módosítandó felhasználó LDAP-útvonalát kell begépelni, a másodikba pedig a személyi számot. A HTML-űrlapok kapcsolatmentességének következtében ez a lap önmagában nem képes elvégezni a módosításokat, minthogy letöltődése és böngészőbeli megjelenése után elszakad a kiszolgálótól. A FORM-tag viszont meg tud hívni egy másik lapot, amelynek a mezőértékeket is át tudja adni.

### ASP-kód a címtár módosítására

Ez a lap a mi példánkban a szszmod.asp, melyet nem meglepő módon az [1] címről lehet letölteni. Belsejében ismét némi ADSI-kódot találunk:

```
<HTML>
<BODY>
<%
Dim ember
set ember=GetObject("LDAP://"+request("FormObj"))
ember.szsz=cstr(request("FormSzsz"))
ember.setinfo
%>
A modositas vagy megtortent, vagy
 nem. A programozo nem irt
 hibakezelest
</BODY>
</HTML>
```

Felhívom a figyelmet a paraméterek átvételére. Az ASP-lap a request objektumban kapja meg a hívó fél üzenetét, a bemenő paramétereket. Ezek neve megegyezik az űrlapon használt mezőnevekkel *(FormObj, FormSzsz)*. A GetObject tehát megkapja az űrlap „FormObj" mezőjének értékét, s ez alapján csatlakozik a címtárhoz. Ezután a felhasználó „szsz" attribútumába beírjuk az űrlap második, „FormSzsz" mezőjének értékét.
Az *ember.setinfo* sor a módosítások véglegesítésére szolgál, ha ezt a sort kihagyjuk, a címtármező tartalma érintetlen marad!
Ha a listázásnál nem volt elegendő jogosultságunk a címtár elérésére, akkor a módosításokhoz sem lesz. Ezt látjuk a böngészőben:

```
Technical Information (for support personnel)
• Error Type:
  Active Directory (0x80070005)
  General access denied error
  /szszmod.asp, line 7
```

Ezen a fájlon is dobjuk ki az NTFS-jogosultsági listából az Everyone csoportot, és helyette a Domain Adminst tegyük be. Vajon milyen NTFS jogosultsággal kellene rendelkeznie a Domain Adminsnak a címtár írásához? Meglepő módon elég a Read, bár ennél nyilván többel rendelkezik. Vajon miért?

### A megfelelő jogosultságok beállítása

Kik tudják használni jelenleg ezt a webalkalmazást? A listázás beindításához az ASP-fájlok NTFS-jogosultságait változtatgattuk. Mi köze ennek a címtárbéli jogosultságokhoz?
Ha magán az ASP-fájlon nem tesszük lehetővé, hogy az IIS névtelen felhasználója, az IUSR_gepnev olvasási jogosultsággal rendelkezzen, a webkiszolgáló HTTP 404, Access Denied üzenettel válaszol a böngészőnek. Ezt megkapva az Internet Explorer azonosítási folyamatba kezd, és feldob egy bejelentkezési ablakot *(kivéve, ha engedélyezve van a Windows Integrated azonosítási mód, mert ebben az esetben automatikusan behelyettesíti az aktuális Windows-felhasználónevet és jelszót)*, így megadhatjuk bejelentkezési nevünk és jelszavunk.
Tehát az ASP-fájlok jogainak megkurtítása az azonosítás kikényszerítését szolgálja. A konkrét hozzáférési jogokat továbbra is az Active Directory joglistái *(ACL)* határozzák meg. A megfelelő jogosultságok beállításához tehát az ADUC szokásos lehetőségeit, jogosultságállítást, delegációt kell használni.

Végezetül hadd emlékeztessek mindenkit arra, hogy a sémamódosítás – ma még – visszavonhatatlan folyamat. Ha a cikk olvasása közben valaki az éles címtárát bővítette személyi szám attribútummal, biztos lehet benne, hogy ez örökre megmarad. Szólhattam volna előbb is…?

Fóti Marcell
MZ/X
marcellf@netacademia.net

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

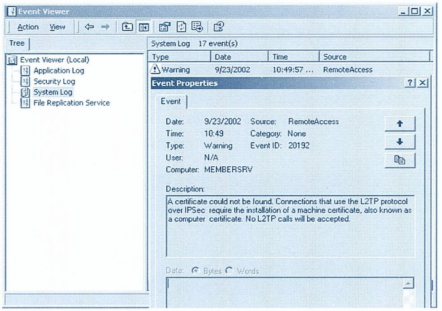
# VPN - Virtuális magánhálózatok

## 2. rész, Az L2TP protokoll

Az előző részben körüljártuk a VPN hálózatok „lényegét", Windows 2000 kiszolgálónkon varázsoltunk egy VPN kiszolgálót és csatlakoztunk is hozzá – egyelőre csak egy munkaállomással, PPTP kapcsolaton keresztül. Ma terítékre kerül az új protokoll, az L2TP is.

Az L2TP protokoll a PPTP és a Cisco által fejlesztett L2F *(Layer 2 Forwarding)* protokollok összeházasításával született. Windows 2000-ben maga az L2TP protokolltitkosítást nem, csak az alagúthálózat kialakításához szükséges funkciókat tartalmazza, így a „csupasz" L2TP forgalom bár alagúton halad, de az alagút fala üvegből van. Szerencsére ezzel az „üvegalagúttal" nem találkozunk, ugyanis a Windows 2000 egy ügyes trükkel az L2TP tudomása nélkül titkosítja a hálózati forgalmat. A titkosítást a Windows IPSec motorja végzi, automatikusan generált szabályok alapján *(amelyek kifejezetten az L2TP forgalom titkosítására készültek).*

Kapcsolódó IPSec cikksorozatunkban a dolog működéséről részletesebben is fellebbentjük majd a fátylat, egyelőre elégedjünk meg annyival, hogy az L2TP kapcsolat létrehozásához mind a kiszolgálónak, mind az ügyfélnek *(azaz az alagút mindkét oldalának, mégpedig nem a felhasználónak, hanem a számítógépeknek!)* rendelkeznie kell egy-egy nyílt kulcsú tanúsítvánnyal. Lehetőség van egyébként arra is, hogy a tanúsítvány helyett a kommunikáló tagok jelszóval hitelesítsék egymást, erről később lesz majd szó.

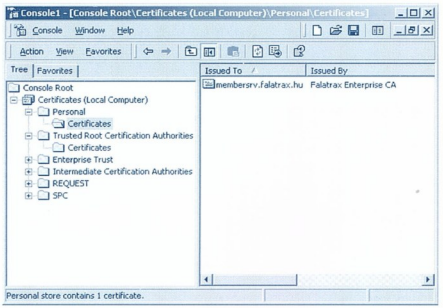



***Az RRAS minden induláskor ellenőrzi a számítógép tanúsítványát, és az eseménynaplóban jelzi, ha probléma van (esetünkben például nincs tanúsítvány)***

Alapértelmezésben, ha az RRAS kiszolgáló nem rendelkezik tanúsítvánnyal, az L2TP VPN kapcsolatokat meg sem kísérli felépíteni, úgysem sikerülne. Ilyenkor minden beérkező L2TP kapcsolatot azonnal elutasít. Erről a fenti ábrán is látható csinos kis eseménynapló-bejegyzéssel tájékoztat minket.

### Tanúsítványok a számítógépen

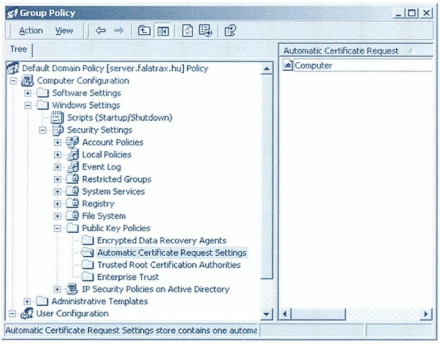
A számítógépnek is lehet tanúsítványa? Lehet bizony! Indítsuk el a Start menü Run parancsa segítségével a Microsoft Management Console-t *(mmc.exe)*, majd válasszuk a Console / Add/Remove snap-in... parancsot. A megjelenő ablakban kattintsunk az Add... gombra, a telepíthető modulok listájában kattintsunk kétszer a Certificates sorra, majd válasszuk a Computer account, majd a Local Computer opciókat. Ezzel megnyitjuk a helyi számítógép tanúsítványtárát. Ezt az MMC konzolt egyébként későbbi használatra érdemes elmenteni.




***Ennek a számítógépnek van tanúsítványa... vajon honnan szerezte?***

A tanúsítványok kezelése-terjesztése Windows 2000 tartományban igazán nem nehéz dolog. Ha valamelyik tagkiszolgálóra telepítünk egy tanúsítványkiadó szolgáltatást *(Enterprise, azaz vállalati üzemmódban)*, a Windows 2000 kiszolgálók és munkaállomások automatikusan beszerzik tőle a megfelelő tanúsítványokat *(mint az a fenti ábrán is látható)*. Ehhez egyetlen Group Policy beállításra van szükség *(amit a tartományvezérlőre telepített CA egyébként meg is tesz)*: a mindenkire érvényes alapértelmezett tartományi házirend *(Default Domain Policy)* Computer Configuration / Windows Settings / Public Key Policies / Automatic Certificate Request Settings pontja alatt – ha még nem létezik – hozzunk létre egy

új Request objektumot. Az objektum típusa legyen Computer. Ezután már csak a használni kívánt CA nevét kell megadnunk – a listában a tartományba telepített vállalati tanúsítványkiadók láthatók.




### *Automatikus tanúsítványkérés beállításai a tartományi alapértelmezett csoportos házirendben*

A házirend érvényesítése után a Windows 2000 munkaállomások és kiszolgálók automatikusan beszerzik a nekik szükséges tanúsítványokat, sőt, idővel azok frissítésére is képesek lesznek. A rendszer Windows 2000 tartományon belül szinte teljesen automatikus. Emlékszünk az előző cikk végén bemutatott két ábrára? Ott, míg a tartományon kívüli Windows XP csak PPTP, a tartományi tag Windows 2000 automatikusan L2TP protokoll segítségével csatlakozott a kiszolgálóhoz. A magyarázat ez: a Windows 2000 Professional tartományi tag lévén hozzáfért a csoportos házirendhez, és az utasításoknak megfelelően telepítette a saját kis tanúsítványát.

#### Tanúsítványok telepítése „kézzel"

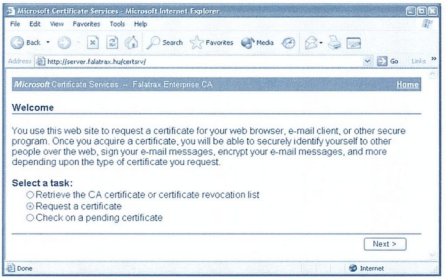
No igen, de mit tehetünk, ha a fenti feltételek bármelyike nem áll fenn, azaz:

- Nincs vállalati tanúsítvány-kiszolgálónk, vagy nem a sajátunkat szeretnénk használni
- A számítógépek nincsenek tartományban, így automatikusan kérni sem tudnak?

Mindkét esetben kettős célunk van: az első, hogy valahogy tanúsítványhoz jussunk. Ehhez használhatunk bármely tanúsítvány-kiszolgálót, akár Windows 2000 CA-t is. A Windows 2000 tanúsítvány-kiszolgálótól a szolgáltatás webes felületén keresztül kérhetünk tanúsítványt, más szolgáltatások esetén érdeklődjünk a CA üzemeltetőjétől. A második cél pedig a megszerzett tanúsítvány telepítése. Ha a Windows 2000 CA-t használjuk *(a webes felületen keresztül)*, ez automatikusan megtörténik, de ellenőrizhetjük a számítógép tanúsítványtárában. Ha a tanúsítványt más szolgáltatótól kapjuk, ugyanitt importálhatjuk a rendszerbe.

#### Tanúsítványkérés weben keresztül

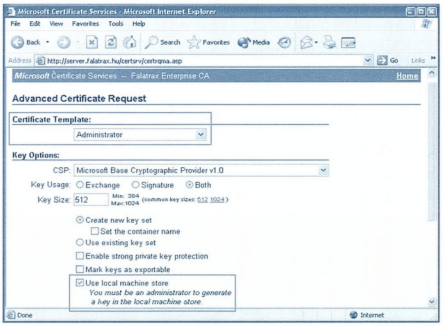
Lássuk mi a teendő, ha a Windows 2000 CA webes felületén keresztül szeretnénk tanúsítványhoz jutni! Esetünkben a tartományon kívüli Windows XP-nek szeretnénk tanúsítványt szerezni. Ennek érdekében a böngésző segítségével megnyitjuk a CA weboldalát a http://server.falatrax.hu/certsrv címen *(jelentkezzünk be egy, a CA-hoz megfelelő jogosultságokkal bíró tartományi rendszergazda nevével és jelszavával)*.




### *A Windows 2000 CA webes felülete*

Ez valójában egy varázsló, amely lépésről lépésre végigvezet a tanúsítványkérés folyamatán. Tartsunk vele!

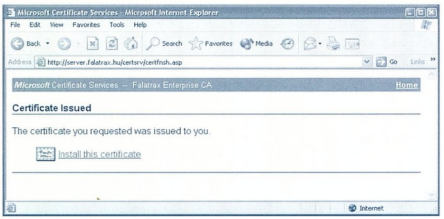
- Válasszuk a „Request a certificate" opciót, majd Next!
- A következő oldalon válasszuk az „Advanced Request", azután pedig a „Submit a certificate request to this CA using a form" lehetőséget!




### *A kért tanúsítvány paraméterei*

A fent látható „Advanced Certificate Request" oldalon a Certificate Template mezőben attól függően válasszunk tanúsítványtípust, hogy egyedülálló *(stand-alone)* vagy vállalati CA-hoz csatlakozunk. Stand-alone CA esetén a „Client Authentication Certificate" vagy az „IPSec Certificate" közül válasszunk *(előbbi „felhasználó"-azonosításhoz is, utóbbi csak az IPSec csatornák kiépítéséhez használható)*, vállalati CA esetén pedig – mint az ábrán is látható – az Administrator típusú tanúsítványra lesz szükségünk.

Aki nem hiszi, járjon utána **[1]**! Fontos még az ábra alján látható másik bekeretezett rész: „Use local machine store", azaz a tanúsítványt a helyi számítógép tanúsítványtárába telepítsük. Ha a hozzá tartozó privát kulcsot később exportálni is szeretnénk *(például adatmentési célból)*, válasszuk ki az egy sorral feljebb látható „Mark keys as exportable" lehetőséget is.




*A kész tanúsítványt egy kattintással telepíthetjük*

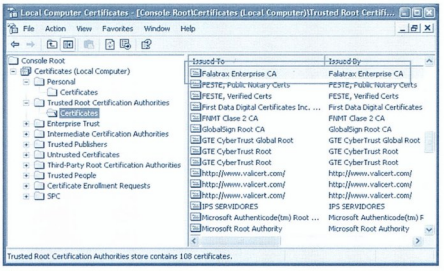
Miután a kérést elküldtük, a vállalati CA – megfelelő jogok birtokában – azonnal elkészíti a tanúsítványt, és a következő oldalon egy kattintással telepíthetjük is azt. Ha a CA nem vállalati típus, várnunk kell addig, míg egy rendszergazda jóvá nem hagyja a kérésünket. Ezután visszatérhetünk a kiszolgáló weboldalára és hozzájutunk a hőn áhított tanúsítványhoz.

### További feltételek a tanúsítványokkal kapcsolatban

A sikeres L2TP kapcsolat kiépítéséhez tehát mindkét tagnak rendelkeznie kell érvényes tanúsítvánnyal, bár mint a mellékelt ábra mutatta, az nem feltétel hogy az a bizonyos tanúsítvány annak a „nevére" szóljon, aki azt használja *(ettől persze ez még biztonságos, hiszen a privát kulcs csak nála található meg)*. A tanúsítvány érvényességét két tényező befolyásolhatja még:

- A tanúsítvány lejárati ideje *(ellenőrizhetjük, ha a tanúsítványra kettőt kattintunk)* és egyéb jellemzői, pl. a visszavonási státusza
- A tanúsítványt kiadó szervezetbe vetett bizalmunk

No ez utóbbi már érdekes kérdés. Hogy kikben bízunk? Nos, a Windows beépítve tartalmaz egy listát a „megbízható" tanúsítványkiadókról. Ezután minden olyan tanúsítványban megbízunk, amit olyan CA adott ki, aki ebben a listában szerepel, vagy legalábbis, ha a kiadó CA-nak van olyan saját tanúsítványa, ami végső soron egy ilyen listatagtól származik *(azaz, a CA-k közti hierarchiában egy megbízott CA-nak „gyermeke")*. Ez a lista a megbízott gyökérkiszolgálók listája, „magyarul" a „Trusted Root Certification Authorities".




*Ellenőrizzük, hogy a házi CA-nk szerepel-e a megbízott gyökérkiszolgálók listájában!*

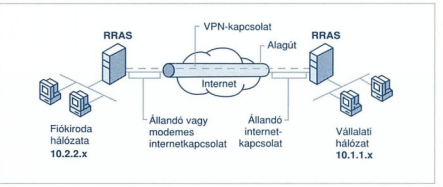
Windows 2000 tartományi környezetben természetesen minden vállalati CA bekerül a bizalmi listába, és persze kézzel is vehetünk fel újabbakat. Tartományon kívül azonban érdemes figyelni, és ellenőrizni, hogy a saját CA-nk szerepel-e ebben a listában. A webes tanúsítványtelepítés során ez a bejegyzés is létrejön, de ha mégsem, a tanúsítványkiadó weboldaláról letölthetjük az ő saját tanúsítványát, és kézzel importálhatjuk a listába. A lényeg, hogy meglegyen. Visszatérve a számítógépek saját tanúsítványaira: egy gépnek több tanúsítványa is lehet, a sikeres csatlakozás feltétele az, hogy mindkét gép rendelkezzen legalább egy olyan tanúsítvánnyal, amely:

- ugyanattól a tanúsítványkiszolgálótól származik
- ráadásul ebben a közös tanúsítvány-kiszolgálóban mindketten megbíznak

Ennyi. Pofonegyszerű, nemde? :-)

### Hálózatok összekapcsolása VPN-en keresztül

Eddig *(gondolhatnánk)* arról volt szó, hogy mi módon csatlakoztathatunk egy-egy számítógépet, munkaállomást a vállalati hálózathoz virtuális magánhálózaton keresztül.




*Alhálózatok összekapcsolása VPN-nel*

A valóság az, hogy ha nem egy számítógépet, hanem egy teljes alhálózatot csatlakoztatunk, a módszer teljesen hasonló, csak a virtuális magánhálózatot nem az egyes számítógépek, hanem az alhálózat „határán" álló RRAS kiszolgáló építi fel – a másik alhálózaton található másik RRAS kiszolgáló felé. Az RRAS kiszolgálók feladata, hogy a túloldali alhálózatokba haladó illetve onnan érkező csomagokat a megfelelő módon továbbítsák *(routolják)*. Ezt a hálózati konfigurációt ezért router-to-router VPN kapcsolatnak is nevezik.

