Szegedi Tudományegyetem

Informatikai Tanszékcsoport

DIPLOMADOLGOZAT

Buczkó Tamás László

2008

Szegedi Tudományegyetem

Informatikai Tanszékcsoport

A LanStore elosztott tárolást támogató rendszer továbbfejlesztése

Diplomadolgozat

Készítette:

Buczkó Tamás László

programtervező

matematikus szakos

hallgató

Témavezető:

Bilicki Vilmos

egyetemi tanársegéd

Szeged

2008

Feladatkiírás

Napjainkban a számítógépek fürtbe kötése egy gyakran használt megoldás melynek segítségével komoly számítási kapacitás, vagy tárhely nyerhető.

A LanStore projekt célja egy olyan nagy megbízhatóságú teljesen decentralizált rendszer megvalósítása melyben az építőkövek egyszerű asztali számítógépek. A hibatűrést egy hagyományos hibajavító kódolás, a Reed-Solomon kód segítségével valósítottuk meg. Az algoritmus minden m adategységhez n hibajavító egységet generál. Az elosztott viselkedés egy szavazáson alapuló algoritmus segítségével lett megvalósítva. A szoftver egyaránt támogatja az IPv4 és az IPv6 protokollokat.

Cél platformként a Windows platform lett kiválasztva mely laborokban és irodákban az egyik leggyakrabban használt operációs rendszer. Mivel a .NET keretrendszer kiválóan integrálódik a Windows operációs rendszerbe ezért fejlesztő, futtató környezetként őt választottuk.

A feladat e már működő elosztott tárolást megvalósító rendszer továbbfejlesztése, új funkciókkal történő ellátása.

Tartalmi összefoglaló

- ***A téma megnevezése:***

  Szerver komponens távoli, központosított menedzselése.

- ***A megadott feladat megfogalmazása:***

A cél a fejlesztés és tesztelés során felmerülő gyakori, a LanStore szerver oldali modulját érintő telepítés, újratelepítés, leállítás, elindítás könnyebbé tétele. A feladatot ellátó Windows Form alkalmazás létrehozása, majd webes megvalósítása.

- ***A megoldási mód:***

A telepítést a szerver oldali gépek megosztott könyvtárának hálózati meghajtóként való csatlakoztatásával, majd helyi fájlműveletekkel oldottam meg. A szerver szolgáltatás távoli leállítását és elindítását a .NET keretrendszer System.Management csomagja kezeli. A Windows Form alkalmazás C# nyelven íródott, a webes felületet ASP felhasználásával hoztam létre.

- ***Alkalmazott eszközök, módszerek:***
  - *Microsoft Visual Studio .NET 2003 és 2005*
  - *Microsoft Virtual PC 2007*
  - *Microsoft Windows XP Professional SP2*
  - *Microsoft Windows Server 2003 Enterprise Edition SP1*
  - *Microsoft Internet Information Services 6.0*
  - *SubVersion*

- ***Elért eredmények:***

Egyszerűen kezelhető grafikus felület, mellyel gyorsan és könnyen elvégezhetőek a kitűzött feladatok. Teljes funkcionalitás nemcsak domain tagsággal rendelkező gépek, hanem egy Windows workgroup esetén is.

- ***Kulcsszavak:***

Elosztott tárolás, távoli menedzsment, WMI, web, IIS

Tartalomjegyzék

Bevezetés

Az elosztott rendszerek fejlesztésekor és tesztelésekor felmerülő egyik nehézség, hogy nem, vagy csak korlátozott mértékben támaszkodhatunk a fejlesztőkörnyezet által biztosított hibakereső rendszerre. Ahhoz, hogy egy módosítás hatásáról megbízható visszajelzést kapjunk, a teljes rendszert frissítenünk kell. Nem elegendő, esetleg nem lehetséges csupán a fejlesztéshez használt számítógépen több példányban futtatni az alkalmazást. Lehetőségünk van ugyan virtuális gépek használatára, de ezzel a módszerrel nem modellezhetünk egy kellően heterogén fizikai hálózatot melyben előfordulhatnak hálózati hibák, csomagok veszhetnek el, és egy átlagos mai PC sem képes nagyszámú virtuális gép futtatására.

A LanStore fejlesztése során felmerült az igény, hogy a teljes hálózatban, amely esetleg nem korlátozódik egy helyiségre, épületre gyorsan és egyszerűen terjeszthessük el az új tesztelésre váró kódot. Rendkívül időigényes lenne egyenként felmásolni az összes számítógépre, minden változatás után.

Két megoldás készült a problémára, az egyik egy Windows Form alkalmazás, míg a másik egy böngészővel elérhető web-es felület. Ez utóbbi előnye, hogy bárhonnan elérhető, így akár otthonról is van lehetőség az iskola hálózatán való tesztelésre.

Nem célom a LanStore részletes bemutatása, működéséről átfogó képet kaphatunk az előző évfolyamok diplomadolgozataiból.

1 Az elosztott hálózati fájlrendszerek és a LanStore

A technika, az alkalmazások és a felhasználói igények fejlődésével egyre nagyobb igény mutatkozik mind nagyobb és nagyobb tárkapacitásra. Az egyik általánosan elterjedt megoldást a fájlszerverek jelentik: a kitüntetett számítógép a kliensekhez képest hatalmas tárkapacitással rendelkezik, amelyhez hálózaton keresztül lehet hozzáférni. Ennek a módszernek nagy hátránya, hogy a szerver meghibásodása vagy a hálózati kapcsolat megszakadása esetén a szolgáltatás leáll, és nem férünk hozzá az adatainkhoz. Kiépítésük meglehetősen drága, a megbízhatóságot pedig általában csak a fájlszerverben lévő merevlemezek RAID tömbje biztosítja. Ha a tárhely kicsinek bizonyul, kénytelenek vagyunk újabb merevlemezeket vásárolni.

Költséghatékonyabb és biztonságosabb megoldás lehet egy elosztott hálózati fájlrendszer, melyhez nincs szükség dedikált drága szerverekre, egyszerű asztali számítógépekből is kiépíthető. Ebben az esetben az adatok nem egy, hanem a hálózat több pontján, elosztva tárolódnak. A fájlrendszer feladata, hogy az önálló számítógépek szabad erőforrásait összefogva, egy olyan tárhelyet kapjunk, amely a felhasználó szemszögéből egyetlen jól definiált egésznek tűnik.