Létrehozhatunk „féloldalas" hálózatot is, amikor mindig csak az egyik oldal *(általában a fiókiroda)* kezdeményezi a kapcsolatot, amikor arra igény van. A fiókiroda RRAS szolgáltatása ilyenkor csak „ügyfélként" üzemel, a VPN szolgáltatást csak a központi RRAS-ra kell telepíteni *(akit hívunk)*. Állandó internetes kapcsolatra is csak a központi szolgáltatásnak van szüksége, a fiókiroda ugyanis azt saját magának képes igény szerint kiépíteni *(pont ugyanúgy, mint az ügyfél, aki a VPN-re való csatlakozás előtt tárcsázza az Internet-szolgáltatóját)*. Ha az állandó kapcsolat mindkét oldalon megvan, létrehozhatjuk a szimmetrikus VPN-t is. Ebben az esetben mindkét RRAS kiszolgáló egyidejűleg VPN kiszolgáló és ügyfél is, a kapcsolat kiépítését pedig mindig az kezdeményezi, aki éppen szeretne „átlátni" a másik alhálózatba. Honnan tudja az RRAS, hogy VPN kapcsolatot kell *(lehet)* kiépíteni, kivel, milyen módon? Honnan tudja az RRAS hogy mi van a „másik" alhálózatban? Honnan tudja az RRAS, hogy a másik alhálózatot a VPN kapcsolaton keresztül érheti el? Ezekre a kérdésekre kell sorban választ adnunk néhány varázsló segítségével.

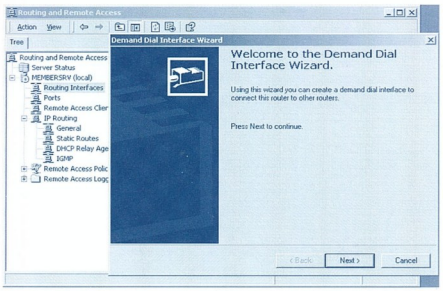
### Új demand-dial kapcsolat létrehozása

Tegyük fel, hogy a feladat a fentebb látható két alhálózat *(a központi 10.1.1.x és a vidéki 10.2.2.x)* összekapcsolása. Ehhez mindkét kiszolgálóra telepítettük és konfiguráltuk már az RRAS szolgáltatást, VPN kiszolgáló üzemmódban. Természetesen rendelkezésünkre állnak az L2TP használatához szükséges tanúsítványok is. A következő lépés, hogy létrehozzuk a

VPN automatikus felépítéséhez szükséges elemeket mindkét kiszolgálón. Ezek:

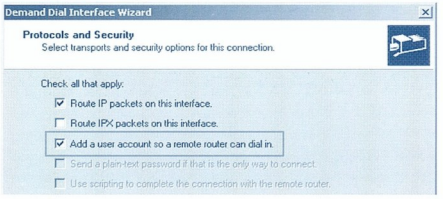
- Egy úgynevezett „demand-dial interface", ami gyakorlatilag megfelel az ügyfélgépeken a telefonos hálózati kapcsolat ikonjának, csak ezt az RRAS képes automatikusan tárcsázni
- Egy felhasználónév arra az esetre, ha a távoli RRAS szeretne bejelentkezni hozzánk
- Egy bejegyzés az útválasztási táblában, ami jelzi, hogy a távoli alhálózat a demand-dial kapcsolat túloldalán található

Először hozzuk létre a VPN interfészt. Üljünk le a vidéki számítógép elé, és az RRAS konzolban a Routing Interfaces sorra kattintva válasszuk a „New demand-dial interface" parancsot!




### *Az új VPN interfészt létrehozó varázsló*

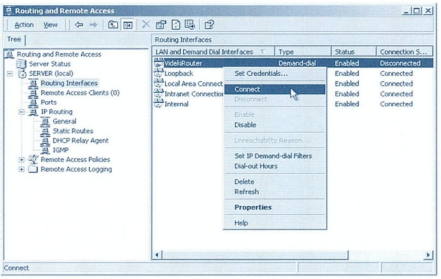
A varázsló első kérdése a létrehozandó kapcsolat nevére vonatkozik. Azt hihetnénk, hogy bátran választhatunk bármit, azonban vigyázzunk: ha nem csak kitárcsázni szeretnénk, hanem a behívást is engedélyezzük, a varázsló ezt a nevet fogja adni a túloldali eszköz számára létrehozott felhasználónak. Ezért érdemes olyan nevet választani, ami utal a túloldali eszközre, nem tartalmaz szóközt és ékezetes betűket. Miután mi a vidéki router előtt ülünk, a név most legyen: „KozpontiRouter". Ezután kiválaszthatjuk a kívánt VPN protokoll típusát: PPTP, L2TP vagy automatikus, majd meg kell adnunk a túloldali VPN kiszolgáló külső IP címét, ahova az alagutat építeni fogjuk.




### *Hozzunk-e létre helyi felhasználói fiókot a betárcsázó router számára?*

Ezután az ábrán is látható oldal következik: kérhetjük az IP és IPX csomagok továbbítását a kapcsolaton keresztül, valamint megkérhetjük a varázslót, hogy hozzon létre helyi felhasználót a távoli routertől beérkező hívások számára. Figyeljünk rá, hogy ez a felhasználó tagkiszolgálók esetén nem az Active Directoryba, hanem a számítógép saját felhasználói adatbázisába kerül! Erre akkor kell figyelnünk, amikor majd a túloldalon meg kell adnunk a betárcsázáshoz szükséges felhasználónevet, jelszót és tartományt: ez utóbbi ekkor nyilván nem a Windows tartomány, hanem a számítógép neve lesz. *(Természetesen a behíváshoz nem feltétlenül kell az így létrehozott felhasználóneveket használni, bármilyen fiók – tartományi is – megfelel, ha rendelkezik a megfelelő behívási jogosultságokkal).*

A következő két oldalon meg kell adnunk először az újonnan létrehozott felhasználó jelszavát, majd a kitárcsázáshoz szükséges adatokat: felhasználónevet, tartományt és jelszót. A VPN interfész ezzel elkészült. Végezzük el ugyanezeket a műveleteket a másik kiszolgálón is *(persze értelemszerűen az ellentétes neveket választva)*. Az interfész ettől a pillanattól kezdve – bár még nem automatikusan, de kézzel – már csatlakoztatható.




### *Az interfész kézzel máris használható*

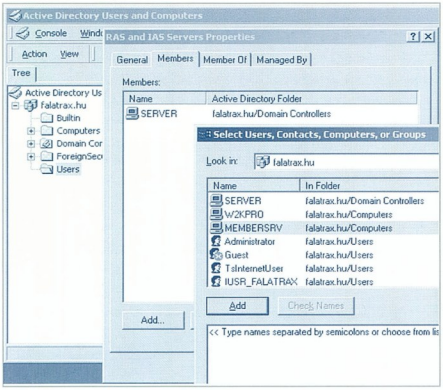
A Connect parancs segítségével próbáljunk aktiválni a VPN kapcsolatot. Ha munkánkat jól végeztük, néhány másodperc alatt a kapcsolat bármelyik irányba *(de nem egyszerre!)* kezdeményezve kiépül. Ezt az interfész Connection State oszlopának „Connected" értéke jelzi. Ha a csatlakozás nem sikeres, az „Unreachability Reason" *(és az eseménynapló)* elárulja nekünk, hogy mi lehet a probléma. Ha a felhasználónév / tartomány / jelszó nem megfelelő *(emlékezzünk, hogy a varázsló csak akkor hoz létre tartományi felhasználót, ha az RRAS tartományvezérlőn fut)*, akkor a Set Credentials... parancs segítségével újra megadhatjuk ezeket az adatokat.

Végül a Properties hatására előbukkanó dialógusablakban ellenőrizhetjük és módosíthatjuk a varázslóban megadott és egyéb tulajdonságokat *(például hogy mennyi idő múlva szakítsa meg a kapcsolatot, ha nincs forgalom rajta)*. Figyelem! A VPN kapcsolat mindig legyen demand-dial, de a várakozási időt kikapcsolhatjuk. Érdemes az újratárcsázási beállításokat is áttekinteni.

Ne ijedjünk meg, ha a sikeresnek tűnő csatlakozás után a Remote Access Clients továbbra is 0 kapcsolatot jelez, a router-to-router kapcsolatot az RRAS itt nem számolja. Ha viszont a Ports sorra kattintunk, láthatjuk, milyen portok aktívak *(rendezzük sorba ezeket is állapot szerint)*. Itt rögtön látszik az is, hogy PPTP vagy L2TP kapcsolatot sikerült kiépíteni. Az L2TP kapcsolat előfeltételei és megoldásai pontosan megegyeznek a korábban leírtakkal.

### Csatlakozási probléma tartományi fiókok esetén

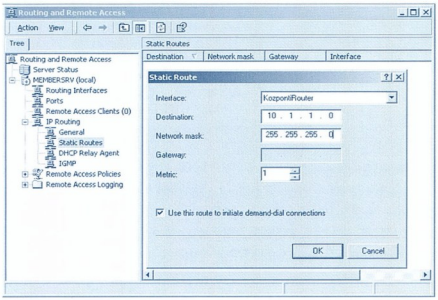
A legjobb szakemberekkel is megesik néha, hogy átsiklanak egy-egy, szinte minden oldalon hangsúlyozott szabály felett. Ha az RRAS telepítése után a csatornák felépítése nem sikerül, mégpedig az eseménynapló szerint azért, mert „a hitelesítő kiszolgáló nem elérhető…" – nos akkor első dolgunk legyen ellenőrizni, hogy az RRAS kiszolgáló tagja-e az Active Directory-ban található RAS and IAS Servers csoportnak.




*Az RRAS kiszolgálót fel kell vennünk a tartományi „RAS and IAS Servers" csoportba*

### A statikus útvonal felvétele

Már csak egy dolog maradt hátra: megmagyarázni az RRAS-nak, hogy a távoli alhálózatokon milyen címtartomány található, illetve azt, hogy ha ide szeretnénk csomagokat küldeni, akkor előtte fel kell építeni a VPN kapcsolatot, majd azon továbbítani az adatokat. Ezt statikus útválasztási útvonal *(static route)* létrehozásával tehetjük meg. Emlékezzünk: a központi alhálózat címtartománya 10.1.1.x, a vidékié pedig 10.2.2.x. Mindkét kiszolgálón az „ellentétes" címtartományhoz kell majd hozzárendelnünk az előzőleg létrehozott demand-dial kapcsolatot. Az RRAS konzol „IP routing / Static Routes" során válasszuk a „New Static Route…" parancsot.

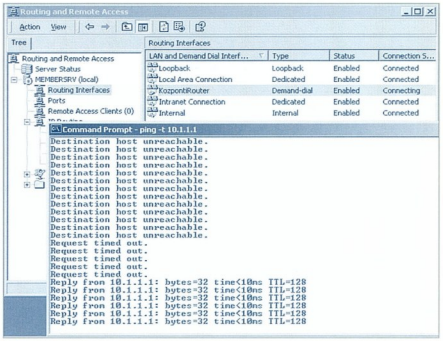



*Kőbe vésett útvonal: a 10.1.1.x bizony a központi router felé keresendő*

Itt válasszuk ki a VPN interfészt, a Destination mezőbe írjuk be a távoli alhálózat hálózatazonosítóját és alhálózati maszkját, valamint hagyjuk bekapcsolva a „Use this rule to initiate demand-dial connections" opciót. Így az RRAS kiszolgáló, amikor a távoli hálózatba tartó csomagot észlel, felépíti a kapcsolatot és továbbítja – alagútban, titkosítva – az adatokat. Ugyanezt a beállítást persze meg kell tennünk a másik kiszolgálón is.

### A puding próbája … a ping

Ezután már csak a puding próbája, a ping van hátra. Ha a VPN interfész épp aktív lenne, kapcsoljuk le, majd nyissunk egy parancssori ablakot és próbáljunk megpingelni egy „távoli" IP címet.




*A ping aktiválja a VPN-t… győztünk!*

Míg a VPN kapcsolat inaktív, „Destination host unreachable" üzenetet kapunk, viszont ezalatt az RRAS elkezdi a kapcsolat felépítését *(„Connecting…")*. Ahogy a kapcsolat felépült, némi gondolkodás után a válaszok is visszaérkeznek: győztünk.

Következő számunkban a DNS, WINS és egyéb címzési mizériáknak, finomságoknak nézünk utána, valamint szó esik majd a RADIUS kiszolgálókról is. Addig is, jó alagutazást!

folytatjuk…

Fülöp Miklós
mick@netacademia.net

A cikkben szereplő URL-ek:

[1] http://support.microsoft.com/default.aspx?scid=kb;en-us;Q253498

# Digital Money

Az elektronikus kereskedelemről és az elektronikus aláírásról szóló törvények megteremtették Magyarországon is a jogszabályi hátterét annak, hogy az Interneten keresztül áruk, illetőleg szolgáltatások cseréljenek egymással gazdát, vagyis végre meginduljon az elektronikus kereskedelem.



Mint a való térben, a virtuális világban is az áruért, vagy szolgáltatásért ellenszolgáltatás jár cserébe, vagyis az esetek nagy többségében pénz. Az ellenérték elektronikus úton való teljesítése során a leggyakoribb félelem a biztonság hiánya, az hogy személyes adatainkhoz hozzáférve bankszámlánk terhére illetéktelen kifizetések is teljesíthetők. Bebizonyosodott, hogy a hitelkártya nem ideális megoldás az Interneten keresztül történő kifizetésekre. Technikai megoldásként a piac a digitális pénz megalkotásával reagált. **A digitális pénz egy bitsorozat, pénzhelyettesítő fizetési eszköz, amely ellenérték fejében vásárolható a kibocsátótól, és a számítógép merevlemezén, elektronikus pénztárcában vagy chipkártyán tárolható.**

A digitális pénzen alapuló leegyszerűsített fizetési modell a következőképp alakul:

1. az ügyfél átutal a bankszámláról 300.000.-Ft-ot a digitális pénz kibocsátójának számlájára,
2. a kibocsátó a megfelelő virtuális forintösszeget jóváírja az ügyfélnek,
3. a 300.000.-Ft mindaddig a kibocsátó számláján marad, amíg valaki, akihez a virtuális forintok kerültek azt be nem váltja,
4. az ügyfél kifizetéseket teljesít a digitális pénzzel,
5. a kereskedő a megkapott összeget visszaküldheti a kibocsátónak és átalakíttathatja igazi pénzzé, vagy kifizetéseket eszközölhet vele további kereskedők felé,
6. a digitális pénz egyedi sorszámmal van ellátva, melynek célja a többszörös felhasználás, és egyéb csalások kiküszöbölése. A felhasználó pénzügyi szokásai, így a pénz nyomon követhetőségének elkerülése végett a vak aláírás *(blind signature)* módszerét alkalmazzák. Ezt a protokollt akkor használjuk, ha a másik féllel nem akarjuk közölni, hogy mit ír alá. Ez jelen esetben azért jelent problémát, mert a banknak a pénzt hitelesítenie kell anélkül, hogy ismerné a pénz sorszámát.
7. az átváltás azt jelenti, hogy a kibocsátó leveszi a virtuális forintnak megfelelő összeget az ügyfél számlájáról, és átutalja a beváltást kérőnek. Profit termelésre akkor van lehetőség, ha a valódi pénz befizetése és kifizetése közti időtartamra be tudta fektetni ezt az összeget.

A digitális pénzhez kapcsolódóan több nagy projekt van folyamatban, melyek főként az Interneten való fizetéshez kapcsolódó problémákat igyekeznek megoldani. Ha a digitális pénz tényleg domináns szereplőjévé válik az Internetes piacnak, akkor fontos a megfelelő szabályozás kialakítása. Kérdés, hogy a jogszabályoknak vagy az önszabályozásnak kell nagyobb teret engedni ezen a területen, annyi viszont bizonyos, hogy elengedhetetlen egy technológiafüggetlen szabályrendszer kialakítása.

### Európai Unió

Jelenleg az Európai Unión belül a 2000/46/EC irányelv szabályozza az elektronikus pénz kibocsátásának kérdéskörét. Az 1. Cikkely értelmében az elektronikus pénz olyan pénzhelyettesítő eszköz, amelyet elektronikus úton, mint például a chip kártyán vagy számítógép memóriában tárolnak, és általában korlátozott összegű kifizetések teljesítésére szolgál.

Kibocsátására – a magyar szabályozástól eltérően – nem csak hitelintézetek jogosultak, hanem olyan jogi személyek is, amelyek megfelelnek az irányelvben részletezett szigorú követelményeknek. Ilyen például az induló tőke 1 millió EURO-ban történő meghatározása, illetve az az előírás, hogy a működés során a tőke nem csökkenhet az irányelvben meghatározott szint alá. A társaságnak csak meghatározott feltételek mellett lehet részesedése más társaságokban, és az elektronikus pénz kibocsátásán kívül is csak az irányelvben meghatározott más tevékenységeket végezheti, mint például adattárolás, elektronikus pénz kezelése. Az irányelv kimondja a digitális pénz visszaváltásának kötelezettségét, és azt, hogy a felek közti szerződésben ennek módját szabályozni kell. Lehetőséget ad a kibocsátónak egy visszaváltási minimumérték meghatározására, amely azonban nem haladhatja meg a 10 eurót.

Az elektronikus pénzhez kapcsolódóan az Európai Unió szabályozása elvárásokat, ajánlásokat fogalmaz meg a kibocsátó és a felhasználó közötti szerződések tartalmi elemei tekintetében is. A tárgykörben már 1997-ben kiadott ajánlás két csoportra osztja az elektronikus fizetési eszközöket:

- a távolról hozzáférést biztosító fizetési eszköz az ügyfél számára lehetővé teszi, hogy a kártyával a pénzintézetnél elhelyezett számlája terhére – a biztonsági kódok megfelelő használata mellett - meghatározott pénzügyi műveletet végezzen;
- az elektronikus pénzeszköz olyan újratölthető kártya vagy számítógép-memória, amelyen értékegységek elektronikus úton tárolhatók, lehetővé téve használójа számára a fizetést.

Ezen kívül - a tranzakciók feltételeinek áttekinthetőségét előtérbe helyezve - szabályozza a felek közötti szerződés tartalmi elemeit, valamint részletezi a kibocsátó és a felhasználó felelősségét és kötelezettségeit. Emellett az elektronikus fizetési eszköz elvesztése esetén bejelentési kötelezettséget ír elő, illetve a szerződő felek közti alternatív vitarendezési lehetőségek megteremtéséről szól.

### Magyarország

Az Uniós szabályozással összhangban Magyarországon a pénzforgalomról, a pénzforgalmi szolgáltatásokról és az elektronikus fizetési eszközökről szóló 232/2001. (XII.10.) Kormányrendelet szabályozza ezt a kérdéskört.

Magyarországon a digitális pénz jogszabályi alapjait megteremtették, bár az uniós szabályozással ellentétben csak a hitelintézetek kapnak lehetőséget digitális pénz kibocsátására, amin érdemes lenne változtatni, a digitális fizetések valódi gyakorlati alkalmazásának minél szélesebb körben való lehetővé tétele érdekében.

**Dr. Illés Blanka**
**Nádas & Mayer Ügyvédi Iroda**
**mayer@nadas-mayer.hu**

# Microsoft Exchange 2000

## *Nyilvános mappák*

Az Exchange 2000-ben a nyilvános mappák területén fontos változtatások történtek. Az Exchange és az Active Directory összefonódásaként a jogosultságok kiosztása, és használata módosult. Ahogy az adatok elérése is sokféle módon történhet, a jogosultságok állítása is bonyolultabb. Miután megismerkedtünk kicsit a nyilvános mappákkal, belevetjük magunkat a jogosultságok dzsungelébe.

A nyilvános mappák rugalmas keretet nyújtanak a csoportos munkavégzéshez. Nem csak leveleket, de névjegyeket, feladatokat, gyakorlatilag bármilyen állományt tárolhatunk a nyilvános mappákban, ez teszi őket univerzálissá. A postaládákat tartalmazó adatbázissal ellentétben a nyilvános mappák tartalmát már alapértelmezésben is mindenki olvashatja, sőt a mappákban írás joga is van mindenkinek. A nyilvános mappáknak éppúgy lehet e-mail címe, mint a felhasználóknak, vagy csoportoknak; lehet nyilvános mappák nevében levelet is írni.

A legegyszerűbb Exchange 2000 is képes több nyilvános mappákat tartalmazó adatbázist kezelni, azonban ezek az adatbázisok nem egyenrangúak, mert csupán a telepítésnél létrehozott nyilvános mappákat tartalmazó adatbázis érhető el Outlookból MAPI protokollon keresztül. Ezenkívül ez az első nyilvános mappa-struktúra az egész szervezet szintjén azonos! *(Ne feledjük, egy Active Directory erdőben egy Exchange szervezet (Organization) létezhet.)*

A nyilvános mappákat számos protokollon keresztül érhetjük el:

- MAPI-val Outlookból – *csak a telepítésnél létrejött nyilvános mappákat érhetjük el vele!*
- HTTP-n keresztül Internet-böngészővel – URL címzéssel
- Közvetlenül a fájlrendszerből – mint egy hálózaton megosztott könyvtár tartalmát láthatjuk, használhatjuk. Az alkalmazásokból megnyithatjuk a nyilvános mappákban tárolt dokumentumokat, közvetlenül menthetjük is őket – mindezt az EXIFS teszi lehetővé.

A nyilvános mappák tartalmát lehet szinkronizálni más szervezetben, vagy azon kívül eső Exchange kiszolgálókra is, így több telephely esetén is könnyedén megoszthatunk információt.

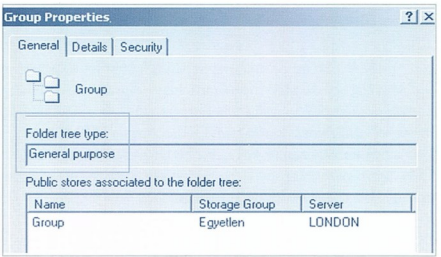
### Nyilvános mappa adatbázisok és mappa-struktúrák

A nyilvános mappákat is adatbázisban tárolja az Exchange, csakúgy, mint a postaládákat. Minden nyilvános mappákat tartalmazó adatbázishoz tartozik egy mappa-hierarchia *(Folder Tree)*, amely az adatbázisban levő hierarchiát reprezentálja. Az adatbázisok és a mappa-hierarchiák szorosan összefüggnek. Egy adatbázishoz pontosan egy mappa-hierarchia tartozik, míg egy mappa-hierarchia tartozhat különböző Exchange kiszolgálókon levő nyilvános mappa adatbázishoz is!

Kétféle mappa-hierarchia létezik az Exchange 2000-ben. A telepítéskor létrejövő nyilvános mappa adatbázishoz - a „Public Folder Store-hoz" – tartozó „All Public Folders" hierarchia például - ha akarjuk, ha nem – a szervezet összes Exchange kiszolgálóján azonos. Mit jelent ez? A szervezetben bárhol vagyunk is, az Outlookban a nyilvános mappák hierarchiája azonos lesz. Persze a jogosultságok beállításával elbújtathatók bizonyos mappák, nem kell feltétlenül látnia mindenkinek mindent. Az Exchange System Managerből tudjuk megnézni, hogy melyik hierarchia melyik kiszolgálón levő adatbázishoz tartozik.


*Az elsőként létrejött nyilvános mappa adatbázisának típusa is azt sejteti, hogy Outlookból való használatra szánták*

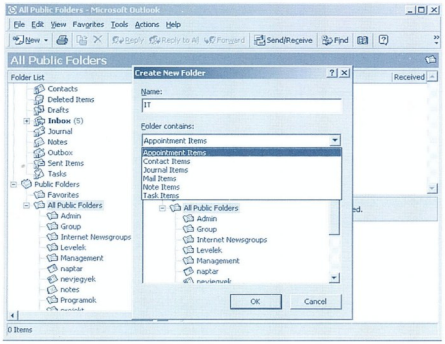
A telepítés után létrehozott nyilvános mappa adatbázisokat nem láthatjuk az Outlookból.




*Általános célú nyilvános mappa hierarchia és a hozzá kapcsolódó adatbázis*

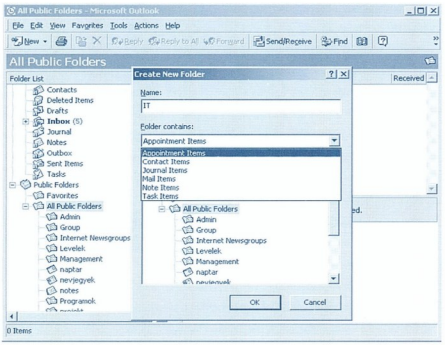
A pluszban létrehozott adatbázisok tartalma HTTP-n keresztül, a fájlrendszerből, megosztott könyvtárként és NNTP kliens használatával érhető el. Míg a MAPI típusú hierarchia az Exchange szervezet minden kiszolgálóján azonos, addig a pluszban létrehozottak csupán azon a kiszolgálón léteznek, ahol létrehozunk, illetve ahová mi magunk replikáljuk őket.

Vegyük sorra, hányféleképpen hozhatunk létre nyilvános mappákat.Talán a legegyszerűbb, leggyakrabban használt módszer, amikor az Outlookból hozunk létre nyilvános mappát. A képen látható, hogy nem csupán leveleket rakhatunk egy nyilvános mappába, hanem létrehozhatunk közös naptárat *(Appointment Items)*, vagy közös névjegytárat *(Contact Items)* is, de feladatokat *(Task Items)* tartalmazó mappákat is.



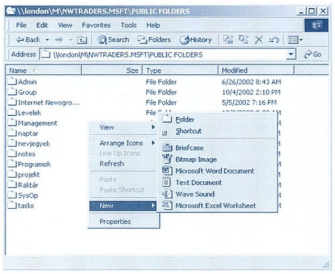

### *Nyilvános mappa létrehozása Outlookkal*

Ugyancsak bármilyen típusú mappát létre tudunk hozni HTTP-n keresztül is:




### *Nyilvános mappa létrehozása HTTP-n keresztül*

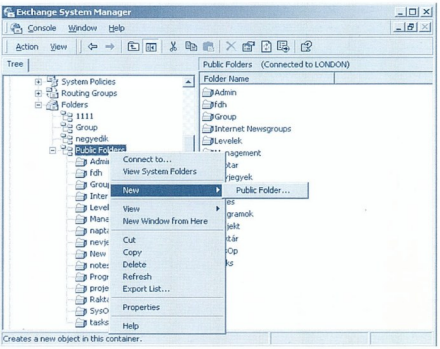
Az alábbi képen látható, hogy létre tudunk hozni újabb mappát, mintha azt egy fájlkiszolgálón tennénk.




### *Mappa létrehozása a fájlkezelőből*

Ebben az esetben azonban csak levéltípusú (Mail Items) mappát tudunk létrehozni, és mindent úgy látunk, mintha könyvtár és fájl lenne. Egész viccesen tud kinézni egy névjegy, vagy egy naptárbejegyzés, ha fájlkezelőből nézzük!
Megjegyzés! Ha az utóbbi módon rakunk be valamit (mondjuk egy ártatlan dokumentumot) egy mappába, majd ugyanazt a mappát megnyitjuk Outlookkal, azt tapasztalhatjuk, hogy nincs feladója *(üres a From mező)* az állománynak.
Létrehozni ugyan sokféleképpen lehet nyilvános mappát, a belső tulajdonságait állítani már csak Outlookból lehet, kivéve a hozzáférési jogosultságokat, amelyeket a fájlkezelőből is állíthatunk, de nem mindig szabad állítanunk!

Az Exchange előző verziójához képest – ahol nyilvános mappát létrehozni kizárólag Outlookból lehetett – láthattuk, hányféle módon tudjuk a nyilvános mappákat létrehozni, kezelni. Az előbbi módszerekkel bármely felhasználó, a megfelelő tudás és jogosultság birtokában képes létrehozni nyilvános mappát. A rendszergazdáknak van még egy lehetőségük, ahonnan létre tudnak hozni nyilvános mappát. Ez pedig az Exchange System Manager.

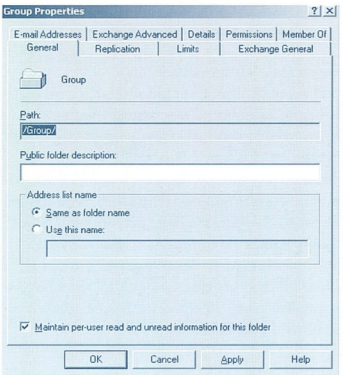



### *Nyilvános mappa létrehozása System Managerből*

Miután ennyi helyről létrehoztunk mappákat, érdemes lenne megnézni, mit is lehet beállítani egy-egy mappával kapcsolatosan.
Válasszuk szét a felhasználói és rendszergazdai szintet!
Ha az adott mappa tulajdonosa nem maga a felhasználó, csak tájékozódhat a beállításokról – például a mappa méretéről, a jogosultságokról.
Aki létrehozza a nyilvános mappát, az lesz a tulajdonosa. Ha ez egy egyszerű felhasználó a megfelelő jogosultságok-

kal, ő csaknem bármit megtehet a mappával. Jogosultságokat állíthat, szabályozhatja a mappa tartalmát moderátorok segítségével, űrlapokat rendelhet a mappához, szabályokat hozhat létre a mappába érkező dokumentumokra vonatkozóan.

A rendszergazda sokkal több mappabeállítást tud szabályozni. Például a mappák replikációját más kiszolgálókra, vagy az e-mail cím beállításokat. Nézzük meg ezeket közelebbről!




***Nyilvános mappák tulajdonságlapjai System Managerből nézve***

A **Replication** lapon állítható minden, ami a nyilvános mappa kiszolgálók közti szinkronizálást érinti. Később részletesen foglalkozunk majd vele.

A **Limits** lapon szabályozható a mappa mérete, valamint lehetőségünk van felülbírálni két beállítást, amik egyébként az adatbázis tulajdonságlapjai közt állíthatók. Az egyik, hogy mennyi ideig tartsa meg a mappából törölt elemeket a kiszolgáló. A másik pedig az „Age limit", az állományok maximális idejét határozza meg. A beállított idő letelte után a fájlok automatikusan törlődnek a mappából.

Az **Exchange General** lapon található **Delivery Options** gomb mögött állítható, hogy ki tudjon levelet küldeni a nyilvános mappa nevében.

Az **Exchange Advanced** tulajdonságlapon lehet beállítani, hogy az e-mail címmel rendelkező nyilvános mappák látszódjanak-e a címlistában, vagy sem.

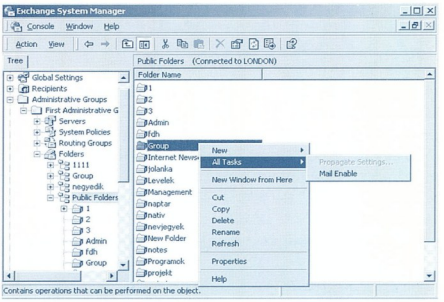
A **Permissions** lap a jogosultságok beállításait takarja, amiről hamarosan bővebben lesz szó.

A System Managerből szabályozható, hogy mely mappáknak legyen e-mail címe. Attól függően, hogy az Exchange mixed vagy natív módban van *(Figyelem: nem a Windows 2000 mixed módjáról van szó, hanem az Exchange 2000-ről!)*, a nyilvános mappák e-mail címkiosztása eltérően működik.

Ha az Exchange mixed módban van, az első nyilvános mappa adatbázisban *(Default Public Store)* létrehozott mappák automatikusan e-mail címmel ellátottak lesznek, nem is lehet elvenni az e-mail címet, de ilyenkor alapértelmezésben a nyilvános mappák nem látszódnak a címlistákban. Exchange mixed módban, ha a nyilvános mappa adatbázist utólag hoztuk létre, az abban létrehozott nyilvános mappákhoz nem keletkezik automatikusan e-mail cím.

Ha az Exchange natív módban van, a nyilvános mappáknak csak akkor képződik e-mail címe, ha a rendszergazda ezt külön beállítja. Az e-mail cím automatikusan bekerül a címlistába is.

A nyilvános mappákhoz e-mail címet létrehozni a System Managerből lehet, ahogy az alábbi képen is látszik.




***E-mail cím létrehozása nyilvános mappához***

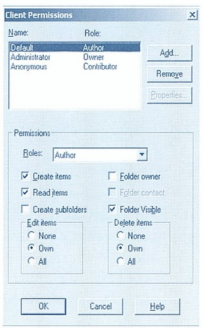
A **„Mail Enable"**-t választva a menüből, létrejön az Active Directoryban egy objektum a nyilvános mappához. Az összes nyilvános mappához tartozó AD objektum az Active Directory Users and Computers MMC konzolban a **Microsoft Exchange System Objects** konténerben látható.

### Jogosultságok

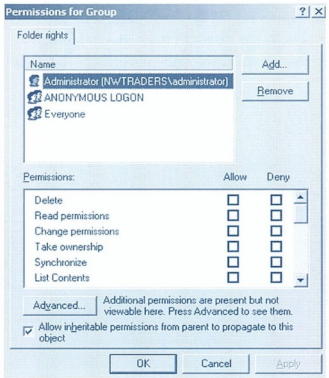
A hozzáférési jogosultságokat több helyről, többféleképpen szabályozhatjuk. A nyilvános mappákhoz rendelt jogosultságok tulajdonképp Windows 2000 jogosultságok, hisz az Exchange az Active Directoryra támaszkodik. A Windows 2000 kezeli az Exchange-specifikus jogosultságokat is *(például ki tud létrehozni a legmagasabb szinten nyilvános mappát)*.

Jogosultságokat az Active Directoryban létező felhasználókhoz, csoportokhoz rendelhetünk. Az Exchange-ben adott jogosultságok öröklődnek. Jogokat külön vagy egyben is adhatunk a mappákra és a bennük levő elemekre.

A Nyilvános mappák tulajdonságlapjai közül a Permissions lapon három nyomógombot találhatunk; mindegyik mögött más jogokat oszthatunk. A legfelső a **Client Permissions** gomb, a hatására feljövő ablakból szabályozható a kliensekről a mappahozzáférés. Az alábbi kép baloldalán látható az az ablak, ami ilyenkor feljön. Itt nem közvetlenül Windows 2000 jogosultságokat látunk, hanem Exchange nyelvezetűt, ami volt már az Exchange korábbi verzióiban is. Különböző szerepekhez, különböző előre csomagolt jogosultságokat rendelhetünk egyszerűen, gyorsan. A lenti képen egy mappa alapértelmezett jogosultságai láthatóak. A „default"-hoz rendelt jogok tulajdonképp az Everyone csoportot jelenti. Ha a Client Permissions gombra úgy kattintunk, hogy közben nyomva tartjuk a CTRL billentyűt, az alábbi képen jobboldalon látható Windows 2000-es jogosultságosztó ablakot kapjuk.




***Mappatulajdonságok I.***




*A mappajogosultságok – Client Permissions – két formája*

Amikor jogokat osztunk, az előre definiált szerepek szerint a kiosztott jogosultságok automatikusan Windows 2000 „nyelvre fordítódnak".

Figyelem! Egy MAPI-val használatos nyilvános mappa esetén tilos a Windows 2000 jogosultságokat közvetlenül állítani! Megtehetjük, de csak egyszer, ugyanis ha közvetlen állítunk Windows 2000 jogosultságokat, az első alkalom után az ACL rögzül és elmozdulni róla nem egyszerű.

Demonstrációként megváltoztattam egy MAPI-s nyilvános mappán a Windows 2000 jogosultságokat. Először minden figyelmeztetés nélkül hagyta is. Mikor újra próbálkoztam, a következő hibaüzenetet kaptam:

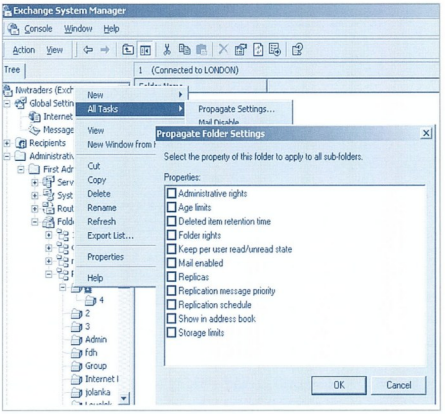

*Hibaüzenet jogosultságok változtatásakor*

A Microsoft tudásbázis Q270905-ös cikke pontosan erről az esetről, és a megoldásról szól. Jó bogarászást!

Még mindig a nyilvános mappák Permissions tulajdonságlapjánál tartunk, nézzük a második gombot. Ez a **Directory Rights** feliratú. Ha a nyilvános mappa nem rendelkezik e-mail címmel, ez a gomb szürke. E-mail címmel rendelkező nyilvános mappák esetén van csupán értelme, mert rákattintva állíthatók az adott mappához tartozó Active Directoryban tárolt objektumhoz rendelt jogosultságok.

A harmadik gomb az **Administrative Rights** amely mögött a nyilvános mappákhoz rendelhető rendszergazdai jogosultságok szabályozhatók, például hogy ki állíthatja a replikációt, vagy kvótákat ki szabályozhatja, ki változtathatja a mappához rendelt kliensoldali jogokat stb. Mindezek segítségével nagyon finoman szabályozható minden mappa és tartalmuk jogosultsága. Van néhány speciális nyilvános mappákkal kapcsolatos jogosultság, amely a legmagasabb szintről öröklődik. Ilyenek a rendszergazdai jogosultságok, valamint az is, hogy ki tud a mappahierarchia gyökerében mappát létrehozni *(pontosan a Create Top Level Public Folder jogról van szó)*. Ezek többnyire az Organization szintről indulnak, az Adminisztratív csoporton *(Administrative Group)* át a nyilvános mappa hierarchián *(Public Folder Tree)* keresztül jutnak le a mappákhoz.

A beállított jogosultságok automatikusan öröklődnek lefelé a struktúrában, azonban a mappák csak létrehozáskor öröklik a beállításokat a szülőtől, a későbbi változások már nem jutnak le. Cserében nemcsak a jogosultságok, hanem egyéb beállítások is terjeszthetők a mappahierarchiákban. A tulajdonságokat terjeszteni az adott mappán állva a menüből All tasks –Propagate Settings parancs segítségével lehet:




*A mappákon örököltethető beállítások*

Csak néhány a sok közül:

- Jogosultságok – A kliensoldali és a rendszergazdai jogok egyaránt örökíthetők
- Replikációs beállítások
- Kvóták – az élettartamra, a méretekre, és a törölt elemekre vonatkozó beállítások
- Active Directoryval kapcsolatos beállítások: látszódjon-e a címlistában, legyen-e e-mail címe stb.

A következő részben a nyilvános mappák replikációjával ismerkedünk meg.

Dorner Csilla
dorner.csilla@netacademia.net

# XMLgessünk 16.

## *SQL XML 3.0 - I. rész*

Az SQL Server 2000 számos beépített XML- szolgáltatással rendelkezik, viszont a termék megjelenése óta több mint három év telt el, és az XML- technológiák pont az utóbbi időben indultak szédületes fejlődésnek. Az SQL Server emiatt újabb és újabb XML- szolgáltatásokat kapott, melyeket külön csomagban lehet letölteni. Ez az SQLXML programcsomag.

Nézzük meg miért jó nekünk e kiegészítés, mire tudjuk bevetni a mindennapi munkában? Az első részben a menedzselt SQLXML osztályok használatával foglalkozunk, a következő számban pedig az SQLXML Webszolgáltatásokat és a DataSet módosítások visszaírását vesszük górcső alá, egy komplett alkalmazáspéldán keresztül.