Egy elosztott rendszernek két fontos tulajdonsággal, viselkedéssel kell rendelkezni: az átlátszóság és skálázhatóság. Az átlátszóság (transparency) alatt azt értjük, hogy rendszerünk bizonyos belső viselkedése rejtve marad a felhasználói veselkedés elől. Úgy is tekinthetjük, mint a rendszer több tervezési szempontjainak nézőpontjait elrejtjük a felhasználó elől. Ez elsődlegesen az elosztottságra vonatkozik, de ilyen lehet a megbízhatóság, vagy a rövid válaszidő: a felhasználó semmit sem vesz észre abból, ha egy újabb szerverrel bővítettük a rendszert, vagy meghibásodás miatt kicseréltünk egyet. A skálázhatóság (scalability) a rendszer méretezhetőségének, „rugalmas növekedésének” a lehetőségét takarja. Ha megnöveljük az erőforrás kapacitást, akkor azt várjuk, hogy nagyobb mennyiségű felhasználót képes az kiszolgálni vagy teljesítménynövekedést kapunk. [1]

A dolgozat írásakor egy átlagos alsó kategóriás asztali számítógép 80-120 GB-os winchestert tartalmaz, valamint egy operációs rendszer a szükséges összetevőkkel együtt sem igényel 20 GB-nál több tárhelyet. Látszik, hogy a helyi hálózatok többségénél az asztali számítógépek tárkapacitása nincs kihasználva, a merevlemezek kapacitása meghaladhatja egy átlagos munkaállomás tárigényét, így sok szabad hely kihasználatlan marad.

1.1 A LanStore

A LanStore ezeket az egyébként „elvesztegetett” darabokat egyesíti egyetlen nagy háttértárrá, így a hálózaton keresztül egy gyors és biztonságos tárolási rendszer építhető ki segítségével. Ezeknek az „általános” hálózatoknak viszont van egy nagy hátrányuk a fájlszerverekkel szemben: a számítógépek nagy része nem működik a nap 24 órájában, hanem – munkaállomásokról lévén szó – gyakran kikapcsolják őket (főleg éjszakára és hétvégére), valamint az újraindítások is meglehetősen gyakoriak. Ezért ezeket a tényeket, szempontokat a rendszer tervezésénél különösen fontosnak tekintették, tehát a LanStore néhány számítógép kikapcsolása, újraindítása, illetve esetleges meghibásodása esetén is működőképes marad.

1.1.1 Architektúra

A LanStore architektúrája kliens-szerver alapú. A szerverek tárolják az adatokat, a kliensek pedig a fel-és letöltésért felelősek. Azt gondolhatnánk, hogy mivel a szerverek az adattárolók, ezért rengeteg adminisztrációs feladatuk van, és ez a processzoridő nagy hányadát fogja elvonni a számítógépeknek szánt eredeti feladattól. Ez abszolút nem elfogadható dolog, hiszen így azért a tárkapacitásért, amit nyerünk, súlyos teljesítménybeli visszaesésben jelentkező árat kell fizetnünk. Egy ilyen kompromisszumba valószínűleg nem sokan mennének bele. Erre a problémára megoldás a vékony szerver vastag kliens architektúra, amit a LanStore is alkalmaz. Ez a kifejezés azt jelenti, hogy egy file feltöltési vagy letöltési művelet esetén minden adminisztrációs és számítási idővel járó feladatot a kliensnek kell elvégeznie, a szerver oldalra ténylegesen csak az adatok tárolása marad.

A kommunikáció közben több probléma is felléphet. Előfordulhat, hogy ugyanazon kliens több egyforma műveletsort hajt végre párhuzamosan, amelyek több üzenetváltást igényelnek, ekkor annak eldöntésére, hogy egy adott üzenet melyik műveletsorhoz tartozik, tranzakció kezelést használunk. Az is előfordulhat, hogy hálózati üzenetekkel túlterhelődik a szerver. Ennek kivédésére a szerver oldal elárasztás védelmet tartalmaz. [2]

1.1.2 Biztonság

Egy elosztott rendszerben, ahol a kommunikáció publikus csatornán folyik, biztosítani kell az adatok eredetiségét, azok sérthetetlenségét, és esetleg a titkosítását. Az eredetiség esetén biztosnak kell lenni abban, hogy az adat attól a féltől származik, aki küldte. A sérthetetlenség azt jelenti, hogy a küldött üzenet tartalmát útközben nem módosította senki. A digitális aláírás használata mindkét feladatot megoldja. A digitális aláírás során nem magát az üzenetet tikosítjuk, hanem az üzenetet kibővítjük egy tranzakció azonosítóval és egy időbélyeggel, majd az így képzett szöveg kivonatát (*hash*) tikosítjuk. Az időbélyeg, és a tranzakció azonosító az adatforgalom visszajátszása ellen nyújt védelmet. Mivel az egész szöveg helyett csak a kivonat kerül titkosításra, így csökken a szükséges számítási kapacitás. A LanStore esetében a napjainkban a leggyakrabban használt MD5 kivonatoló függvényt alkalmazták. [2]

A LanStore rétegei közötti belső kommunikációt is a biztonsági réteg felügyeli. Azaz bármelyik réteg, amelyik üzenetet akar küldeni a hálózaton, vagy adatot kér az adatbázisból, szükségszerűen a biztonsági rétegen keresztül teheti ezt meg.