### Előzmények

Korábban *(tech.net nem tudom mikori szám)* már beszéltünk az SQLXML szolgáltatásokról, akkor a 2-es verziót alapul véve. Ismétlésképpen tekintsük át, milyen alapszolgáltatások laknak a programcsomagban.

Az SQLXML programcsomag célja az SQL Server adatbázisaiban, tábláiban tárolt relációs adatok elérése és manipulálása XML adatként. Azaz egy kétirányú interfészt definiálnak az SQL Server és az XML világ közé.

Az átjárásnak többféle módját is létrehozták. Ezek egy része közvetlenül az SQL OLEDB providerben van megvalósítva, így közvetlenül is kommunikálhatunk XML adatokkal az SQL Server felé.

Más esetben az adatbázisokat publikálhatunk az IIS segítségével is, így HTTP-n keresztül is elérhetők az adatbázisok. Ezzel a lehetőséggel a hivatkozott SQLXML cikkben már foglalkoztam, így most nem fejtem ki bővebben.

### SQLXML menedzselt osztályok

Ha .NET alkalmazásokból szeretnénk kihasználni az SQLXML szolgáltatásokat, akkor az SQLXML menedzselt osztályokat kell használnunk.

Az osztályokat tartalmazó assembly az SQLXML telepítő **[2]** futtatása után a Global Assembly Cache-ben található, Microsoft.Data.SqlXml néven, 3.0.1523.0 verzióval.

Fontos, hogy az XSD sémákkal kapcsolatos szolgáltatások helyes működéséhez az MSXML 4-et is fel kell telepíteni a gépre **[3]**.

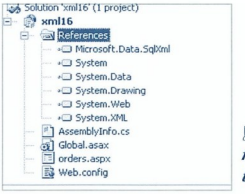
Az ADO.NET SQL osztályokhoz hasonlóan elnevezett főbb objektumok a következők: SqlXmlCommand, SqlXmlParameter és SqlXmlAdapter.

Az SqlXmlCommand segítségével lehet végrehajtani XML kimenetű SQL parancsokat, tárolt eljárásokat, XML nézeteket és sablonokat *(ez utóbbi kettőről a korábbi SQLXML cikkben olvashatnak)*. Azaz olyan mint az ADO.NET-es SqlCommand, csak nem „közönséges", hanem XML kimenetű parancsokra van specializálva.

Az SqlXmlParameter a parancsokhoz kapcsolódó paraméterek kezeléséhez fog segítséget nyújtani, teljesen analóg módon az SqlParameter osztályhoz, csak ez XML sablonokhoz és nézetekhez is tud paramétereket szállítani.

Az SqlXmlAdapter közönséges ADO.NET DataSet feltöltésére és a módosítások DiffGram formátumban történő visszatöltésére van kialakítva.

A tesztelést egy webprojektben fogjuk végrehajtani *(1. ábra)*, mert az XML kimeneteket Internet Explorerben lehet kényelmesen elemezni. A kódokat a teszt weblapok Page_Load eseménykezelőjében írjuk meg, azaz a lap elindulásakor generáltatunk valamiféle kimenetet.




***A teszt webprojekt referenciával a Microsoft.Data.Xml assemblyre***

Nézzük meg az SqlXmlCommand inicializálását!

```
string connString =
   @"Provider=SQLOLEDB;
     Server=(local);
     Database=Northwind;
     Integrated Security=SSPI";
SqlXmlCommand cmd = new SqlXmlCommand(connString);
```

Valójában itt már le is bukott a menedzselt, .NET-es SqlXmlCommand objektum, aki a COM-os SQLOLEDB providert használja, hisz a konstruktorban egy közönséges OLEDB connection stringet használ. Ha egy picit belenézünk az assembly belsejébe, nyilvánvalóvá válik, hogy a COM-os ADODB.Stream és egyéb objektumok köré írtak csomagolóosztályokat, és ezek segítségével hívják meg az OLEDB XML szolgáltatásokat. Valószínűleg csak a következő SQL Server verzióban várható közvetlen menedzselt osztályokkal történő XML elérés. Sebaj, amíg működik a COM Interop ez csak esztétikai *(és némi teljesítmény)* probléma.

A lekérdezések kimenete általában nem tartalmaz xml gyökérelemet, hisz így könnyű egymás után fűzni több parancs kimenetét. Ha viszont egyetlen lekérdezés kimenetét kell feldolgozni, célszerű kérni egy mesterséges gyökérelemet az adathalmaz köré, így DOM és egyéb xml technológiákkal feldolgozhatóvá válik az eredményhalmaz. A mesterséges gyökérelem kijelölése nagyon egyszerű:

```
cmd.RootTag = " orderlist";
```

Tesztpéldánkban egy nyomtatásra szánt riportot szeretnénk készíteni megrendelésekről és azok részletező tételeiről. ASP.NET Data bound vezérlőkkel *(DataGrid és társaik)* és User Controlokkal persze megoldható a dolog, de most egy alternatív, XML-XSLT alapú megoldást tekintünk át. Mindkettővel elérhetjük ugyanazt a célt, de ha bonyolultabb programozott logikára is szükség van, általában egyszerűbbek az ASP.NET-es megoldások, míg nem túl komplex és nem interaktív riportok esetén sokszor egyszerűbb az XML-XSLT páros használata. Különösen sok táblából JOIN-okkal összevadászott lekérdezések

esetén az XSLT-s feldolgozás sokkal egyszerűbb tud lenni. Kiindulásként egy XML kimenetű SQL parancsot fogunk végrehajtani, ebben benne lesz minden szükséges információ a riport elkészítéséhez:

```
cmd.CommandType = SqlXmlCommandType.Sql;
cmd.CommandText= @"
SELECT [order].OrderID,
  OrderDate,
  ProductName,
  detail.UnitPrice,
  Quantity,
  Discount
FROM
  [Order Details] detail
INNER JOIN
  Orders [order]
ON
  detail.OrderID = [order].OrderID
INNER JOIN
  Products product
ON
  detail.ProductID = product.ProductID
FOR XML AUTO
";
Response.ClearContent();
cmd.ExecuteToStream(Response.OutputStream);
```

Az XML kimenetű SQL parancsot az ExecuteToStream() metódussal hajtjuk végre. Ő a megadott System.IO.Stream példányba képes az XML kimenetet eljuttatni. A Response.OutputStream az ASP.NET oldal kimeneti adatfolyama, amelybe többek között az ASP-ből is jól ismert Response.Write() is ír. Az oldal kimenete egyelőre egy nyers XML adatfolyam:


### *XML kimenetű lekérdezés a böngészőben*

Tegyük fel, hogy ezt az adatfolyamot nem akarjuk megformázni, hanem így, ebben a formában akarjuk publikálni, mert például a számlázóprogramnak szüksége van a megrendelések adataira. Ha ez a program az Interneten keresztül kapcsolódik a rendszerünkhöz, igény lehet a titkosított elérésre. Ennek legegyszerűbb módja, ha beállítjuk az SSL használatát a webkiszolgálón, és https-en keresztül érjük el az orders.aspx-et.

Másképp megközelítve saját titkosítást alkalmazhatunk az adatfolyamra. Na nem saját algoritmust, nem vagyunk sarlatánok, és nem hisszük a betűfelcserélős vagy xor-os módszerünkről, hogy megfejthetetlen. A .NET keretrendszer számos nagyon egyszerűen használható titkosítási módszert ismer. Esetünkben a két oldal ismeri egymást, így egy közönséges szimmetrikus titkosítás teljesen megfelelő lehet. Azért fontos, hogy ismerik egymást, mert így biztonságosan *(pl. személyes találkozással, titkosított levélben :)* megegyezhetnek a közösen használt szimmetrikus kulcsban, amivel a titkosítás és a dekódolás is történik.

Például Rijndael algoritmussal a következőképpen titkosíthatjuk az xml kimenetünket *(ordersenc.aspx)*:

```
Stream xmlContent = cmd.ExecuteStream();
//Csak az első 16 karaktert használjuk,
//ez a maximális kulcshossz a Rijndaelnél.
string keyText = "Ez lesz a titkos kulcs";
string ivText = "Indulási vektor érték";
//Fontos, hogy az encoding minden karaktert
//pontosan egy bájtra képezzen le.
Encoding en = Encoding.GetEncoding("ISO-8859-2");
byte[] key; byte[] iv;
key = en.GetBytes(keyText.Substring(0, 16));
iv = en.GetBytes(ivText.Substring(0, 16));
//Rijndael algoritmust használunk
RijndaelManaged Crypto = new RijndaelManaged();
//Ez a titkosító objektum
ICryptoTransform tr =
  Crypto.CreateEncryptor(key, iv);
//Amit benyomunk az origStream-be, az kijön
//titkosítva a Response.OutputStream-be.
CryptoStream origStream =
  new CryptoStream(Response.OutputStream,
    tr, CryptoStreamMode.Write);
int buffSize = 8192;
byte[] buff = new byte[buffSize];
int readed;
while ((readed =
  xmlContent.Read(buff, 0, buffSize)) != 0) {
  //Átvödrözzük az xml tartalmat.
  origStream.Write(buff, 0, readed);
}
origStream.Close();
```

Láthatóan most az SqlXmlCommand.ExecuteStream() metódust használtam, mert egy új Streamre volt szükség a feldolgozáshoz. A kimenet nem embernek való csúnya, titkosított bájthalmaz.

A fogadóoldali alkalmazásnak ismerni kell a kulcsot *(key)* és az Initialization Vector *(iv)* értékét, hogy ki tudja bontani a kapott tartalmat.

Tegyük fel, hogy a fogadó egy konzolalkalmazás, amely szeretné letölteni és kicsomagolni a tartalmat. A feldolgozás egy lehetséges módja látható az következő kódblokkban.

```
string uri =
  "http://localhost/xml16/ordersenc.aspx";
HttpWebRequest req =
  (HttpWebRequest)WebRequest.Create(uri);
//A belépett felhasználó nevében
//történik a hozzáférés
req.Credentials =
  CredentialCache.DefaultCredentials;

req.Method = "GET"; //HTTP metódus
WebResponse resp = req.GetResponse();
Stream xmlEncrypted = resp.GetResponseStream();
int readed; int buffLen = 8192;
byte[] buff = new byte[buffLen];
//HACK HACK HACK
MemoryStream ms = new
 MemoryStream();
 while ((readed =
  xmlEncrypted.Read(buff, 0,buffLen)) > 0) {
  ms.Write(buff, 0, readed);
}
ms.Position = 0;
string keyText = "Ez lesz a titkos kulcs";
string ivText = "Indulási vektor érték";
Encoding en = Encoding.GetEncoding("ISO-8859-2");
byte[] key; byte[] iv;
key = en.GetBytes(keyText.Substring(0, 8));
iv = en.GetBytes(ivText.Substring(0, 8));
RijndaelManaged crypto = new RijndaelManaged();
ICryptoTransform tr =
  crypto.CreateDecryptor(key, iv);
CryptoStream decrStream =
  new CryptoStream(ms,
  tr, CryptoStreamMode.Read);
```

```
StreamReader reader = new StreamReader(decrStream);
Console.WriteLine(reader.ReadToEnd());
reader.Close();
xmlEncrypted.Close();
decrStream.Close();
resp.Close();
```

A titkosított Stream feldolgozása nem teljesen fájdalommentes. Elvileg az xmlEncrypted NetworkStream közvetlenül beleírányítható a CryptoStream konstruktorba, azaz az

```
CryptoStream decrStream =
  new CryptoStream(ms,
  tr, CryptoStreamMode.Read);
```

sor helyett elvileg lehetne használni az

```
CryptoStream decrStream =
  new CryptoStream(xmlEncrypted,
  tr, CryptoStreamMode.Read);
```

sort. A valóságban azonban egy Exception emelkedik fel, mintha túlolvasna a CryptoStream a forrás Streamen. Ez a hiba leginkább akkor jelentkezik, ha a forráseweboldal és a letöltőalkalmazás ugyanazon a gépen helyezkedik el.

Tankó Péter barátom vette a fáradságot és kinyomozta, hogy a probléma oka az, hogy az ASP.NET vagy az IIS nem jól formázza meg az elküldött tartalmat. A titkosított tartalom Chunked Transfer Encoding **[4]** kódolással jut el az ügyfélprogramhoz, amelyben a kiszolgáló a hosszú tartalmat darabjaira vágja, és a darabokat megjelöli hossz és egyéb terminálo karakterekkel. Sajnos azonban az utolsó darab lezáró karaktereit elhagyja. Az ügyfélprogram - a kiszolgálóval ellentétben - jól ismeri a HTTP szabványt, és keresi a lezáró karaktereket, ezért olvas túl a forrás NetworkStreamen.

A probléma megkerülésére a HACK részben a letöltött Stream tartalmát először áttöltöttem egy MemoryStreambe, majd az dekódoltam. Így már hibamentes lett a dekódolás:


*Az ügyfélprogram kimenete a dekódolt xml adatfolyammal*

Egy ennél elegánsabb megoldás arra épít, hogy a HTTP 1.0 szabványban még nem létezett a Chunked Transfer Encoding, így ha az ügyfélnek megmondjuk, hogy HTTP1.0 protokollal kérje le a kiszolgálótól a tartalmat, a kiszolgáló nem fog Chunked Transfer Encodingot használni, így nincs módja a hibát elkövetni. Ehhez csak egy sorral kell bővíteni az ügyfelet a kapcsolatfelvétel előtt:

```
req.ProtocolVersion = HttpVersion.Version10;
```

Talán a példából látszik, a kézi titkosítás nem teljesen triviális, valamint a kulcs értéke bele van kódolva az alkalmazásba, így nagyon könnyen kinyerhető. Például az Anakrino nevű C# decompilerben így néz ki a kulcsok értékadása a szamlazo.exe-ből visszafejtve *(eh :)*:

```
local8 = "Ez lesz a titkos kulcs";
local9 = "Indulási vektor érték";
```

Emiatt ha a követelmények megengedik, sokkal egyszerűbb az SSL beállítása az IIS-en, mint a kézi titkosítás. Ott a nyilvános kulcsú titkosításon alapuló kulcscsere-algoritmus leveszi a vállunkról kulcselosztás problémáját, és gyakorlatilag nulla kódolással érünk célt.

### Formázott kimenetek előállítása

Ha helyben, a webkiszolgálón szeretnénk feldolgozni a kapott XML tartalmat, az egyik legkézenfekvőbb lehetőség az XSLT-vel történő transzformáció. Olyannyira, hogy az SqlXmlCommand objektumnak eleve készítettek egy XslPath jellemzőt. Ebben meg kell adni egy XSLT fájl elérési útját, amelyet az SqlXmlCommand az xml kimenetre alkalmaz:

```
cmd.XslPath = Server.MapPath("ord.xslt");
cmd.ExecuteToStream(Response.OutputStream);
```

A korábbi xml kimenethez például az alábbi XSLT-t használhatnánk *(ordersformatted.aspx)*:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output omit-xml-declaration="yes" />
<xsl:template match="/">
  <html>
   <head>
    <title>Megrendelések listája</title>
    <link rel="stylesheet" type="text/css"
&     href="css/orders.css" />
   </head>
   <table class="list">
    <tr>
     <td>
      <xsl:apply-templates
&      select="orderlist/order">
       <xsl:sort select="@OrderDate" />
      </xsl:apply-templates>
     </td>
    </tr>
   </table>
  </html>
 </xsl:template>
 <xsl:template match="order">
  <table class="order" width="100%">
   <tr class="orderheaderrow">
    <td>Sorszám:
    <xsl:value-of select="@OrderID" /></td>
    <td>Dátum:
    <xsl:value-of select="translate(substring-
&    before(@OrderDate,'T'), '-', '.')" />.</td>
   </tr>
   <tr>
    <td colspan="2">
     <table class="product">
      <xsl:apply-templates select="product">
       <xsl:sort select="@ProductName" />
      </xsl:apply-templates>
     </table>
    </td>
   </tr>
  </table>
 </xsl:template>
 <xsl:template match="product">
  <tr class="prodrow">
   <td class="prodnamecell">
    <xsl:value-of select="@ProductName" />
   </td>
   <td><xsl:value-of
&        select="detail/@Quantity"/> db, </td>
   <td><xsl:value-of
&        select="detail/@UnitPrice"/> $/db</td>
  </tr>
 </xsl:template>
</xsl:stylesheet>
```

A némi költői túlzással pofásnak is mondható eredmény. Példánkban az XML kimenetet maga az SQL Server állította elő, az OLEDB provider csak fogadja és átadja nekünk. Ez azt jelenti, hogy a kacsacsőrök vígan repülnek át az SQL Server és az IIS között a hálózaton. Nos, ez nem egy világbajnok hatékonyságú feldolgozás. Ennél sokkal hatékonyabb tud lenni, ha az SQL Server az eredményhalmazt a maga tömör, bináris formátumában adja vissza, és az OLEDB provider alakítja azt át xml-lé. E módszer előnye, hogy átadja a terhelés egy részét az SQL Serverről a webkiszolgálóra, illetve csökkenti a két réteg közötti hálózati forgalmat. Ennek ellenére a módszer visszaüthet, mert egy többszörös JOIN kimente SQL eredményhalmazként olyan redundáns, annyi ismétlődő adatot tartalmaz, hogy

még az is lehet, hogy a FOR XML AUTO szószátyár kimenete is tömörebb lesz. Tehát nem csodafegyver az OLEDB providerre taksált xml konverzió, de érdemes megnézni a mikéntjét. Ismét központi objektumunk, az SqlXmlCommand egy újabb paramétere segít:

```
cmd.ClientSideXml = true;
```

Esetünkben a Client szó a webkiszolgálón futó ASP.NET lapunkra utal. A kívánt hatás eléréséhez az SQL parancsunkat is módosítani kell. A lekérdezés végén a FOR XML AUTO záradékot ki kell cserélni FOR XML NESTED-re. Ezt érzékeli az SQL OLEDB provider, kiveszi a FOR XML NESTED-et, és csak a SELECT-et küldi el az SQL Servernek. Ezt könnyű ellenőrizni az SQL Server Profilerrel. A módszer kiválóan alkalmas arra is, hogy meglévő tárolt eljárások kimenetét xmlként dolgozzuk fel. Ha a lekérdezésünk például a GetOrders nevű tárolt eljárásban van megírva *(FOR XML nélkül)*, a következő paranccsal könnyedén xml kimenetet generáltathatunk az eredeti eredményhalmazból:

```
cmd.CommandText = "EXEC GetOrders FOR XML NESTED";
```

### Paraméterezett lekérdések

A lekérdezések külső paraméterekkel kiegészítése az ADO illetve az OLEDB paraméterkezeléséhez hasonló, nem az ADO.NET-éhez *(nyilván a már tárgyalt OLEDB háttér miatt)*. Az SQL parancsok szövegében a paraméterek helyére ?-et kell írni. Ezt azt jelenti, hogy a paramétereket a pozíciójuk alapján érhetjük el, ellentétben az ADO.NET-tel, ahol a tárolt eljárásokhoz hasonlóan nevet adhatunk nekik.

Készítsünk most egy olyan weboldalt, ami a megadott dátumhatárok között kilistázza a megrendeléseket. Ehhez a felrakunk két TextBoxot és egy Submit gombot a webformunkra:

```
<form id="orderlist" method="post"
 runat="server">
 <table id="SearchTable" runat="serer">
  <tr>
   <td>Listázás kezdete</td>
   <td><asp:textbox id="StartDate"
 runat="server">1998-03-01</asp:textbox></td>
  </tr>
  <tr>
   <td>Listázás vége</td>
   <td><asp:textbox id="EndDate"
 runat="server">1998-03-03</asp:textbox></td>
  </tr>
   <tr>
    <td colspan="2"><input type="submit"
 value="Submit" id="Submit1"
 name="Submit1" runat="server"></td>
   </tr>
  </table>
</form>
```

A gomb megnyomására az alábbi kódrészletet hajtjuk végre:

```
DateTime start = DateTime.Parse(StartDate.Text);
DateTime end = DateTime.Parse(EndDate.Text);
SqlXmlParameter pStart = cmd.CreateParameter();
pStart.Value = start;
SqlXmlParameter pEnd = cmd.CreateParameter();
pEnd.Value = end;
SearchTable.Visible = false;  //már nem kell
cmd.XslPath = Server.MapPath("ord.xslt");
cmd.ExecuteToStream(Response.OutputStream);
```

A kapott szövegeket átalakítjuk dátummá, így a hibás formátum már a lekérdezés végrehajtása előtt kiderül. Persze itt elég sokat el lehet szórakozni a formátumokkal (*ld. CultureInfo, DateTimeStyles, stb.)*, illetve a dátum kinyerését érdemes try-catch blokkba rakni, és a felhasználók felé megfelelően kommunikálni a probléma okát.

Ha a dátumhatárok megvannak, SqlXmlParameter objektumokat hozunk létre az SqlXmlCommand CreateParameter() hívásával. Nyilván a CreateParameter belül rögzíti a létrehozott paraméterek sorrendjét. A tényleges értékeket a Value jellemzőn keresztül lehet elérni. Ezek után a végrehajtás teljesen azonos a korábbi példákéval. A SearchTable.Visible = false; sorral tüntetjük el a kimenetből a kereséshez használt táblázatunkat.

Az SQL parancs eleje azonos az eddigiekkel, a vége így néz ki:

```
WHERE OrderDate BETWEEN ? AND ?
FOR XML AUTO
```

A két kérdőjel helyére kerülnek be a paraméterek.

XML sablonok és XML nézetek végrehajtása során viszont használhatjuk az SqlXmlParameter objektum Name jellemzőjét is, így ott használhatunk megnevezett paramétereket is. Például nézzük az alábbi XML sablont *(XML template)*:

```
<orderlist xmlns:sql="urn:schemas-microsoft-
com:xml-sql">
 <sql:header>
  <sql:param name="StartDate"></sql:param>
  <sql:param name="EndDate"></sql:param>
 </sql:header>
 <sql:query>
  SELECT [order].OrderID,
    OrderDate,
    ProductName,
    detail.UnitPrice,
    Quantity,
    Discount
  FROM
    [Order Details] detail
  INNER JOIN
    Orders [order]
  ON
    detail.OrderID = [order].OrderID
  INNER JOIN
    Products product
  ON
    detail.ProductID = product.ProductID
  WHERE
    OrderDate BETWEEN @StartDate AND @EndDate
  FOR XML AUTO
  </sql:query>
</orderlist>
```

Ez nem más, mint a korábbi sql parancs xml fájlba csomagolva *(GetOrders.xml)*.

A sablon megadása:

```
//A gyökeret a sablon adja.
//cmd.RootTag = "orderlist";

cmd.CommandType = SqlXmlCommandType.TemplateFile;
cmd.CommandText = Server.MapPath("GetOrders.xml");
```

A paramétereket most a nevükkel azonosítjuk, és hogy ez tényleg így van, a sorrendjüket szándékosan megcseréltem:

```
SqlXmlParameter pEnd = cmd.CreateParameter();
pEnd.Name = "@EndDate";
pEnd.Value = end;

SqlXmlParameter pStart = cmd.CreateParameter();
pStart.Name = "@StartDate";
pStart.Value = start;
```

folytatjuk...

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

**A cikkben szereplő URL-ek:**

[1]: Letölthető példakódok: http://technet.netacademia.net/download/xml

[2]: SQLXML 3.0 SP1 http://msdn.microsoft.com/downloads/default.asp?URL=/downloads/sample.asp?url=/msdn-files/027/001/824/msdncompositedoc.xml

[3]: MSXML 4 : http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/001/766/msdncompositedoc.xml

[4]: HTTP1.1 RFC2616: http://www.ietf.org/rfc/rfc2616.txt

# .NET Akadémia VIII.

## Többszálú és aszinkron programozás

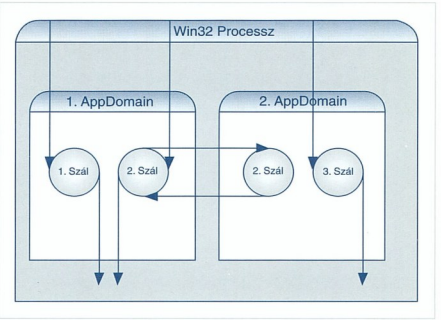
A többszálú programozás az egyik legtipikusabb felhasználói módú programozási terület, ahová csak a legelszántabb C++ programozók merészkedtek el. Windows API használatával időnként tényleg nehéz helyzetben találjuk magunkat, de a Common Language Runtime nagyon leegyszerűsíti az életünket, ha menedzselt környezetben kell többszálú programokat fejleszteni.

Sajnos azonban látni fogjuk, hogy a többszálú programozásban nem a szálak létrehozása, hanem a párhuzamosan futó szálak szinkronizációja lesz a kihívás.
Cikkünkben először áttekintjük a szálkezelés alapjait, megnézzük a szinkronizációs lehetőségeket, majd a következő részekben megismerünk egy témába vágó programozási módszert, az aszinkron programozást.

### A menedzselt futtatási környezet

Mielőtt vadul nekilátnánk szálakat létrehozni, nem árt közelebbről megismerni azt a keretet, amelyben a szálaink futni fognak: a CLR-t.
Alapkérdés: mi az a thread, magyarul szál? A szál a programfuttatás alapvető egysége, amelyhez az operációs rendszer processzoridőt rendel. Minden szálnak saját belső struktúrái vannak, amelyben az operációs rendszer elmenti a szál pillanatnyi állapotát, mielőtt átkapcsolna egy újabb szál végrehajtására. Mivel a szálak közötti végrehajtás-átkapcsolás tipikusan másodpercenként több tízszer, több százszor zajlik le, a kívülálló úgy érzékeli, hogy a szálak párhuzamosan futnak.

Az Windows32 a szálakat processzek belsejében futtatja. Egy processzben akárhány szál futhat, és a szálak közvetlenül együtt tudnak működni: közös memóriaterületen dolgoznak a processz belsejében.
Ezzel szemben különböző processzekben dolgozó szálak nem tudnak közvetlenül kommunikálni, nem látják egymás memóriaterületeit. Pont ezért létezik a processz fogalma, mely elszigeteltségi egységként funkcionál.
.NET-ben a Win32-es processzt tovább bontották, és bennük akárhány Application Domain létezhet. Az AppDomain tulajdonképpen menedzselt processznek felel meg. A menedzselt programok részére éppoly erős elszigeteltséget biztosít, mint a Win32 processz, csak sokkal kevesebb plusz memória- és processzorterheléssel.
Milyen kapcsolat van az AppDomainek és a szálak között? Általában egy AppDomainben akárhány szál futhat párhuzamosan, azonban *(és ez teljesen eltérő a processzektől)* egy AppDomainben futó szál könnyedén átléphet egy másik AppDomainbe, ott végrehajthat némi kódot, majd visszatérhet az induló Domainbe. Ilyen természetesen processzek között nem működik. Ez a helyzet akkor áll elő, ha egy AppDomain létrehoz egy új AppDomaint, abba betölt egy assemblyt, és a betöltött assemblyből létrehozott típusnak meghívja egy metódusát. Ekkor a hívás ugyan a Remoting *(lásd előző cikkünket)* felületén keresztül zajlik le, de az új AppDomain kódját akkor is a hívó domain szála fogja végrehajtani.



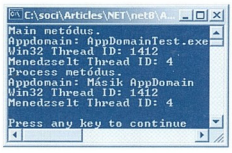

*Szálak és AppDomainek kapcsolata*

Bár az AppDomainek témája nem tartozik szorosan a cikk fő vonulatához, de az érdeklődőknek íme egy AppDomain-létrehozás, assemblybetöltés, típuslétrehozás és -hívás Remotingon keresztül. A példakód:

```
using System;
using System.Reflection;
using System.Threading;

namespace AppDomainTest {
 class MainClass {
  static void Main(string[] args) {
   Console.WriteLine("Main metódus.");
   Info.ShowInfo();

   AppDomain appDomain;
   appDomain =
    AppDomain.CreateDomain("Másik AppDomain");
   string assName =
    Assembly.GetExecutingAssembly().FullName;
   WorkerDomain dom;
   dom = (WorkerDomain)
   appDomain.CreateInstanceAndUnwrap(
   assName, "AppDomainTest.WorkerDomain");
   dom.Process();
   Console.ReadLine();
  }
 }
```

```
/// <summary>
/// MarshalByRefObject, így remotingon keresztül
/// elérhető.
/// </summary>
public class WorkerDomain : MarshalByRefObject {
  public void Process() {
    Console.WriteLine("Process metódus.");
    Info.ShowInfo();
  }
}

public class Info {
  public static void ShowInfo() {
    Console.WriteLine("Appdomain: {0}",
      AppDomain.CurrentDomain.FriendlyName);
    Console.WriteLine("Win32 Thread ID: {0}",
      AppDomain.GetCurrentThreadId());
    Console.WriteLine("Menedzselt Thread ID: {0}",
      Thread.CurrentThread.GetHashCode());
  }
}
```

A kimenet:




Látható, hogy a hívás során ugyanaz a szál először a kiinduló AppDomainben futtatott kódot, majd az újonnan létrehozott Domainben *(azonos a Thread ID)*.

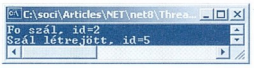
### A Thread osztály

Alapvetően a szál egy operációsrendszer-konstrukció, az hozza létre, allokál neki processzort és semmisíti meg. Emiatt a System.Threading.Thread osztály csak egy burkolótípus a Win32-es Thread API köré, nem sok újdonságot ad hozzá. Minden egyes CLR-ben létrehozott vagy a CLR-ben futó alkalmazásba kívülről belehívó szál köré kiépül egy Thread objektumpéldány, amely segítségével egyszerűen kezelhetjük a szálakat.

Első lépésként nézzük meg egy új szál létrehozását és elindítását:

```
using System;
using System.Threading;

class Tester {
 [STAThread]
 static void Main(string[] args) {
  Console.WriteLine("Fő szál, id={0}",
    Thread.CurrentThread.GetHashCode());
  Thread t1 = new Thread(
   new ThreadStart(T1Metodus));
  t1.Start();
 }
 static void T1Metodus() {
  Console.WriteLine("Szál létrejött,id={0}",
  Thread.CurrentThread.GetHashCode());
 }
}
```

A Thread konstruktor egy ThreadStart delegate példányt vár paraméterül, ez lesz az új szál belépési pontja. A konstruktor lefutása után a szál még nem fut, Unstarted állapotban van. Az elindítást a Start() metódus hívás váltja ki:



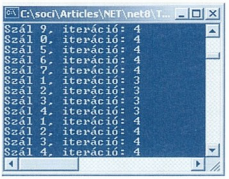

Egy szál pillanatnyi állapotát lekérdezhetjük a ThreadState jellemzőn keresztül:

```
Console.WriteLine(t1.ThreadState);
```

A különböző állapotokat a ThreadState nevű felsorolás típusban szedték össze. Az Unstarted állapot közvetlenül a létrehozás után jön létre, mígnem meghívjuk a Start() metódust. Ekkor a szál Running állapotba kerül. Ha a szál megállt *(például, mert véget ért a szál által végrehajtott metódus)*, akkor a Stopped állapotban találhatjuk. Ezen felül további állapotok is léteznek, amelyeket hamarosan számbaveszünk.
A ThreadStartban megadott delegate példányszintű metódusra is mutathat, ami kitűnő lehetőséget ad indulási paraméterek átvitelére, illetve sok munkaszál létrehozására:

```
ArrayList threads = new ArrayList();
for(int i=0; i<10; i++) {
  Szalhuzo sz = new Szalhuzo();
  sz.PeldanyNo = i;
  Thread t = new Thread(
    new ThreadStart(sz.TMetodus));
  threads.Add(t);
  t.Start();
}

class Szalhuzo {
  public int PeldanyNo;
  public void TMetodus() {
    for(int i=0; i<5; i++) {
      Console.WriteLine("Szál {0}, iteráció: {1}",
                        PeldanyNo, i);
      Thread.Sleep(5);
    }
  }
}
```

A munkametódusba 5 ms késleltetést tettem, így szépen elversenyezgetnek a szálak egymás között:




A Sleep statikus metódus a megadott ideig *(milliszekundum)* blokkolja annak a szálnak a futását, amiben meghívják. Hosszabb idők esetén kényelmesebb a TimeSpan paraméterű metódus használata.

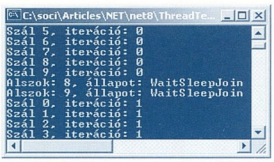
```
Thread.Sleep(TimeSpan.FromMinutes(5));
```

Amikor a szál éppen alszik, az állapota ThreadState.WaitSleepJoin. Ezt persze nem tudjuk megfigyelni magából a szálmetódusából, mivel ha alszik, nem tud végrehajtani még egy Console.WriteLine-t sem. A megfigyelést nyilván egy másik szálból kell megtennünk, például a kiinduló főszálunkból:

```
while(true) {
  for(int i=0; i<10; i++) {
    Thread t = (Thread) threads[i];
    if (t.ThreadState ==
        ThreadState.WaitSleepJoin) {
      Console.WriteLine("Alszok: {0}, állapot:
```

```
            {1}", i, t.ThreadState);
        }
    }
}
```

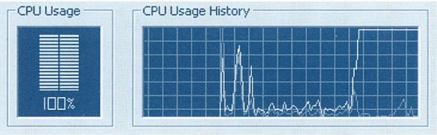
A kimenetben szépen látszik, hogy időnként elkapjuk, amint valamelyik szál alszik:




A példánkban elég után egy végtelen ciklusban figyeltük a szálak működését. Valamilyen módon azért jó lenne kijönni a külső while ciklusból akkor, amikor minden szál befejezte a működését. Ennek egyik módja lenne, hogy figyeljük, valamennyi szál Stopped állapotba került-e?

```
int stoppedNum = 0;
while(stoppedNum < 9) {
  stoppedNum = 0;
  for(int i=0; i<10; i++) {
    Thread t = (Thread) threads[i];
    //...
    if (t.ThreadState == ThreadState.Stopped)
      stoppedNum++;
  }
}
```

A módszer hátránya, hogy a várakozó ciklusunk feleslegesen emészti a processzoridőt, hisz végül is pollozzuk mikor van készen minden szál. A Task Manager gyönyörű zöld oszloppal díjazza kontárságunkat:

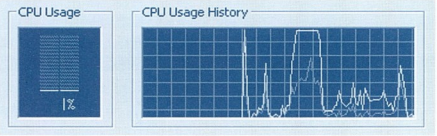



*A buta pollozás hatása*

Némi javulást jelent, ha minden vizsgálat után várunk egy csöppet *(a while ciklusban)*:

```
Thread.Sleep(1);
```

Ez már processzorbarátabb megoldás:




*Barátságban a processzorral*

Ha tényleg nagyon rövid időt akarunk csak várni, nehogy egy milliszekundummal később lépjünk ki, mint ahogy a szálak véget érnek, akkor a Sleep() metódust 0 értékkel kell felparaméterezni. Ezt azt jelenti az OS ütemezőjének, hogy a szálunk *(amiben kiadtuk a Sleep()-et)* elvégezte a feladatát, az ütemező mehet tovább a következő szál futtatására. Ha bármilyen eseményt pollozással kell figyelnünk, feltétlenül emlékezzünk erre! A Task Manager szeme most is kiugrik száz százalékra, de amint bármely más szál kér processzoridőt azonnal megkapja, nem veszünk el senkitől időt.

Talán egyesek emlékeznek a különböző elosztott, internetes RSA kódtörő és egyéb programokra. Azok megpróbálták a létező összes szabad processzoridőt anélkül felhasználni, hogy más folyamatokat lelassítanának. Nos, azok pont ezt a módszert használták.

Ha tényleg nem akarunk tolakodóak lenni, a szálak prioritását is alacsonyabbra vehetjük. Erre a szál Priority jellemzője való:

```
Thread.CurrentThread.Priority = ThreadPriority.Lowest;
```

Ebben a példában még a várakozó ciklusunk előtt kiadjuk a fenti sort, így a Maint elindító szálunk Idle prioritással fog futni. A Thread statikus CurrentThread jellemzője kényelmes hozzáférési lehetőséget biztosít a kódot futtató szálhoz.

Ha az általunk indított szálak prioritását akarjuk módosítani, akkor a szálreferencia ismeretében közvetlenül megtehetjük:

```
Thread t = new Thread(...);
t.Priority = ThreadPriority.Lowest;
```

A szálak jellemzői közül lássunk még néhány fontosat.

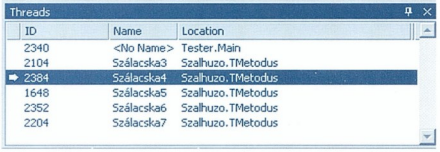
Egy szál lehet foreground vagy backgound jellemzőjű. A foreground szálak életben tartják az elindított programot, azaz amíg az összes foreground szál el nem éri a Stopped állapotot, a processz futni fog. Természetesen a Main()-t elindító szál foreground állapotú, azaz normál esetben a Main() lefutása után a programfutás megszakad.

A háttérben folyamatosan sürgölődő szálak esetén célszerűbb a background beállítás, így az általában egyetlen foreground szál, a Main() szálának lefutása után a CLR automatikusan leállítja a még futó backgound szálakat, majd terminálja az alkalmazást. Gondoljunk például a Word laptördelőjére vagy helyesírás-ellenőrzőjére. Ezek suttyomban futnak a háttérben, és ha backgound szálként futnának *(az Office még nem menedzselt alkalmazás)*, nem kellene velük törődni a program kilépésekor.

A jellemző állítása rendkívül egyszerű, például háttérszálakhoz:

```
t.IsBackground = true;
```

A szálaknak nevet is adhatunk a Name jellemzőn keresztül, diagnosztikai célokból jól jöhet, különösen, hogy a Visual Studio.NET Threads ablaka kiírja a szálak nevét is, így debuggoláskor nagy segítség lehet az eligazodásban:




*Szálak a VS.NET Threads ablakában*

Különösen nekünk magyaroknak *(a nem angol anyanyelvűségünkre, és nem a Szíriuszi származásunkra gondolok)* fontos a CurrentCulture jellemző. A CurrentCulture egy a szálhoz tartozó System.Globalization.CultureInfo osztálypéldányt tárol. Ez meghatározza, hogy a „kulturált", azaz a szál

CurrentCulture beállítására érzékeny típusok hogyan alakítják át magukat szöveggé, illetve hogyan tudnak szövegeket alaptípusokká átalakítani.
Például egy szám pénzként történő kiírásakor („C" formátumsztring) a kiírás a megfelelő kultúrához tartozó pénznemben fog megjelenni:


*A Thread CurrentCulture jellemző hatása az alaptípusok kiíratására*

A fenti ablakot megjelenítő kód:

```
Thread mThread = Thread.CurrentThread;
CultureInfo magyar = new CultureInfo("HU-hu");
CultureInfo nemet = new CultureInfo("DE-de");
double d = 12.4; string s;
s = string.Format("Alap: {0:C}\n", d);
mThread.CurrentCulture = magyar;
s += string.Format("Magyar: {0:C}\n", d);
mThread.CurrentCulture = nemet;
s += string.Format("Német: {0:C}", d);
MessageBox.Show(s);
```

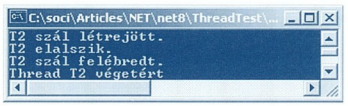
Egy másik jellemző, a CurrentUICulture hasonló célokra felépített jellemző, csak ez a ResourceManager osztályon keresztül felolvasott assembly erőforrások nyelvére van befolyással. Erről bővebben majd egy későbbi Windows Forms cikkben fogok értekezni.

### Thread metódusok

Korábbi példánkban a szálak állapotát vizsgálva vártuk meg, hogy minden háttérszál végezzen a munkájával. Mivel a más szálakra várakozás nagyon gyakori feladat, erre a célra egy külön metódust, a Join()-t találtak ki. A Join()-t az a szál hívja meg, amelyik be akarja várni egy másik befejezését:

```
Thread th = new Thread(
              new ThreadStart(T2Metodus));
th.Start();
th.Join();
Console.WriteLine("Thread T2 végetért");

static void T2Metodus() {
  Console.WriteLine("T2 szál létrejött.");
  Console.WriteLine("T2 elalszik.");
  Thread.Sleep(1000);
  Console.WriteLine("T2 szál felébredt.");
}
```

Azaz esetünkben a főszálunk bevárja az újonnan létrehozott szál befejeződését:

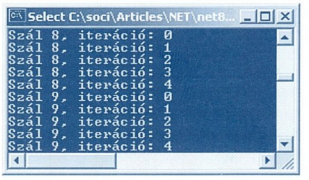



Amikor egy szál a Sleep() meghívásától alszik, az állapota WaitSleepJoin. Ha a Join() hatására blokkolódik, szintén ebbe az állapotba kerül, innen az állapot neve. A harmadik tagról, a Waitről hamarosan szó lesz.
Hogyan írhatjuk át a korábbi példánkat, hogy mind a tíz segédszál futását bevárjuk? Első körben hajlamosak lennénk a szálak elindítása után azonnal Join()-olni az új szálhoz, ám ezzel a következő szál csak az előző vége után indulna el:

```
for(int i=0; i<10; i++) {
  Szalhuzo sz = new Szalhuzo();
  sz.PeldanyNo = i;
  Thread t = new Thread(
               new ThreadStart(sz.TMetodus));
  threads.Add(t);
  t.Start();
  t.Join();  //ajaj, nem lesz ez így jó!
}
```

A kimenetből látható, hogy a szálak csak egymás után tudnak lefutni. Ezért külön kell választanunk a létrehozás-indítás és a bevárás fázisokat:

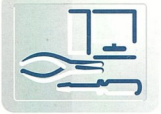
```
threads.Clear();
for(int i=0; i<10; i++) {
  Szalhuzo sz = new Szalhuzo();
  sz.PeldanyNo = i;
  Thread t = new Thread(
               new ThreadStart(sz.TMetodus));
  threads.Add(t);
  t.Start();
}
for(int i=0; i<10; i++) {
  Thread t = (Thread) threads[i];
  t.Join();   //ez már igen
}
```




*A rosszul irányzott Join() sorbaállította a szálakat*

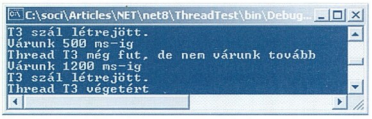
Itt nem baj, ha felakad a várakozó ciklus valamelyik Join()-on, hisz már minden szál dolgozik, az első részben már elindítottuk. A Join()-nak több előnye van a korábbi pollozással szemben. Mivel a szálak állapotát alapvetően az operációs rendszer ismeri legközelebbről, ha a szálak várakoznak egymásra, az operációs rendszer ütemezője tudja, hogy mikor lehet elindítani a várakozó szálat, hisz tudja, mikor ért véget a blokkoló szál. Emiatt a várakozás nem visz el szinte semmilyen plusz processzoridőt, ellentétben a ciklusos pollozással. A Join() mögött valójában a WaitForSingleObject Win32 API függvény hívása zajlik le.
A Join() másik előnye, hogy lehetséges korlátozott ideig is várni. Vagy véget ér a blokkoló szál futása, vagy letelik a megadott időintervallum:

```
th = new Thread(new ThreadStart(T3Metodus));
th.Start();
Console.WriteLine("Várunk 500 ms-ig");
bool b = th.Join(500);
if (b)
  Console.WriteLine("Thread T3 véget ért");
else
  Console.WriteLine("Thread T3 még fut,
↳ de nem várunk tovább");

th = new Thread(new ThreadStart(T3Metodus));
th.Start();
Console.WriteLine("Várunk 1200 ms-ig");
b = th.Join(1200);
if (b)
  Console.WriteLine("Thread T3 végetért");
else
  Console.WriteLine("Thread T3 mégfut,
```



```
de nem várunk tovább");

static void T3Metodus() {
 Console.WriteLine("T3 szál létrejött.");
 Thread.Sleep(1000);
}
```

Az első esetben még nem ér véget a létrehozott szál futása, de a Join() visszatér, mert letelt a megadott fél másodperc. A második esetben türelmesebbre vettük a Join()-t, akkor már kap elég időt a háttérszál:




## WaitHandle

Több szál munkáját bevárni mindenképpen ciklusszervezést igényelt, habár a Join()-os megoldás jelentősen egyszerűbb lett.

Több folyamat közötti szinkronizációra sokkal kényelmesebb lehetőséget biztosítanak a WaitHandle osztály és leszármazottai. A Join() alapvető hiányossága, hogy csak a szálak vége esetén jelez. Pedig sokszor cél, hogy egyes szálak a munkájuk bizonyos szintjén bevárják a többieket, majd folytassák munkájukat. Ekkor vethetjük be a WaitHandle családot.

A WaitHandle osztálynak három leszármazottja van: az AutoResetEvent, a ManualResetEvent és a Mutex. A család tagjai mögött Win32 OS handle-ök állnak, így programunk más rendszerre portoláskor lehet, hogy gondban leszünk. Ha kifejezetten fontos a többplatformos futtathatóság, használjuk a később tárgyalandó Monitor osztályt.

Az AutoResetEvent és a ManualResetEvent lehet Signaled és Nonsignaled állapotban. Az egyszerűség kedvéért nevezzük bekapcsolt és kikapcsolt állapotnak.

A bekapcsolás a Set() metódus hívásával, a kikapcsolás a Reset() hívásával érhető el. A WaitHandle osztály WaitAll() metódusa WaitHandle objektumok tömbjét vár paraméterül, és addig blokkolja az aktuális szál futását, míg a paraméterként megadott WaitHandle objektumok bekapcsolt *(Signaled)* állapotba nem kerülnek.

Nézzük meg ezt egy példán keresztül. Létrehozunk 10 szálat, amelyek végrehajtásuk egy bizonyos pontján visszajeleznek a főszálnak, hogy eddig készen vannak. A szálat futtató metódust tartalmazó osztály definíciója:

```
class Szalhuzo {
  public int PeldanyNo;
  public ManualResetEvent eddigOk;
  public ManualResetEvent vege;

  public void T4Metodus() {
    Console.WriteLine("T4({0}) szál létrejött.",
                      PeldanyNo);
    Thread.Sleep(200);
    //Visszajelzünk, eddig kész vagyunk.
    eddigOk.Set();
    Thread.Sleep(2000); //Dolgozunk tovább...
    vege.Set();
  }
}
```

Minden egyes szál mögött ebből az osztályból áll egy-egy példány, így mindegyik tartalmaz két ManualResetEvent referenciát *(eddigOk és vege)*, melyet majd a szálakat létrehozó kód hoz létre:

```
// ManualResetEvent használata
ManualResetEvent [] eddigOk = new ManualResetEvent [10];
ManualResetEvent [] vege = new ManualResetEvent [10];
for(int i=0; i<10; i++) {
  Szalhuzo sz = new Szalhuzo();
  sz.PeldanyNo = i;
  Thread t = new Thread(new
             ThreadStart(sz.T4Metodus));
  threads.Add(t);

  //Induláskor kikapcsolt
  eddigOk[i] = new ManualResetEvent (false);
  sz.eddigOk = eddigOk[i];
  vege[i] = new ManualResetEvent (false);
  sz.vege = vege[i];
  t.Start();
}
//Várunk...
WaitHandle.WaitAll(eddigOk);
Console.WriteLine("Mindenki végzett a részfeladatával.");
//Várunk tovább...
WaitHandle.WaitAll(vege);
Console.WriteLine("Minden szál terminált.");
```

Minden szálhoz létrehozunk két ManualResetEvent példányt. Mindkettő kikapcsolt *(Nonsignaled)* állapotban lesz, ezért a false a konstruktorban. A példányokat átadjuk a szálak osztályainak az eddigOk és vege mezőkön keresztül.

A szálak elindulnak, dolgoznak egy darabig, majd visszajeleznek, hogy az adott szintig készen vannak a feladatukkal. Ezt egyszerűen az eddigOk.Set() hívásával teszik meg. Ettől a példányhoz tartozó ManualResetEvent bekapcsolt állapotba kerül.

A főszál *(karmester) a WaitHandle.WaitAll(eddigOk)* segítségével bevárja, míg az összes szál végez a feladatával. A WaitAll addig vár, míg az összes WaitHandle leszármazott *(ManualResetEvent)* példány a megadott tömbben Signaled *(bekapcsolt)* állapotba nem kerül. Hasonlóan várunk arra az eseményre is, amikor minden szál kilép.

A várakozás ismét operációs rendszer szinten van megvalósítva, azért nem igényel jelentős erőforrásokat.

Abban az esetben, ha már a leggyorsabban visszajelző szál után tovább akarunk lépni, használjuk a WaitHandle.WaitAny() metódust. Például több szálon kiküldünk kéréseket valamilyen tükrözött *(mirrored)* internetes erőforrás elérésére, majd a leggyorsabban válaszolót futni hagyjuk, a többi szálat pedig megállítjuk. Erre a célra kiválóan használható a WaitAny() metódus.

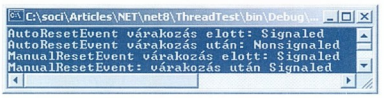
Ha egyetlen szál bizonyos pontjáig kell csak várnunk, használhatjuk a példányszintű WaitOne() metódust. Ugye emlékszünk, ennek előnye a Thread.Join()-nal szemben, hogy a segédszál végrehajtásának bármely pontján visszajelezhetünk, míg a Join() csak a szál terminálásakor jelez vissza.

A Wait() metódusok is képesek az időtúllépésre figyelni, így biztosak lehetünk abban, hogy a várakozó szál nem blokkolódik meghatározatlan ideig.

Mi a különbség a ManualResetEvent és az AutoResetEvent között? A bekapcsolt *(Signaled)* állapotban lévő AutoResetEvent automatikusan Nonsignaled *(kikapcsolt)* állapotba kerül, ha az őrá várakozó szál várakozása véget ért. Azaz, ha a főszál várakozik egy háttérszálra, ami bekapcsol egy AutoResetEventöt, a várakozás után az AutoResetEvent automatikusan Resetelődik, azaz Nonsignaled állapotba kerül. Az alábbi példa szemlélteti a két WaitHandle közötti különbséget. Némi trükközésre azért van szükség, mert közvetlenül nem lehet lekérni a belső kikapcsolt/bekapcsolt állapotot, így csak ismételt - időtúllépéssel megvalósított - várakozással tudjuk kitalálni melyik milyen állapotban van a várakozás után.

```
bool bAuto = test.Auto.WaitOne (1000, false);
string state = bAuto ? "Signaled": "Nonsignaled";
Console.WriteLine("AutoResetEvent várakozás
↳ előtt: {0}", state);
bAuto = test.Auto.WaitOne(1000, false);
state = bAuto ? "Signaled" : "Nonsignaled";
Console.WriteLine("AutoResetEvent várakozás
↳ után: {0}", state);