A szerver oldalon a biztonsági réteg feladata a rajta átfolyó meta–adatok és adatok szűrése is, ellenőrizve, hogy a kliens oldalon lévő felhasználó valóban jogosult a kért műveletre. Az átküldött adatok bájt folyammá alakítása (szerializálása) is ezen a szinten történik meg

1.1.3 Hibatűrés

Következő ellenérvünk egy ilyen rendszer ellen az, hogy egy valós élethelyzetben rengetegszer előfordul, hogy egy-egy irodában, vagy teremben egy számítógép lefagy, elromlik, vagy csak egyszerűen kikapcsolják. Ekkor a rendszer konzisztenciája súlyosan megsérülhet, mivel bizonyos adatokat nem tudunk visszaállítani. A LanStore ezen problémára megoldást kínál a kódoló rétegének segítségével, amelynek működése a következő módon írható le: Az adatokat redundánsan tároljuk, vagyis egy bizonyos adatot egyszerre több gép is tartalmaz. Ennek hátránya, rögtön látszik: nem tudjuk kihasználni az összes rendelkezésre álló tárolókapacitást, de cserébe a rendszer konzisztenciája megmarad még akár több gép kiesése esetén is. Ez a kompromisszum vállalhatónak tűnik. Mint minden számításigényes feladat ez is a kliens oldalon végeződik el. A hibatűrő képességet a Reed-Solomon kódolás biztosítja. A kódolás működésének lényege a következő: tegyük fel, hogy a szervereink száma n. A csíkozás úgy működik, hogy fogjuk a fájlrendszerbe

feltölteni kívánt állományt és k < n részre bontjuk. A k darab részből valamilyen hibajavító algoritmussal (esetünkben ez a Reed-Solomon eljárás) n darab részt képezünk. Az n darab részből „valamennyit” elhagyva a hibajavító algoritmus képes visszaállítani az eredeti állományt. Ezt a „valamennyit” az egészre nézve nevezzük a hibatűrés mértékének. Az is nagyon fontos, hogy ez a hibatűrés skálázható legyen. Sajnos az egy rendkívül nehéz feladat, hogy a skálázhatóságot is dinamikusan valósítsuk meg. Ebben az esetben a rendszer, működés közben is alkalmazkodna a terheléshez. Ehelyett be kell érnünk a statikus skálázhatóság lehetőségével. Ez annyit jelent, hogy a LanStore telepítésekor határozhatjuk meg a fent említett k és n számokat. Esetünkben a számok a következőt jelentik: n határozza meg a számítógépek számát, k pedig azt a számot amennyi gépnek minimálisan működnie kell. [2]

2 A fejlesztés háttere

Már a tavalyi diplomamunkákban is olvashattuk, hogy a LanStore bár működőképes, ahhoz hogy igazán használhatóvá váljon, még sok munkára van szükség. A fejlesztések célja a rendszer biztonságosabbá tétele, a fájlműveletek során fellépő hibák kijavítása, a LanStore driver továbbfejlesztése. Szükség volt még a LogService nevű modulra is, hogy a fejlesztés és tesztelés során a hibakeresés megfelelően támogatva legyen.

Mint már a bevezetőben leírtam, ahhoz, hogy tesztelhessük a rendszert, először el kell juttatni a megfelelő számítógépre, majd el kell indítani a programot. Ez egyáltalán nem egyszerű és rövid feladat egy elosztott architektúra esetén.

2.1 Előző megoldások

A 2006-os évfolyam elkészített egy grafikus felülettel rendelkező, WMI-n (Windows Management Instrumentation) alapuló megoldást. A WMI a Windows 2000 Professional SP2 operációs rendszerben jelent meg, de a többi Windows platform számára is elérhetővé tették. A .NET keretrendszer teljes körű támogatást biztosít hozzá. Ezt a kis alkalmazást RemoteCommandRunner-nek, vagy röviden RCR-nek nevezték el. [3] A 2007-es évfolyam egy új megoldást adott a problémára, amely nehezen használható, parancssoros felhasználói felülettel rendelkezett. Több probléma is felmerült használata során, ezért egy új, könnyen kezelhető, az összes terjesztéssel kapcsolatos feladatot ellátó programot kellet készíteni.

3 A kód terjesztése a hálózatban

A legegyszerűbben úgy tudjuk a programot és konfigurációs állományait eljuttatni a távoli számítógépre, hogy a célgép megosztását hálózati meghajtóként csatlakoztatjuk, majd rajta lokális fájlműveleteket végezünk. A feladatot a NetworkDrive osztály látja el. Mivel a .NET keretrendszer nem támogatja hálózati meghajtók csatlakoztatását, szükség van három, az mpr.dll fájlban implementált, unmanaged kódban található függvényre.

1. **WNetAddConnection** – Létrehozza a kapcsolatot a hálózati és a helyi erőforrás között. Konkrétan a 32 bites, ANSI kódolású WNetAddConnection2A változatot használtam fel. A függvény törzse:

```
DWORD WNetAddConnection2(
  _in LPNETRESOURCE lpNetResource,
  _in LPCTSTR lpPassword,
  _in LPCTSTR lpUsername,
  _in DWORD dwFlags
);
```

Az egyes paraméterek jelentése:

lpNetResource – Egy NETRESOURCE struktúrára, mely az erőforrásokat írja le.

```
typedef struct _NETRESOURCE {
  DWORD dwScope;
  DWORD dwType;
  DWORD dwDisplayType;
  DWORD dwUsage;
  LPTSTR lpLocalName;
  LPTSTR lpRemoteName;
  LPTSTR lpComment;
  LPTSTR lpProvider;
} NETRESOURCE;
```

Az egyes paraméterek jelentése:

dwScope – Alkalmazási terület

dwType – A hálózati erőforrás típusa (printer, diszk)

dwDisplayType - A Windows beépített böngészőablakára vonatkozó opciók

dwUsage – A csatolmány felhasználhatósága (csatolható, csak jelszóval)

lpLocalName – A csatolmány helyi megnevezése

lpRemoteName – A távoli erőforrás neve, (IP-cím, elérési út)

lpComment – A távoli erőforrás leírása

lpProvider – Az távoli erőforrás szolgáltatójának neve

lpPassword – A kapcsolat felépítéséhez használt jelszó

lpUsername – A kapcsolat felépítéséhez használt felhasználói név

dwFlags – Egyéb kapcsolók

2. **WNetCancelConnection** – Megszünteti a már felépített kapcsolatot a helyi és a hálózati erőforrások között. Itt is a 32 bites, ANSI kódolású WNetCancelConnection2A változatot használtam fel. A függvény törzse:

```
DWORD WNetCancelConnection2(
  _in LPCTSTR lpName,
  _in DWORD dwFlags,
  _in BOOL fForce
);
```

   Az egyes paraméterek jelentése:

   lpName – A távoli vagy a helyi erőforrás megnevezése

   dwFlags – A kapcsolat leírása (újra csatolandó, végleges lecsatolás)

   fForce – Ha igaz értékre állítjuk, akkor is elvégzi az erőforrás lecsatolását, ha fájlműveletek vannak folyamatban

3. **WNetRestoreConnectionW** – Helyreállítja a már felépített, de megszakadt kapcsolatot. Windows Vista operációs rendszeren nem támogatott. A függvény törzse:

```
DWORD WNetRestoreConnectionW(
_in  HWND hwndParent,
_in  LPCWSTR lpDevice,
     BOOL fUseUI
);
```

   Az egyes paraméterek jelentése:

   hwndParent – A hívó alkalmazás ablakának azonosítója

   lpDevice – A helyi meghajtó betűkódja (Z:)

   fUseUI – Ha igaz értéket kap, felhasználónév és jelszó ablakot dob fel a Windows

3.1 A NetworkDrive osztály

A könnyebb használhatóság érdekében a fenti függvényeket a NetworkDrive osztály megfelelően paraméterezi, és a hálózati meghajtók csatolásához csak a távoli és a helyi elérési útvonalakra valamint, a felhasználói név és jelszó párosra van szükség.

1. Használatához először létre kell hoznunk egy új példányt, majd állítsuk be a helyi gépen használni kívánt meghajtó betűjelét:

   .LocalDrive = tetszőleges, nem használt meghajtó pl.: "Z:"

2. Adjuk meg a célgép nevét, vagy IP címét, valamint a megosztás elérési útvonalát

   .ShareName = "\\\\" + Szerver neve, vagy IP címeName + "\\" + megosztás;

3. Csatoljuk fel a hálózati meghajtót. Használhatunk felhasználó nevet és jelszót is.
   .MapDrive(felhasználói név, jelszó);
   .MapDrive();
4. Végezzük el a szükséges fájlműveleteket.
5. Csatoljuk le a meghajtót.
   .UnMapDrive();

4 Szerver komponens irányítása

Mindössze két nagyon hasonló feladatot kellett megoldani. A szerver feladatát ellátó program elindítását és leállítását a távoli számítógépen. A hálózati meghajtóval ellentétben a .NET keretrendszer támogatja a szükséges funkciókat. A háttérben RPC (Remote Procedure Call) hívások történnek.

4.1 Az RPC

Az RPC-t 1976-ban definiálták az RFC 707 szabványban. Először a Xerox használta üzleti alkalmazásaiban 1981-ben Courier néven. UNIX rendszeren a SUN alkalmazta az NFS (Network File System) alapjaként. Ma ezt ONC RPC (Open Network Computing Remote Procedure Call) néven ismerjük. Egy másik UNIX-os megvalósítás az Apollo Computer nevéhez fűződik, amely a DCE (Distributed Computing Environment) keretrendszer fejlesztéséhez használta fel a kliens-szerver kommunikációban, a 1990-es évek elején. [4]

A Microsoft ezt, az eredetileg nyílt forráskódú DCE technológiát módosította és kibővítette UNICODE támogatással. Ez adja a DCOM (Distributed Component Object Model) hálózati kommunikációs technológia alapját.

4.2 Folyamatkezelés a .NET keretrendszerben

A .NET keretrendszer System.Management csomagja tartalmazza a szükséges eljárásokat és elemeket. A csomag a WMI (Windows Management Instrumentation) objektumaira épül.

4.2.1 A WMI (Windows Management Instrumentation)

A WMI a WBEM (Web-based Enterprise Management) szabvány Microsoft-féle megvalósítása, amelynek segítségével egységes módon érhetők el a Windows operációs rendszerek különféle objektumai. A WMI technológia biztosítja a hálózatfelügyeleti szoftverek számára szükséges infrastruktúrát. Kifejlesztésének célja elsősorban az volt, hogy egységes keretet (és felületet) biztosítson a már létező felügyeleti technológiáknak (SNMP, DMI, stb.).

A WMI első verziója a Windows NT SP4-ben jelent meg, de az SMS 2.0 verziója már teljes egészében erre épül. Jelentős különbségek vannak azonban a Windows 2000-ben, az XP-ben és a Windows Server 2003-ban található változatok között. A legfontosabb különbség az, hogy a későbbi WMI változatok egyre több írható tulajdonságot tartalmaznak, vagyis lehetőséget adnak nem csak a rendszerjellemzők lekérdezésére, hanem beállítására is. WMI-re alapuló rendszerfelügyeleti megoldásokat természetesen nem csak a Microsoft, hanem számos más szoftvercég palettáján is találhatunk.

A WMI a CIM (Common Information Model) segítségével jeleníti meg az operációs rendszer felügyelt objektumait. Felügyelt objektum lehet bármelyik szoftver, hardvereszköz, logikai vagy fizikai komponens. A WMI osztályai különböző névterekhez tartoznak annak megfelelően, hogy melyik rendszerterületet jelenítik meg. A névterek szervezése hierarchikus, a fa gyökere a root névtér. Az egyes osztályok útvonalának megadásakor először is meg kell határoznunk az a számítógépet, amelyik a CIM Repository-t tartalmazza, majd sorban meg kell adnunk a hozzá vezető névtér-hierarchia elemeit. [5]

4.2.2 Kapcsolódás távoli számítógéphez

A kapcsolatot a ManagementScope objektum hozza létre.

```
public ManagementScope(
    string path,
    ConnectionOptions options
)
```

Az egyes paraméterek jelentése:

path – A célszámítógép neve és a megfelelő WMI névtér

options – ConnectionOptions objektum, mely a kapcsolat paramétereit tartalmazza.

A folyamatok kezeléséhez a „cimv2” névtérre lesz szükségünk, ezért a következő formában kell létrehozni az objektumot:

```
new ManagementScope("\\\\" + computer +\\root\\cimv2", options);
```

Az *options* paraméter ConnectionOptions típusú objektumának számunkra csak a felhasználó név és jelszó adattagja lényeges. Ezután nincs más dolgunk, mint meghívni a Connect metódust.