bool bMan = test.Man.WaitOne(1000, false);
state = bMan ? "Signaled" : "Nonsignaled";
Console.WriteLine("ManualResetEvent várakozás
↳ előtt: {0}", state);
bMan = test.Man.WaitOne(1000, false);
state = bMan ? "Signaled" : "Nonsignaled";
Console.WriteLine("ManualResetEvent: várakozás
↳ után {0}", state);
```

Látható, hogy az AutoResetEvent méltó a nevére, és automatikusan kikapcsolja magát, ha beteljesítette feladatát:




## Mutex

A Mutex is WaitHandle utód, ám különlegessége, hogy processzeken átívelő szinkronizációra is alkalmas. Ehhez a Mutex példányunkat névvel kell ellátni, amely névnek a számítógépen egyedinek kell lenni.

Egy mutexobjektumot egyszerre mindig csak egy szál birtokolhat. Amíg a szálnál a Mutex, a többi szál kénytelen rá várakozni. Amikor a szál végzett a munkájával, meghívja a Mutex ReleaseMutex metódusát, így a következő várakozó szál kapja meg a vezérlést.

Az alábbi példa egy olyan WinForms program Main() metódusát mutatja, amelyből csak egy példány futhat:

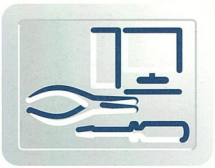
```
static void Main() {
  bool first;
  Mutex m = new Mutex(true, "OnlyOne", out first);
  if (first) {
    try {
      Application.Run(new Form1());
    }
    finally {
      m.ReleaseMutex();
    }
  }
  else {
    MessageBox.Show("Már fut egy példány",
 "Mutex", MessageBoxButtons.OK);
  }
}
```

Folytatjuk...

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

**A cikkben szereplő URL-ek:**

[1]: Letölthető példakódok
http://technet.netacademia.net/download/dotnet



# Az UML

## 1. rész



A modellezés témát folytatva, pontosabban részletezve UML sorozatom első részében a módszer kialakulásának körülményeit szeretném bemutatni, illetve néhány olyan alapfogalmat, amelyek megértése elengedhetetlen a későbbiek megértéséhez.

### Az igények

Az előző számban már írtam arról, hogy csekélyértelmű medvebocs korom óta mennyit fejlődtem, legalábbis a programozási szokások terén. Eleinte minden előzetes tervezés nélkül vágtam bele a dolgokba, majd az első hibák, memória-túlcsordulások és reménytelennek tűnő debuggolások után gondolkodni is elkezdtem *(mekkora előrelépés volt ez!)*. Egy idő után megtanultam, hogy ha előre gondolkodom, sokkal messzebb jutok.

Valahogy így történt ez a szoftverfejlesztéssel, mint globális fogalommal is. A megvalósítható dolgokkal együtt az igények is egyre nőttek, nőnek ma is, a növekvő igényeknek meg kell felelni, egyre több, addig kivitelezhetetlennek tűnő dolgot meg kell valósítani, és így tovább. Úgy tűnik, ebből az ördögi körből nincs kiszállás. A folyamatosan növekvő igények kielégítéséhez pedig hatékony technikák és módszerek szükségesek.

Ezeket a mindig újabb és újabb fogásokat persze folyamatosan tanulni kell, ha lépést akarunk tartani rohanó világunk kihívásaival. Sok informatikus szakember azonban még mindig a több évtizedes, „jól bevált" módszereket használja, az idő múlásával pedig egyre nehezebben és nehézkesebben változtatja meg ezeket a szokásokat, beidegződéseket.
Sajnos ezen egyik pillanatról a másikra nem tudunk változtatni. Nem is létezik módszer, amellyel mindenki meggyőzhető lenne, talán nem is cél ez. Én egyetlen dolgot szeretnék most megpróbálni: bemutatni az UML-t, mint módszertant, és érveket felsorakoztatni mellette.
Először is gondoljuk végig, hogy egy átlagos, hétköznapi fejlesztőnek *(persze ha létezik ilyen…)* milyen igényei lehetnek egy tervezési metodikával kapcsolatban:

- jól elkülöníthető, személyekre és csoportokra egyértelműen felbontható, pontosan definiált feladatokat határozzon meg, elősegítve a hatékony csapatmunkát;
- pontosan definiálja a fejlesztendő terméket;
- a termék minőségének méréséhez *(és eléréséhez)* egyértelmű mérési szempontokat állapítson meg;
- a felhasználói igényeket teljes mértékben vegye figyelembe, a fejlesztők és a megrendelők között magasfokú együttműködést tegyen lehetővé;
- az egyéni és csoportos feladatok egységes kezelését is tegye lehetővé, ugyanakkor ne jelentsen túl szigorú megkötéseket sem, ezáltal biztosítson teret az egyéni fejlesztői kreativitásnak.

Természetes igény, hogy mindezen követelményeknek egy szemléletes, sokatmondó ábrázolásmódot feleltessünk meg, amely érthető mindenki számára. Ezáltal a fejlesztők közötti belső, és a kereskedők, megrendelők, felhasználók felé irányuló külső kommunikáció is egyszerűbbé, egyértelműbbé válhat.

### A „sikersztori"

Az egységesítés első gondolata a '70-es évek közepén merült fel, és persze eleinte korántsem egységes javaslatok születtek. *(Sajnos sok helyen ma is jellemző, hogy egy fejlesztőnek annyi jelölésmódot, ábrázolási technikát kell ismernie, ahány különböző projektben dolgozik.)*
A szoftverek komplexitásának növekedésével és az objektumorientált nyelvek megjelenésével, terjedésével egyre sürgetőbb lett új, mindenki által használható tervezési, elemzési módszerek és egységes nyelvek, jelölésrendszerek kidolgozása. A '90-es évek elejéig több javaslat is született, ezek többsége azonban nem felelt meg az előbb felsorolt igényeknek *(legalábbis nem mindegyiknek)*, illetve a fejlesztendő rendszer bonyolultságának, az egyre növekvő problématér nagyságának kezelése is kívánalmakat hagyott maga után.
Ekkor azonban jött a Nagy Áttörés: olyan új módszertanok jelentek meg, amelyek sikeresen megfeleltek a kihívásnak, és kiállták a fejlesztők próbáját. Ilyen eszközök voltak például: Coad-Yourdon, Fusion, Martin-Odell, OMT, OOSE, Shlaer-Mellor, stb. [1]

### Az egységesítés

Három élvonalbeli fejlesztő *(Grady Booch, Ivar Jacobson és James Rumabugh)* felismerte az igényt egy egységes módszertan kidolgozására, így összefogtak, hogy az addig megszületett metodikák előnyös tulajdonságait kiemeljék, és azok alapján egy közös koncepciót dolgozzanak ki. Az ő munkásságuknak köszönhetően, több más szerző módszertanának és tapasztalatainak felhasználásával született meg 1997-ben az UML *(Unified Modeling Language)* modellezőnyelv, illetve 1998-ban a RUP *(Rational Unified Process)* módszertan *(később több, UML-re építő módszertant is kidolgoztak a szoftvercégek)*.

Mindeközben alakult ki az a fejlesztési metodika is, amelyet inkrementális fejlesztésnek nevezünk. Ezt az a felismerés hívta életre, hogy a gyakorlatban a legtöbb szoftver életének nagyobbik részét teszi ki *(és a költségek nagyobb részét is igényli)* az újabb és újabb verziók előállítása, a különböző módosítások végrehajtása, korábbi hibák kijavítása, hiányosságok pótlása. Ebben a folyamatban az új szoftver mindössze az első változat előállítását képviseli. A „semmiből" létrehozott verzió után az újabbakat mindig a meglévő változatok módosításával, kiegészítésével hozzuk létre.

Egy-egy verzió megvalósítása azonban hosszú ideig *(akár évekig is)* elhúzódhat, közben a fejlesztők egyre több dolgok megtanulnak, tapasztalnak, így az új ismeretek fényében szükség lehet a követelmények módosítására *(ugye ismerős a helyzet?)*. Ezért célszerű lehet egy verzió készítését több kisebb lépésben, alváltozatok egymásutánjaként előállítani. Ezt a folyamatot nevezzük a fejlesztés inkrementális modelljének. Ennek során lehetőség van arra, hogy a rendszer problematikus részeit fejlesszük ki először, majd ehhez hozzáfejlesztjük az újabb és újabb részleteket. *(Először többnyire a rendszer viselkedését és kezelési stílusát érzékeltetjük, vagyis a felhasználói felületeket készítjük el.)*

A rendszer tehát nem egyenletesen fejlődik, hanem bizonyos részeivel egészen a megvalósításig előreszaladunk, tapasztalatokat gyűjtünk, és az elkészült részekhez rakjuk hozzá a még hiányzókat.

A másik, meghatározó paradigma az architektúraszemléletű fejlesztés. Ez azt jelenti, hogy a rendszer architektúráját különböző nézetekben képzeljük el, így a modellt sokkal jobban le lehet írni, mintha egyetlen diagramba szeretnénk mindent besűríteni.
Ezzel el is érkeztünk jelen cikksorozatom témájához, az UML-hez.

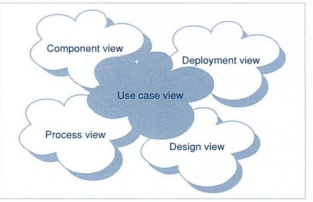
### Az UML

Az UML egy szabványos, egységesített modellezőnyelv, amelynek segítségével a fenti leírások, fejlesztési modellek rendkívül jól szemléltethetőek; a tervezés, a specifikáció, a dokumentálás mind grafikus formában, beszédes ábrák, diagramok, táblázatok segítségével végezhető. A legtöbb vezető szoftvervállalat felismerte már az UML-ben rejtőző lehetőségeket, foglalkozik a szabvány továbbfejlesztésével, így ma már az UML-eszközök kínálata szerfelett széles, mindenki megtalálhatja a számára legmegfelelőbbet.
A szoftverfejlesztők egymás közötti, és a felhasználók felé irányuló kommunikációt csak egy közösen elfogadott, mindenki által ismert modellezőnyelv teszi lehetővé. A legtöbb fejlesztési módszertan ajánl valamiféle jelölésrendszert, viszont a nyelv és a tervezési módszer élesen elkülöníthető *(ha kész a terv, és azt valamilyen módon ismertetni tudjuk a többiekkel, akkor már mindegy, hogy hogyan, milyen módon jutottunk el a kész tervig)*. Ezt ismerték fel az UML készítői, és fejlesztették ki a jelölésrendszert.

A grafikus szemléltetés rendkívül hatékony módszer, azonban szem előtt kell tartanunk, hogy a fejlesztésben részt vevő különböző szakemberek mind különböző szemszögből szemlélik a rendszert, illetve ugyanaz a szakember is láthatja másképp ugyanazt a rendszert a fejlesztés más és más szakaszaiban. Ezért olyan módszert kell alkalmazni, amely képes kezelni ezt a sokrétűséget, viszont kellően egyszerű ahhoz, hogy széles körben elterjedjen.

### Az UML nézetei

Az UML-ben mindez úgy valósul meg, hogy egy rendszerhez több különböző nézetet rendelünk, melyek kiegészítik, és bizonyos értelemben át is fedik egymást *(lásd az ábrát)*. A rendszerről oly módon kapunk teljes képet, ha ezekre a nézetekre együtt, egy egészként tekintünk.




*Az UML nézetei*

1. **A használati eset nézet** ***(use case view)***
   A rendszer viselkedését, funkcionalitását írja le a szereplők és a feladatok megjelölésével, a felhasználó szemszögéből nézve. *(A szereplő (actor) olyan személy vagy elem, amely kapcsolatban áll a rendszerrel, és aktívan kommunikál azzal, funkciókat indít el, vagy hajt végre.)*
   A használati esetek jól meghatározott funkciók, amelyek végrehajtása üzenetváltást kíván. Meghatározó szerepet játszanak a fejlesztési folyamatban, hiszen a működés leírása a többi nézetet is jelentősen befolyásolja.

2. **A komponens/implementációs nézet** ***(component view)***
   A rendszer struktúráját, a programkomponensek, állományok kapcsolatát írja le. Elsősorban a programfejlesztők használják, hiszen az elemek, kódkomponensek egyetlen működőképes rendszerré integrálását valósítja meg.

3. **A folyamatnézet** ***(process view)***
   A rendszert folyamataira, végrehajtható egységeire bontva ábrázolja. Célja a párhuzamosítható műveletek felismerése, az aszinkron események megfelelő kezelése, ezáltal hatékony erőforrás-gazdálkodás elérése.

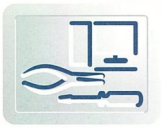
4. **A telepítési/működési nézet** ***(deployment view)***
   A rendszer fizikai felépítését rögzíti, a hardvertopológiát, az adott szoftverkomponensek által igényelt erőforrásokat írja le.

5. **A logikai/tervezési nézet** ***(design view)***
   Azokat az elemeket, feltételeket határozza meg, amelyek a megfelelő működéshez kellenek. Elsősorban a tervezők és fejlesztők számára fontos, hiszen a rendszer statikus struktúráját, az együttműködést, az objektumok közötti kommunikációt írja le. Itt kell pontosan meghatározni a belső struktúrát és interfészeket is.

### Elemek és relációk

A rendszer egyes nézeteinek statikus és dinamikus sajátosságait különböző diagramokkal fejezhetjük ki, amelyek a rendszer elemei közötti relációkat írják le, több különböző szemszögből nézve.
Az UML négy relációtípust különböztet meg:

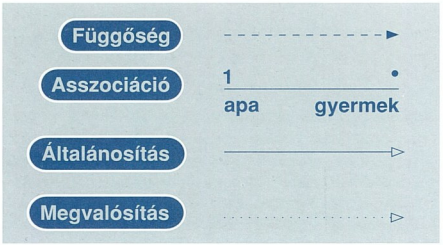
1. **függőség** ***(dependency)***
   Két elem között akkor áll fenn, ha az egyik (a független) elem változása hatással van a másik (a függő) elemre. Kölcsönös a függőség akkor, ha mindegyik elem hatással van a másikra.
   Grafikus ábrázolásában a szaggatott nyíl a független elem felé mutat.



**2. asszociáció** ***(association)***
Az objektumok kapcsolatát, ezek struktúráját határozza meg. Speciális esete a rész-egész viszony, amely kétféle lehet: aggregáció vagy kompozíció. Aggregáció esetén a rész az egészhez tartozik, de önmagában is létező entitás, míg kompozíció esetén a rész önmagában nem létezhet, csak az egész elemeként.
A szemléltető nyílon jelöljük az asszociáció irányát, multiplicitását. Rész-egész viszony esetén az egésznél lévő vonalvég egy csúcsára állított, aggregációnál „lyukas", kompozíciónál tömött rombusz.

**3. általánosítás és specializáció** ***(generalization/ specification)***
Az objektumok speciális viszonya, gyermek-szülő kapcsolat, amelyben a fölérendelt elem az általános, az alárendelt a specializált.
Ábrázolása egy „lyukas" nyíl, amely a szülő felé mutat.

**4. megvalósítás** ***(realization)***
Annak kifejezése, hogy egy osztály biztosít egy másikat arról, hogy elvégez számára egy bizonyos feladatot.
Grafikus szimbóluma egy szaggatott, „lyukas" fejű nyíl.

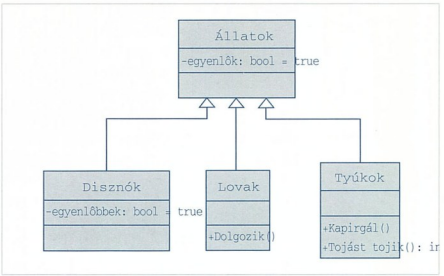



***A relációk UML szimbólumai***

## Diagramok

A diagramok olyan gráfok, amelyek csomópontjai elemeket, élei az elemek közötti kapcsolatokat képviselik. A különböző diagramok közös elemeket is tartalmazhatnak, hiszen ugyanazt a rendszert ábrázoljuk többféle megközelítésben. Az UML-ben két nagy csoportról, statikus és dinamikus diagramokról beszélünk.
A statikus diagramoknak öt fajtáját különböztetjük meg:

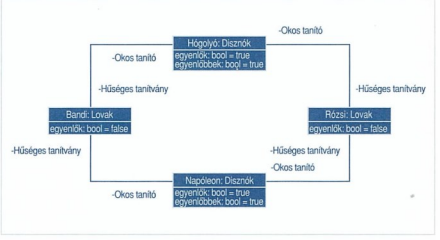
**1. Osztály-diagram** ***(class diagram)***
Az osztályok, interfészek, ezek együttműködésének és kapcsolataiknak ábrázolására szolgál.
Egy tipikus osztálydiagramot mutat a következő ábra, melyet Orwell: Állatfarmja alapján rajzolhatunk fel. (Bizonyára mindenki ismeri a nagysikerű regényt.)
Az alábbi ábráról jól látható, hogy az osztályokat olyan téglalapokkal jelöljük, amelyek három részből állnak: felső harmadában az osztály neve, középen az osztály attribútumai, alul pedig az osztályhoz tartozó metódusok szerepelnek.




***Osztálydiagram („Minden állat egyenlő, de egyes állatok egyenlőbbek a többinél")***

A nyilak mentén érvényesek az öröklési szabályok, azaz minden gyermek-osztály rendelkezik egy bool típusú egyenlők attribútummal, míg a Disznók osztálynak még egy egyenlőbbek nevű attribútuma is van. A Lovak dolgoznak, a Tyúkok kapirgálnak és bizonyos számú tojást tojnak (például éves szinten).
Az attribútumok előtt szereplő apró jelek a következőket jelenthetik:
+ : public
- : private
# : protected
attribútumról van szó.

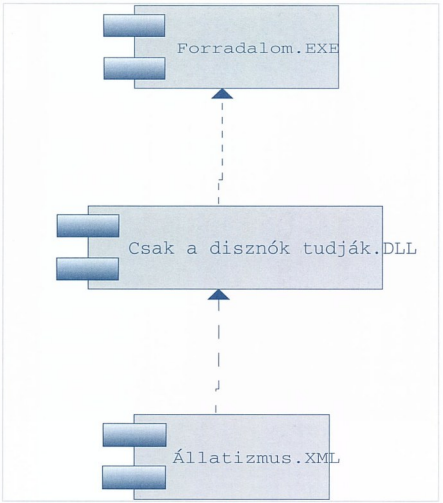
**2. Objektum-diagram** ***(object diagram)***
Az osztály-diagram elemeinek pillanatnyilag létező példányait, azok kapcsolatait szemlélteti.




***Objektumdiagram („A tanítás és a szervezés munkája természetszerűen a disznókra hárult, mert általánosan elismerték, hogy ők a legokosabbak az állatok között... Leghűségesebb tanítványuk a két igásló lett, Bandi és Rózsi...")***

Az objektumokat két részből álló téglalapok reprezentálják, amelyek felső felében az objektum nevét és osztályát, alsó felében attribútumait tároljuk (aktuális értéküket is feltüntetve).

**3. Komponens-diagram** ***(component diagram)***
A komponensek egymáshoz való viszonyát fejezi ki. Ha egy komponens osztályok, interfészek és köztük lévő kapcsolatok együttese, ez az ábrázolásmód szoros kapcsolatban áll az osztály-diagrammal.

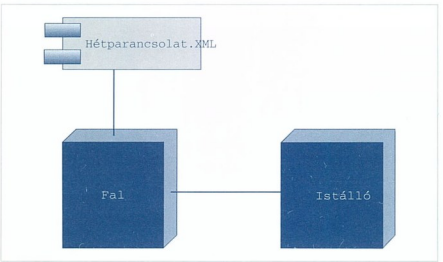



*Komponens-diagram („Nem tudták, hogy az őrnagy megjósolta Forradalomra mikor kerül sor, és jó okkal nem is gondolhattak arra, hogy az ő életükben bekövetkezik, de világosan látták, hogy kötelességük felkészülni rá. ...a rendszernek az Állatizmus nevet adták.")*

A komponenseket a fent látható módon téglalapokkal jelöljük, bal oldalukon két kis „fogacskával". A szaggatott nyilak az mutatják, hogy mely komponens melyiknek szolgáltat valamilyen információt, eljárást, stb.

### 4. Telepítési/működési diagram *(deployment diagram)*

A futás közben igényelt erőforrásigényt, és a csomópontokon működő komponenseket ábrázolja.

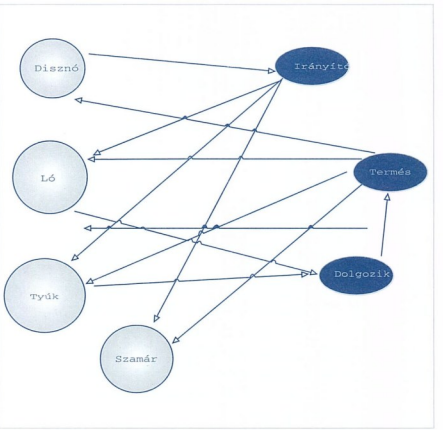



*Telepítési diagram („Ezt a Hétparancsolatot most fel fogják írni a falra, ezek változtathatatlan törvények lesznek, és az Állatfarm valamennyi állatának mindörökké ezek szerint kell élnie.")*

Az erőforrásokat téglatestek ábrázolják, ezek közötti kapcsolatot, illetve a szoftverkomponensek „hovatartozását" szemlélteti a fenti ábra.

### 5. Használati eset *(use case)* diagram

A valós rendszer szereplőit, ezek kapcsolatát és tevékenységeit mutatja be. A rendszer szervezése, viselkedésének leírása és ellenőrzése szempontjából létfontosságú!

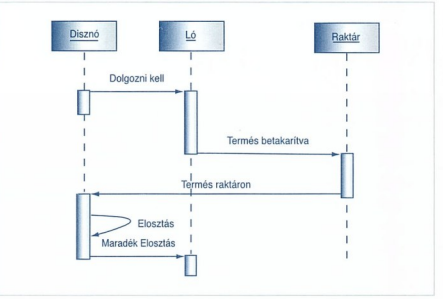



*Használati eset diagram („Mennyit gürcöltek, izzadtak, hogy begyűjtsék a szénát!... De a disznók olyan okosak voltak, hogy minden nehézséget le tudtak győzni.")*

Az UML diagramok másik csoportja, a dinamikus (más néven viselkedés-) diagramok az objektumok egymásra hatását, kommunikációját, üzenetváltásait mutatják be. Négy típus sorolható ide:

1. Szekvenciadiagram *(sequence diagram)*

Az üzenetek küldésének és fogadásának időrendi sorrendjét határozza meg, a használati esetekből kiindulva.



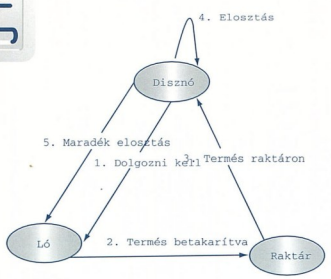

*Szekvenciadiagram („Az állatok ebben az évben úgy dolgoztak, mint a rabszolgák.")*

Itt az üzenetváltás szereplőit nagy téglalapokkal jelöljük az ábra tetején. Az idő múlását a függőleges tengely szemlélteti, a cselekvéseket a szaggatott vonalú tengelyeken lévő hosszú téglalapok (*ezek hossza a cselekvés időtartamával arányos*). Az üzenetek a küldőtől a fogadóig húzott nyíllal szerepelnek az ábrán.

### 2. Együttműködési diagram *(collaboration diagram)*

Az üzeneteket váltó objektumok kapcsolatát, és az üzenetváltás struktúráját ábrázolja. A szekvenciadiagramból

egyszerű algoritmus alapján megkapható.

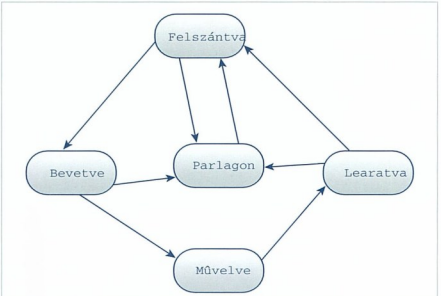



*Együttműködési diagram a szekvenciadiagram alapján*

Az üzenetküldőket és –fogadókat itt egyszerű ellipszisek jelképezik, az üzenetek itt is a fentihez hasonló nyilak, rajtuk az üzenet küldésének relatív időpontja (az üzenet sorszáma) és az üzenet neve.

**3. Állapot- vagy állapotátmeneti diagram** ***(state-chart)***

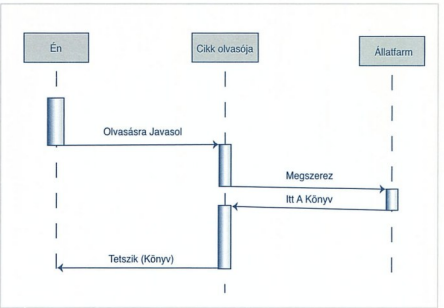
A diagram csomópontjai állapotok, az irányított élek az állapotok közötti átmeneteket reprezentálják. Rendkívül fontos az eseményorientált viselkedés vizsgálatánál.