4.2.3 Program indítása távoli számítógépen

Ahhoz, hogy egy programot elindítsunk a másik számítógépen először szükségünk van a feladatot ellátó ManagementClass objektumra.

```
public ManagementClass(
    ManagementScope scope,
    ManagementPath path,
    ObjectGetOptions options
)
```

Az egyes paraméterek jelentése:

scope – az előző pontban ismertetett ManagementScope objektum

path – ManagementPath objektum, amely a megfelelő Win32_Process ManagementClass osztály elérési útvonalát reprezentálja:

```
    new ManagementPath("Win32_Process");
```

options – ObjectGetOptions objektum, értéke ebben az esetben „null”

A program elindításához meg kell hívnunk az InvokeMethod metódust:

```
public ManagementBaseObject InvokeMethod(
    string methodName,
    ManagementBaseObject inParameters,
    InvokeMethodOptions options
)
```

Az egyes paraméterek jelentése:

methodName – A végrehajtandó függvény neve, ebben az esetben „Create”

inParameters – ManagementBaseObject objektum, a hívandó függvény bemeneti paramétereit tartalmazza. A ManagementClass GetMethodParameters függvényével hozzuk létre, a konstruktor paramétere megegyezik a *methodName* paraméter értékével. Miután létrehoztuk, állítsuk be a következő két adattagot:

`inParams["CommandLine"]` = A programot elindító parancs (server.exe start)

`inParams["CurrentDirectory"]` = A futtatható állomány teljes elérési útvonala a távoli gépen (C:\Program Files\LanStore)

options – InvokeMethodOptions objektum, értéke ebben az esetben „null”

4.2.4 Program leállítása távoli számítógépen

Az ismertetett eljárás az összes adott nevű folyamatot leállítja, ami a LanStore esetében nem jelent problémát, hiszen minden számítógépen a szerver, ha fut, akkor pontosan egy példányban fut. Ehhez szükségünk van az egyes folyamatokat reprezentáló osztályokra, objektumokra. Erre szolgál az ObjectQuery osztály, melynek segítségével SQL utasításhoz hasonlóan leválogathatjuk a Win32_Process névtér megfelelő nevű folyamatot reprezentáló objektumait, a következő kéréssel:

```
SELECT*FROM Win32_Process WHERE Name='process_name'
```

Az így létrehozott lekérdezést a ManagementObjectSearcher osztály segítségével futtathatjuk. Használjuk a következő alakú konstruktort:

```
public ManagementObjectSearcher(
    ManagementScope scope,
    ObjectQuery query
)
```

Az egyes paraméterek jelentése:

scope – ManagementScope objektum, megegyezik a 4.2.2 pontban ismertetettel

query – ObjectQuery objektum, melyet futtatni szeretnénk

A kérés a célgépen a ManagementObjectSearcher Get metódusának hívásakor fut le, visszatérési értéke egy ManagementObject típusú objektumokat tartalmazó ManagementObjectCollection típusú objektum. A ManagementObject osztály példányai a WMI egyes elemeit reprezentálják, ebben az esetben a futó szerver program példányát. A program futása a ManagementObject következő metódusával állítható le:

```
public Object InvokeMethod(
    string methodName,
    Object[] args
)
```

Az egyes paraméterek jelentése:

methodName – A végrehajtandó függvény neve, ebben az esetben „Terminate”

args – A hívott függvény paramétereit tartalmazó tömb, értéke ebben az esetben „null”

4.2.5 Program futásának ellenőrzése

Az előző ponthoz hasonlóan kell eljárni, annyi különbséggel, hogy csak azt kell ellenőrizni, hogy a ManagementObjectSearcher Get metódusának hívásakor visszakapott ManagementObjectCollection hány elemet tartalmaz. Ha ez az érték nagyobb, mint nulla, akkor a szerver (legalább egy példányban) fut.

5 A RemoteComputer osztály

A RemoteComputer osztály feladata a szerver folyamat távoli gépen való kezelése, az előző pontban ismertetett osztályok és eljárások felhasználásával.

1. Használatához hozzunk létre egy új példányt, konstruktorát a célgép nevével, vagy IP címével paraméterezve.
2. A Kapcsolat létrehozásához szükségünk van egy megfelelő jogosultságokkal rendelkező felhasználó nevére és jelszavára.
3. Folyamat indításához használjuk a következő függvényt:
   ```
   public object Run(string commandLine, string workDir)
   ```
   Az egyes paraméterek jelentése:
   commandLine – A programot elindító parancs (server.exe start)
   workDir – A futtatható állomány teljes elérési útvonala a távoli gépen (C:\LanStore)
4. Egy program futását a következő hívással ellenőrizhetjük:
   ```
   public bool IsRunning(string processName)
   ```
   Az egyes paraméterek jelentése:
   processName – A program, folyamat neve (server.exe)
5. Folyamat leállításához használjuk a következő függvényt:
   ```
   public void KillAll(string processName)
   ```
   Az egyes paraméterek jelentése:
   processName – A program, folyamat neve (server.exe)

6 A ServerManager osztály

Az osztály a 3.1 és az 5. pontban bemutatott NetworkDrive és RemoteComputer osztályokat felhasználva látja el a telepítés és törlés, valamint az indítás és leállítás feladatát. A hívások visszatérési kódját szöveges üzenetté alakítja, amit lehetőség van egy tetszőleges TextBox control-ban megjeleníteni.

6.1 Consol

A Consol egy TextBox controlt jelent, melyben a ServerManager az elvégzett művelet és lépéseinek eredményei jelennek meg. Segíti a hibakeresést, visszajelzést kapunk a műveletek sikeres, vagy sikerttelen befejezéséről. Használata:

```
public void SetConsol(System.Windows.Forms.TextBox consol)
```

6.2 Kód terjesztése

A következő három alpontban ismertetett függvény feladata a futtatható és konfigurációs állományok eljuttatása, törlése, és meglétének ellenőrzése a távoli számítógépeken. Paraméterlistájuk több közös elemet mutat, ezért e mezők jelentését itt ismertetem:

serverName – A célgép neve, vagy IP címe

mapDriveAs – A távoli megosztást csatlakoztató hálózati meghajtó helyi betűjele

share – A távli gép megosztott könyvtára

installationFolder – A megosztáson belüli alkönyvtár, ahova installálni szeretnénk

useNamePassword - Értéke „true" vagy „false". Amennyiben „true" a hálózati meghajtót felhasználói név és jelszó felhasználásával kerül csatolásra.

userName – A megosztáshoz megfelelő jogosultságokkal rendelkező felhasználó neve

password - A megosztáshoz megfelelő jogosultságokkal rendelkező felhasználó jelszava

6.2.1 Telepítés

A következő függvény végzi a LanStoreServer.cfg konfigurációs állomány létrehozását és a fájlok célszámítógépre történő másolását:

```
public bool InstallServer(string serverName, string mapDriveAs,
string share, string sourceFolder, string strServerConfig, string
installationFolder, bool useNamePassword, string userName, string
password)
```

Az egyes paraméterek jelentése:

sourceFolder – A telepítésre szánt állományokat tartalmazó mappa elérési útvonala a helyi számítógépen

strServerConfig – A LanStoreServer.cfg konfigurációs állomány szöveges tartalma

6.2.2 Törlés

A következő függvény végzi a célszámítógépen lévő fájlok törlését:

```
public bool UninstallServer(string serverName, string mapDriveAs,
string share, string installationFolder, bool useNamePassword,
string userName, string password)
```

6.2.3 Ellenőrzés

Lehetőség van annak ellenőrzésére, hogy az adott célszámítógépen már telepítve van-e a LanStore szerver komponense:

```
public int CheckServerInstallation(string serverName, string
mapDriveAs, string share, string installationFolder, bool
useNamePassword, string userName, string password)
```

6.3 Szerver irányítása

A következő három alpontban ismertetett függvény feladata a szerver komponens indítása leállítása és a futás ellenőrzése a távoli számítógépeken. Paraméterlistájuk több közös elemet mutat, ezért e mezők jelentését itt ismertetem:

serverName - A célgép neve, vagy IP címe

userName – Távoli eljáráshívás kezdeményezésére jogosult felhasználó neve

password - Távoli eljáráshívás kezdeményezésére jogosult felhasználó jelszava

processName - A szerver program, folyamat neve (server.exe)

6.3.1 Indítás

A szerver indítását a következő függvény hívásával kérhetjük:

```
public bool StartServer(string serverName, string userName,
string password, string workDir, string processName, string
commandLine)
```

Az egyes paraméterek jelentése:

workDir – A futtatható állományt tartalmazó mappa elérési útvonala a távoli számítógépen

commandLine – A futást indító parancssori üzenet (server.exe start)

6.3.2 Leállítás

A következő függvény a szerver leállását eredményező üzenetet küld a célgépnek, majd egy másodpercig várakozik, és ellenőrzi, hogy a leállás sikeresen megtörtént-e.

```
public bool StopServer(string serverName, string userName, string
password, string processName)
```

6.3.3 Ellenőrzés

Annak ellenőrzésére, hogy a célszámítógépen a szerver már el lett-e indítva, vagy legalább egy példányban fut-e, a következő függvény használható:

```
public int CheckServerRunning(string serverName, string userName,
string password, string processName)
```

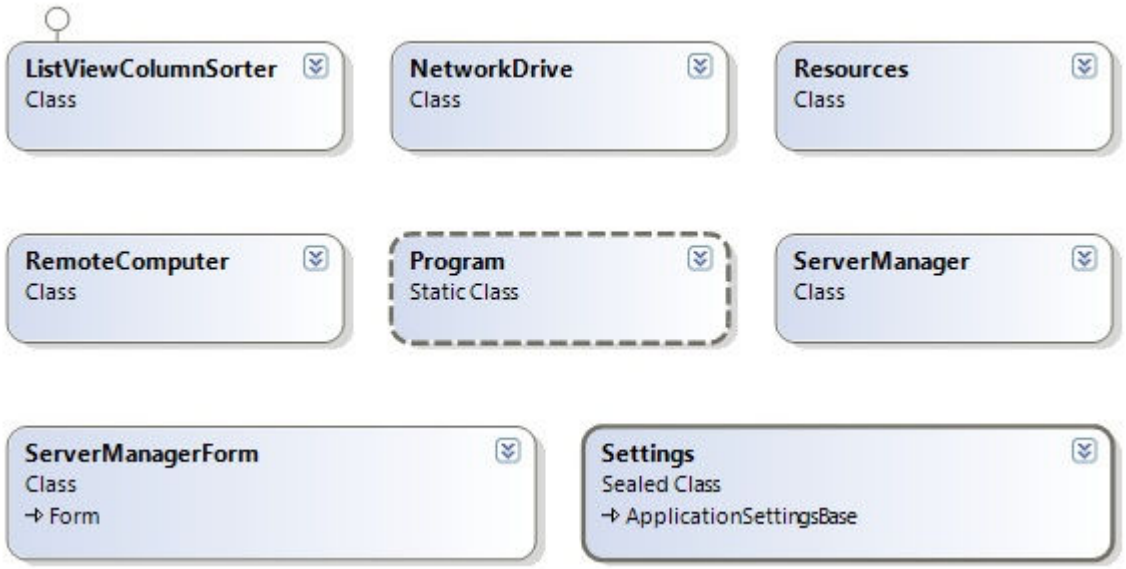
7 Az LSServerManager program

Mint azt már a bevezetőben említettem két grafikus felhasználó felülettel (Graphical user interface - GUI) rendelkező programot készítettem. Ezek közül az első egy Windows Form alkalmazás. Előnye, hogy a Windows Workgroup, vagy domain bármely számítógépén futtatható, mindössze a .NET keretrendszer 2.0 verziójára van szükség. Hátránya, hogy használatához a szerverekkel egy közös alhálózatban kell futtatni. Az LSServerManager úgynevezett Front-End program, feladata csupán az adatok beolvasása és megjelenítése. A szerverek irányítását az előző pontban ismertetett ServerManager osztály végzi. A program GUI alapja egy TabControl, amely három fület (Tab) tartalmaz. A következő alpontokban részletesen ismertetem ezek funkcióit és használatát.

Szükség van még egy, a szerverek nevét és elérhetőségét tartalmazó adatbázisra. A megbeszélések során felmerült az ötlet, hogy ne egy fix listával dolgozzunk, hanem egy erre a célra kifejlesztett program, vagy modul automatikusan derítse fel a hálózat szerkezetét. Ezt elvetettük, mert a ráfordítandó energia nem lenne arányban a kapott előnyökkel. Egy másik lehetőség, hogy a szerverek címét egy Active Directory-ból olvassuk ki. Ehhez azonban szükség van egy Domain Controller-re és Active Directory-ra. Az egyetem tesztelésre és fejlesztésre szánt laborjában a számítógépek csupán Windows Workgroup, nem pedig domain tagjai, így bár ez egyszerűbb és életképesebb megoldásnak tűnik szintén nem járt volna semmilyen előnnyel.