*Állapotátmenet-diagram („Ez a munka szigorúan önkéntes volt, de aki kihúzta magát belőle, annak felére csökkentették a fejadagját.")*

**4. Aktivitás- vagy tevékenység-diagram** ***(activity diagram)***

Speciális állapotdiagram, amely a végrehajtandó tevékenységek folyamatát mutatja. Jelentősége az objektumok vezérlési folyamatainak tervezésénél a legnagyobb.




*Aktivitás-diagram*

Amint ezekből az egyszerű példákból is látszik, az UML rendkívül sokoldalú modellezőnyelv. Remélem, sikerült mindenkit meggyőznöm arról, hogy a valós *(vagy valósnak vélt)* folyamatok több oldalról, több szempontból történő vizsgálata mennyire fontos.

Sokoldalúsága mellett egyszerűsége az UML-t alkalmassá teszi arra is, hogy mindennapi munkánkban alkalmazzuk, nem kell megijedni az első ránézésre kacifántosnak tűnő ábráktól! Az UML óriási előnye, hogy a jelölésre koncentrál, nem a konkrét tervezési módszerekre vagy megvalósításokra, így bármit le lehet írni vele. Egy UML-lel megtervezett rendszer pedig megvalósítható C++-ban, Javaban, C#-ban, VB.NET-ben...

A következő hónapokban részletesen is bemutatok néhány hasznos „fogást", „trükköt", amelyet a modellalkotás során érdemes alkalmazni, és hamarosan már minden Tech.Net olvasó jártas lesz ebben a „misztikus" világban.

**Molnár Ágnes**
**agnes.molnar@t-systems.com**

**A cikkben szereplő URL-ek:**
[1] http://wwweb.org/smo/bmc/

# Office XP Resource Kit

Akik kicsit jobban belemélyedtek az Office lelkivilágába, azok már jól ismerik az Office Resource Kit eszközök nyújtotta lehetőségeket. Most egy újabb verzióját tölthetjük le ennek a csomagnak.



A megszokott eszközök között újdonságokat is találhatunk, azonban a már ismert varázslók is nagyon jó szolgálatot tehetnek. Nézzünk hát néhányat

### Answer Wizard Builder

Aki használta már az Office súgóját, az biztos használta az Answer Wizardot is. Hányszor volt olyan, hogy hiába kerestünk rá egy-egy kulcsszóra mégsem találtuk meg a megfelelő bejegyzést? Ennek az is oka lehetett, hogy az adott információ nem szerepelt a súgóban. Az Answer Wizard Builderrel ezen segíthetünk. Ha van olyan speciális információ, melyet szeretnénk a súgóban szerepeltetni (legyen az egy nyomtató beállítására vonatkozó adat, vagy speciális, az adott környezetben használható sablon vagy makró használata), akkor azt az Answer Wizard Builder segítségével beépíthetjük súgónkba.

### *System Management Server (SMS) package definiton files (PDFs)*

A javított PDF fájlok lehetővé teszik, hogy SMS 2.0 segítségével távolról telepíthessünk Office XP-t. Két PDF fájlt tartalmaz ez az update. Az egyik az OFF2002.SMS, amely a teljes Office XP-t magába foglalja, a másik pedig az LPK2002.SMS, mely az Office XP Multilingual User Interface Packs-t tartalmazza.

### Custom Installation Wizard

Ez az eszköz az egyik leghasznosabb varázsló. A Custom Installation Wizard segítségével testreszabhatjuk az Office XP telepítését. Ehhez csupán a telepítéshez használt msi fájlra van szükségünk *(a Office XP telepítő CD-jén megtaláljuk)*. A varázsló egy transzformációs fájlt (.mst) készít, melyben azok a változtatások, beállítások kerülnek mentésre, amelyek az alapértelmezett telepítéstől különböznek. A telepítéskor pedig a Windows Installer ezeket a beállításokat fogja alakal mazni. Mivel a varázsló az eredeti msi fájlt nem módosítja, ezért ahányféle mst fájlt készítünk, annyiféleképp telepíthetjük majd az Officet. Nézzünk néhány beállítási lehetőséget:

- Kiválaszthatjuk, hogy melyek azok az alkalmazások, amelyeknek előző verzióját el szeretnénk távolítani.
- Megadhatjuk, melyek azok a komponenssek, amelyeket telepíteni szeretnénk, és ez a telepítés azonnal vagy csak első használat esetén történjen meg, illetve az adott alkalmazást az adott gépről vagy hálózatról szeretnénk-e futtatni.
- Létező Office Profilet *(.ops fájl)* adhatunk az mst fájlhoz, így a telepítés után már egy testreszabott profilt használhatunk.
- Beállíthatunk olyan fájlokat, melyeket a telepítés után szeretnénk akár törölni, akár elhelyezni az adott gépen. Például elhelyezhetjük a vállalat fejléces papírját tartalamazó Word dokumentum sablonját minden felhazsnálónál.
- Registry bejegyzéseket adhatunk meg, elindíthatjuk egyéb alkalmazások telepítését és még lehetne folytatni a sort a beállítási lehetőségekről.

Az mst fájl mentése után egy ablakot kapunk, melyben található egy parancssori utasítás. Ezzel az utasítással futtathatjuk a varázslóval beállított egyéni telepítéseket.

### Office Profile Wizard

A Profile Wizard segítségével elmenthetjük egyéni beállításainkat, mint például mely eszköztárakat használjuk, mi az alapértelmezett elérési útja a fájloknak. A varázslóval azt is beállíthatjuk, hogy a felhasználók ne változtathassák meg beállításaikat, de a megosztott sablonokat megnyithassák.
A felhasználói profilokat OPS fájlokba menthetjük, valamint azokat vissza is állíthatjuk a varázsló segítségével.

### Removal Wizard

Ezzel az eszközzel eltávolíthatjuk azokat az állományokat, melyeket szeretnénk, azt is megadhatjuk, melyek maradjanak. Ezt a varázslót kell akkor is használni, ha egy korábbi Office verzióról szeretnénk Office XPre upgrade-elni. A Removal Wizard a következő programokat ismeri fel és távolítja el: Microsoft Office 4.x, Office 95, Office 97 és Office 2000, Multilanguage csomagok és egyedi Office alkalmazások.

### Setup INI Customization Wizard

Ezzel az eszközzel egyéni telepítési beállításokat készíthetünk, illetve módosíthatunk. A varázsló automatikusan az ini fájl megfelelő részébe írja az általunk kért beállításokat. Valamint létrehoz egy parancssort is, mely tartalmazza a /settings kapcsolót és az általunk megadott ini fájlt.

### OPS File Viewer

Az OPS File Viewer segítségével megnézhetjük azokat a módosításokat, melyeket egy Office profil beállításakor *(OPS fájl)* eszközöltünk. Ez a nézőke azoknak a lehetséges változtatásoknak a listáját is megmutatja, amelyeket az Office Profile Wizardban beállíthatunk egy felhasználónak OPS fájl segítségével.
Az OPS File Viewer használatához ismernünk kell egy Profile Wizard által létrehozott ops fájl nevét és helyét. Ez az alkalmazás az megadott ops fájlt olvassa majd készít belőle egy szöveges állományt, melyet akár Notepad segítségével is elolvashatunk.

Borsi Katalin
bobo@netacademia.net

# Orregér



Azok, akik hosszú órákat töltenek a számítógép előtt, tudják mit jelent egy nem éppen kifogástalanul működő egér. A mikrokapcsolók – inkább előbb, mint utóbb - már csak eredménytelenül kattognak, a golyóra minden szösz ráragad, a görgőkre kiválóan tekeredik fel a leghosszabb hajszál, és az egéralátét vagy a kábel éppúgy véget ér, mint a türelmünk.

A műszaki élet persze már rég túllépett ezeknek a problémáknak a többségén: kapható drót nélküli, rádiós egér, golyó nélküli egér, trackball. Ezek sem enyhítik azonban a kar zsibbadását, a féloldalas testtartást és az ülőhely*(zet)* korlátozott megválasztását.

### Jedi trükk

Amikor nemrég a fenti problémákkal küszködtem, arra gondoltam, de jó lenne, ha az egérkurzor oda menne, ahová gondolom, valahogy így: le, elég, kicsit jobbra, kattints, gyorsmenüt, növeld meg a betűméretet stb. Azt hittem szép álom, míg a [1] cikk fel nem világosított az ellenkezőjéről. Röviden összefoglalva az történt, hogy egy majom agyának több, motoros funkciót végző idegsejtjéhez rendkívül vékony elektródákat vezettek, és megfigyelték, mi történik, ha a majom egy bizonyos betanult cselekvéssorozatot hajt végre. A szemfüles kutatóknak sikerült a neuronok akciópotenciáljának függvényében egy robotkart a majom karjával szinkronban mozgatni. Pardon, a majom tudja azt a bizonyos kart mozgatni a kutatók segítségével. Sajnos ez a megoldás sem nélkülözi a vezetéket, de a kutatók szerint lehetséges olyan agyi implantátum készítése, amely rádiójelekkel juttatja el a szükséges információt a célberendezéshez, például egy egérhez. A pillanatnyilag távolinak tűnő jövő világából térjünk most vissza a jelenbe, és nézzük meg, mi történik e téren az egeret felfedező nagy testvér műhelyében.

### Menjünk inkább az orrunk után

Így gondolja Kentaro Toyama, a Microsoft kutatója, aki olyan egyszerű és olcsó rendszer megalkotásán fáradozik, amely az orr irányának megfelelően változtatja a kurzor helyét [2]. A monitorra helyezett webkamera képéből megállapítható az arc helye és síkja, majd az orrból erre állított merőleges és a képernyősík metszéspontjából máris megkapjuk az egér helyét. Ez amilyen egyszerűnek hangzik, pontosan olyan bonyolult, különös tekintettel az arc paramétereinek meghatározására. Ügyelni kell a különféle bőrszínekre, a bajszosokra, a szakállasokra, a fényviszonyok megváltozására, a kamera látóterében lévő más személyekre, és arra is, ha elfordulok néhány másodpercre beszélgetni valakivel. Bármi is történjék, az egérnek ott kell cincogni az orrom előtt, ha visszanézek a képernyőre.

Toyama az IFA *(Incremental Focus of Attention)* architektúra arcközetésre átalakított változatát használja a feladat megvalósítására. Az IFA több követőalgoritmus összegyúrásával az arc robusztus, valós idejű követését teszi lehetővé. Toyama jelenlegi megoldásában egy véges állapotú gép hat réteg működését vezérli. Ahogy az első rétegtől a hatodik felé haladunk, egyre pontosabban tudjuk, hol áll a fejünk. Ekkor keresésből követésbe csapunk át, egészen addig, amíg a zavaró körülmények miatt nem kell a 3. rétegnél lejjebb ereszkedni. Vegyük sorra, mi történik a keresés hat fázisában, így képet kapunk mindegyik réteg funkciójáról:

Az 1. réteg a pixeleket osztályozza a színüket jellemző RGB értékük alapján, pontosabban aszerint, hogy az R/G és az R/B hányadosok egy meghatározott tartományba esnek-e. A 2. réteg az intenzitás változását vizsgálja spirális pályákon, és ennek megfelelően azokra a képterületekre szűkíti a keresést, amelyek megfelelnek a bőrszínnek, és emellett a mozgásuk is jelentős. Ez a két réteg pixelek halmazát állítja elő, és csak a keresésben vesznek részt, a követésben nem.

A 3. réteg addig terjeszkedik a középpontból 16 különböző sugárirányban, amíg bőrszínt talál. Ekkor már hozzávetőleg ismerjük az arc helyét a képernyővel párhuzamos síkban. A 4. réteg ugyanezt teszi, de a helyen túl az irány közelítő számítását is elvégzi. Az 5. réteg Greg Hager munkáját dicséri azzal, hogy előre megadott sablonokat próbál iteratívan a webkamera képeire illeszteni a távolságok négyzetes összegének segítségével. Végül a 6. réteg az arc öt karakteres pontját figyeli: a szemeket és az orrlyukakat az intenzitás kis területre szűkített vizsgálatával és a szájat a kontúr követésével. Az öt pontot a legkisebb négyzetek elvén összevetjük az előre megadott referenciapontokkal, ami néhány egyszerű mátrixműveleten keresztül eljuttat bennünket a kívánt eredményhez.

### Alkalmassági vizsga

Toyamanak egy ma már a legjobb esetben is csak átlagosnak mondható számítógépen legfeljebb egy-két tizedmásodperc alatt sikerül az egeret a helyére raknia a legnehezebb helyzetekben, az átlagos felhasználó pedig képes 1 cm x 1 cm-es ablakba pozicionálni a kurzort. A zajszűrés tovább javítható és így a kurzor helyzetét még pontosabban lehetne beállítani, azonban a válaszidő ekkor már észrevehetően megnőne.

Mindez biztosíték arra, hogy némi finomítás után az orregérnek jó hasznát vehetik a mozgáskorlátozottak, a játékok és egyéb szórakoztató-programok szerelmesei vagy bárki más a mindennapi munkája során, néhány speciális, az egér rendkívül finom mozgását megkövetelő esettől eltekintve.

Kimaradt a kattintás, ez azonban viszonylag könnyen megoldható, például pislantással, szájmozgással vagy hangfelismeréssel. Aggasztóbb az, hogy a szabadság kiterjesztése egyes területeken korlátozásokkal jár másokon: probléma lehet az irányítás átadása a mellettünk ülőnek és a felhasználótól megkívánt fegyelem is *(elsősorban nem a reggeli borotválkozásra, hanem a fej megfelelő szögben tartására gondolok)*.

Az orregérrel mindenesetre már kényelmesen hátradőlhetnénk a fotelben, azonban sokat kell még kattintani, amíg a boltokban is kapható lesz.

Zacco@fw.hu

A cikkben szereplő URL-ek:

[1 http://www.sciam.com/article.cfm?chanID=sa006&articleID=00065FEA DAEA-1D80-90FB809EC5880000

[2] http://research.microsoft.com/toyama/research.htm

A NetACADEMIA és a NETLOCK Az Első Hitelesítés Szolgáltató közös szervezésében

## Szerezzen átfogó tudást az elektronikus aláírás és titkosítás vállalati szintű alkalmazásában

**Technikai blokk:**

- Mi az a kriptográfia?
- Hogyan tudunk nyílt hálózaton kommunikálni úgy, hogy azonosítani tudjuk kommunikációs partnerünket?
- Meg tudunk bizonyosodni üzeneteink **sértetlenségéről?**
- Hogyan működik a **titkosítás**? Mi a **kulcspár**, a **tanúsítvány**, mi a szerepe az intelligens kulcstároló **eszközöknek**?

**További témakörök:**

- szimmetrikus algoritmusok, nyílt kulcsú **RSA**, hash, a **X.509** tanúsítványok szerepe és **használata**,
- hitelesítésszolgáltató, a hierarchia felépítése,
- kulcstároló eszközök: **intelligens kártya** és **USB Token**,
- bejelentkezés tanúsítvánnyal (Single Sign On), SSL, S/MIME, HTTPS,
- **virtuális magánhálózatok**.

**Jogi blokk:**

- az elektronikus aláírási törvény célja, következményei, a végrehajtási rendeletek.
- Jogkövetkezmények, és a szükséges feltételek megléte.

## Ajándék!


Hallgatóink a tanfolyamon használt kriptográfiai eszközöket a tanfolyam után megtarthatják, és - a mellékelt NetLock tanúsítványokkal - hiteles elektronikus aláírást és nyílt kulcsú titkosítást használhatnak!

**Hozza el a főnökét!** **Jelentkezési határidő: 2002. november 29.**

Ha cégüktől **két fő** jelentkezik a PKI-workshopra, mindketten ingyen részt vehetnek Mikulás-konferenciánkon!

(Mikulás-konferenciánkról további információk az első borító hirdetésében vagy a http://www.netacademia.net/mikulas címen!)

| A tanfolyam időpontjai | Hossz [nap] | Bővebb információ és jelentkezés | Ár [Ft] | Jelentkezés |
|---|---|---|---|---|
| 2002. december 10. | 2 | http://www.netacademia.net/workshop/PKI | 140,000 | ... fő |
| **2003. január 14.** | **2** | **http://www.netacademia.net/workshop/PKI** | **140,000** | **... fő** |

Részletekről érdeklődjön a 06/1/472-1214-es telefonszámon vagy az info@netacademia.net e-mail címen!

A tanfolyam helyszíne: Budapest 1062, Andrássy út 62.

# Microsoft Project 2002
## – hogy a tervekből valóság legyen!

A projektirányítás számos szervezet sikerében és bukásában játszik fontos szerepet. Egy hatékony tervezési eszköz birtokában csökkenthetők a költségek és javítható a termelékenység, ami nyereségnövekedést eredményez. A Microsoft **Project 2002 Standard** minden eddiginél egyszerűbbé teszi az ütemezések és erőforrások kezelését, a projekt állapotának közlését és a projekt adatainak kimutatását.

Cégek, nagyvállalatok részére, a csoportmunkát teljes mértékben támogató **Project 2002 Professional** kínálja a Microsoft, mely segítségével és a központi nyilvántartást és kiszolgálást biztosító **Project 2002 Server** termékkel együttműködve teljeskörű nagyvállalati, projektirányítási, tervezési és döntéshozási feladatok végezhetők el, és annak adatai bármikor, bárhonnan hozzáférhetőek, publikálhatóak.

## Ismerje meg közelebbről tanfolyamainkon!

### Microsoft Project 2002 felhasználói kurzusok

Az érdeklődők **kétnapos, intenzív** tanfolyamok keretében ismerkedhetnek meg a Microsoft Project 2002-es verzióinak használatával.

### Rugalmas időbeosztás

Megpróbáltuk figyelembe venni, hogy a cégek dolgozóikat nem tudják nélkülözni hosszabb időre, ezért a tanfolyamokat **intenzív, egész napos formában** hirdettük meg, délelőtt 9.00 és 16.00 óra között. A tanfolyamok díja az oktatás és a tananyagok mellett az étkezést is tartalmazza a kurzus napjaira.

### Project 2002 Professional és Server – testreszabott oktatások

Oktatóközpontunkon kívül, a megrendelő telephelyén, kihelyezett formában (vidéken is) is vállaljuk az adott cégprojektekhez kapcsolódva tanfolyamok vagy konzultációk megtartását és lebonyolítását, különös tekintettel az **összetett, Project 2002 Professional és Project 2002 Server használatával és bevezetésével** kapcsolatos témákban. Ezt követően igény esetén további **utólagos támogatást** és szaktanácsadást is nyújtunk.

## Menjen biztosra!

## Tervezze üzleti folyamatait Microsoft Project 2002-vel!

**Minőség, egyéneknek kedvező konstrukciók, cégeknek partneri kedvezmények!**

A képzésekkel, szakmai kérdésekkel kapcsolatosan kérjük keresse értékesítési vezetőnket, Projekt oktatónkat, Lovász Attilát a 203-0304/3040 mellékű telefonszámon.

Microsoft CERTIFIED Technical Education Center

SZÁMALK TOVÁBBKÉPZÉS

SZÁMALK

✆ 203-0304/3040 **www.szamalk.hu/tisza** 1115 Budapest, Etele út 68.

## Kedves Olvasónk!

A tech.net magazin több mint két éve látja el olvasóit hónapról hónapra információkkal, tanácsokkal. Szerkesztői és készítői most szeretnék megismerni az Ön véleményét is a kiadványról. Ahhoz, hogy januártól még inkább az igényeinek megfelelő lapot tarthasson kezében, kérjük, válaszoljon az alábbi kérdésekre. A kitöltött kérdőívet **2002. december 10-ig** postán vagy faxon visszaküldők között **Microsoft Kormányt és játékszoftvereket** sorsolunk ki.

**Címünk:** Max & Future Kft. 1519. Bp. Pf. 427.
**Fax:** 06 1 372 8466

A kérdőív interneten is kitölthető és visszaküldhető: **http://microsoft.com/hun/technet/Survey.asp**

**1. Mióta olvasója a tech.net magazinnak?**

**a.** 2000-től **b.** 2001-től **c.** 2002-től

**2. Hogyan jut a magazinhoz?**

**a.** előfizetem **b.** rendezvényeken

**c.** tiszteletpéldányt kapok **d.** egyéb: .....

**3. Milyen gyakorisággal olvassa a magazint?**

**a.** minden számot elolvasok **b.** időnként, amikor ráérek

**c.** elteszem, és konkrét problémánál olvasom el az arról szóló cikket **d.** egyéb.....

**4. Mi az, amit kedvel a magazinban?** .....

.....

.....

**5. Mi az, amit hiányol belőle?** .....

.....

.....

**6. Mit változtatna meg benne?** .....

.....

.....

**Folytatás>>**

**7. A magazinnal kapcsolatos egyéb észrevételek:** .....

.....

.....

**8. Ismeri a Microsoft TechNet eseménysorozatát?**

**a.** igen **b.** nem

**9. Ha igen, milyen gyakran látogatja?**

**a.** félévente 1-2 alkalommal **b.** félévente 3-4 alkalommal

**c.** félévente 5-6 alkalommal **d.** még nem vettem részt egyen sem

**10. Olvassa-e a tech.net Klub levelező listáit, ha igen mely listákat?** .....

.....

**11. Az Ön neve:** .....

**Foglalkozása:** .....

**Beosztása:** .....

**Címe:** .....

**E-mail címe:** .....

**Telefonszáma:** .....

**KÖSZÖNJÜK, HOGY VÁLASZOLT KÉRDÉSEINKRE!**