Maradt tehát egy statikus adatbázis, lista, melynek szerepét egy egyszerű szöveges állomány tölti be. A szerverhez való kapcsolódáshoz szükséges felhasználó neve és jelszava kódolás nélkül szerepelnek a fájlban, de ez a használat jellegéből kifolyólag nem jelenthet biztonsági kockázatot.

7.1 Osztály diagram



1. ábra Server Manager – Osztály diagram

7.2 Szerverlista

A szöveges állomány minden sora egy szerver adatait tartalmazza. Minden sor öt mezőt tartalmaz, vesszővel elválasztva. Az egyes mezők jelentése sorrendben a következő:

1. Szerver neve, vagy IP címe
2. A megfelelő jogosultságokkal rendelkező felhasználó neve
3. A megfelelő jogosultságokkal rendelkező felhasználó jelszava
4. A szervert tartalmazó mappa elérési útvonala a távoli számítógépen
5. A szerver azonosító száma (ID)

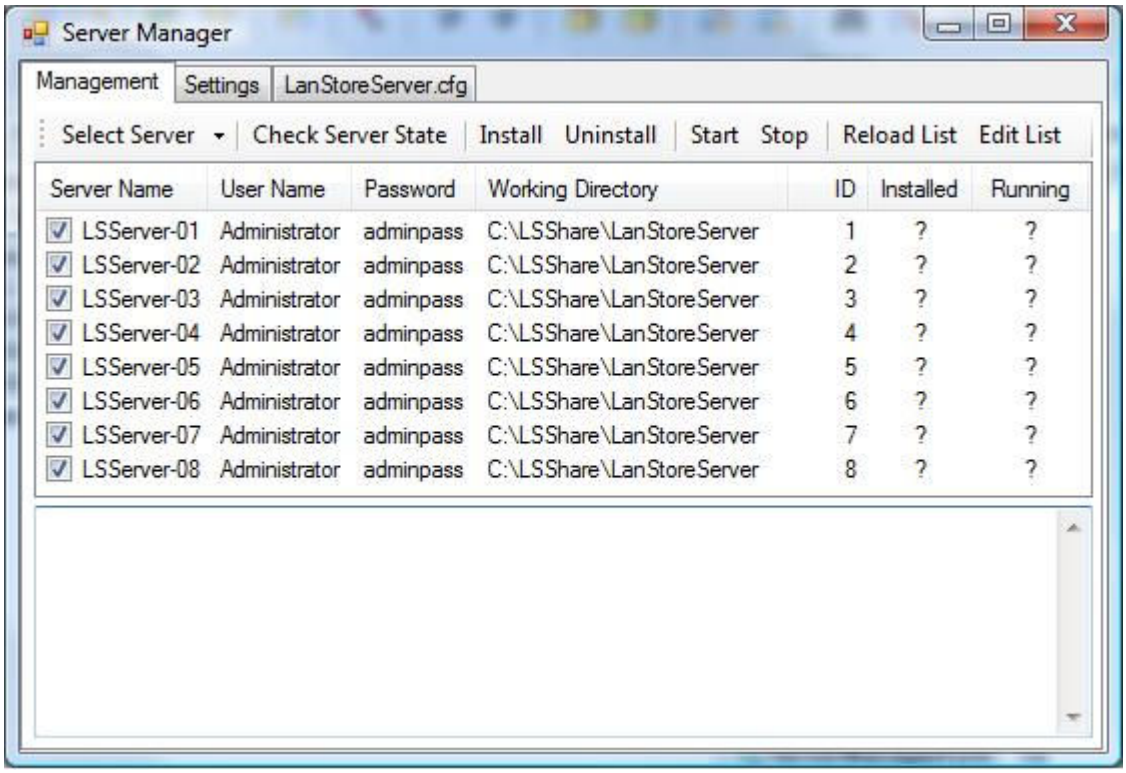
Példa az adatbázis egy sorára:

```
LSServer-01,Administrator,adminpass,C:\LSShare\LanStoreServer,1
```

7.3 Szerverek kezelése

A Management fül szolgál a szerverek irányítására. A fül felső része tartalmazza a kezelőgombokat, alatta a szerverlista, majd egy TextBox, mely a ServerManager osztály Consol (6.1 pont) funkcióját tölti be.

Minden szerver egyedileg kezelhető, de egyszerre több példányon azonos műveletet is végezhetünk. A „Server Name” mezőben jelöljük ki azokat a szervereket, majd kattintsunk a felső sorban található műveleti gombra. Az egyes gombok funkcióját, valamint a lista oszlopainak jelentését a következő alpontokban ismertetem.



2. ábra Server Manager - Management fül

7.3.1 Kezelőgombok

Select Server – Legördülő menüvel segíti a megfelelő szerverek kiválasztását

Select All – Minden szerver kiválasztása a listában

Deselect All – Az összes kijelölés törlése

Change Selection – Kijelölések felcserélése (kiválasztott <=> nem kiválasztott)

Select Running – Az éppen futó szerverek kijelölése

Select Installed – A telepített szerverek kijelölése

Check Server State – Szerver futásának és telepítésének ellenőrzése

Install – Szerver telepítése

Uninstall – Szerver törlése

Start – Szerver indítása

Stop – Szerver leállítása

Reload List – Szerverlista újbóli beolvasása, frissítése

Edit List –Szerverlista szerkesztésre, Notepad programmal

7.3.2 Szerverek listája és állapota

Server Name - Szerver neve, vagy IP címe

User Name - A megfelelő jogosultságokkal rendelkező felhasználó neve

Password - A megfelelő jogosultságokkal rendelkező felhasználó jelszava

Working Directory - A szervert tartalmazó mappa elérési útvonala a távoli számítógépen

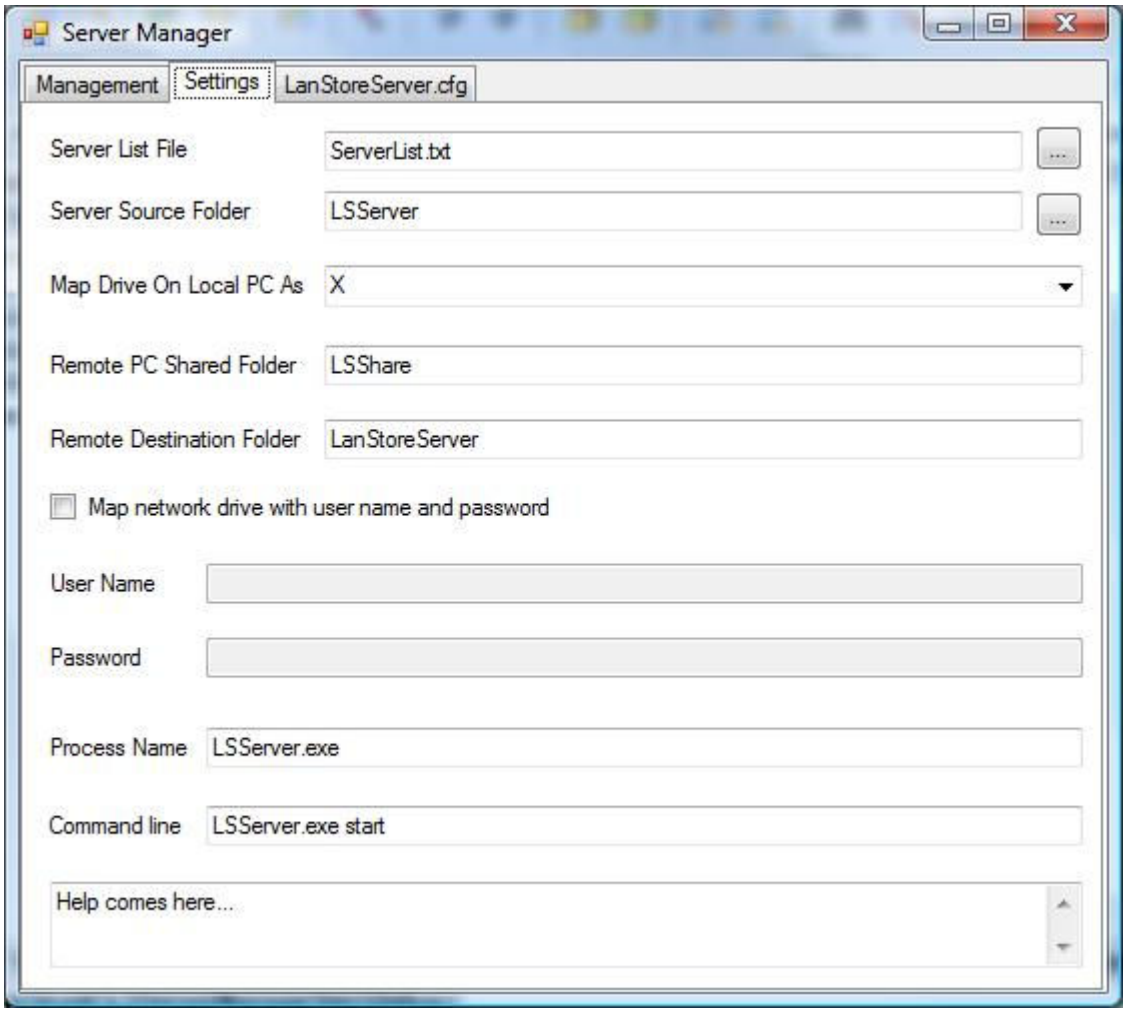
ID - A szerver azonosító száma

Installed –Telepítés állapota (ismeretlen állapotú szerverek esetében kérdőjel)

Running – Futás állapota (ismeretlen állapotú szerverek esetében kérdőjel)

7.4 Beállítások

A Settings fülön végezhetjük el a program beállításait. Az itt megadott paraméterek globálisan vonatkoznak az összes szerverre. A fül alján tartalmaz egy rövid, tömör leírást az egyes mezők jelentéséről.



3. ábra Server Manager - Settings fül

Az egyes mezők jelentése:

Server List File – A 7.1 pontban ismertetett szerver lista elérési útvonala

Server Source Folder – A szervert és állományait tartalmazó mappa elérési útvonala

Map Drive On Local PC As – A felcsatolt hálózati meghajtó betűjele

Remote PC Shared Folder – Megosztott könyvtár neve a távoli gépen

Remote Destination Folder – A telepítési alkönyvtár elérési útvonala a megosztáson belül

Map network drive with user name and password – Kijelölése esetén a program, a távoli gép megosztását, felhasználói név és jelszó alkalmazásával csatlakoztatja

User Name – A megosztás csatolásához használt felhasználói név

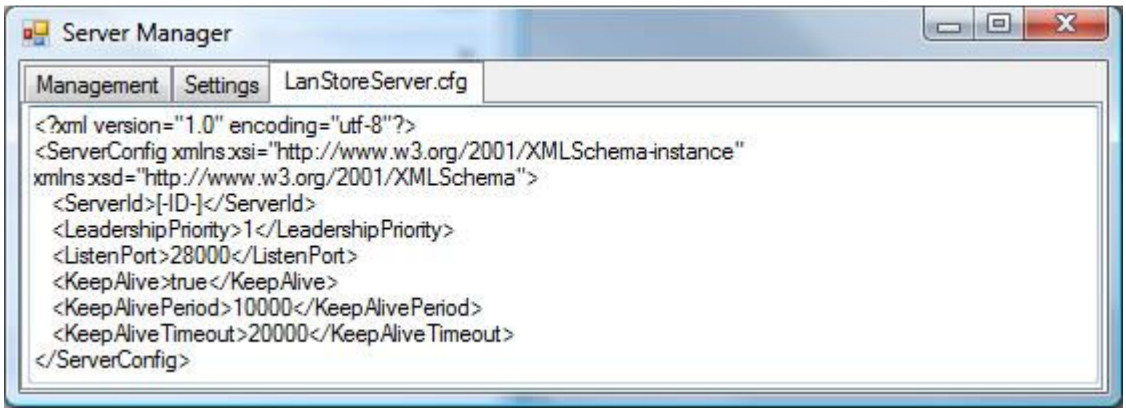
Password – A megosztás csatolásához használt felhasználó jelszava

Process Name – A szerver folyamat neve

Command Line - A szerver indítását elvégző parancssori utasítás

7.5 LanstoreServer.cfg konfigurációs fájl

A szerver indulásakor beolvassa a LanstoreServer.cfg fájlt, amely a beállításait tartalmazza. Általános esetben ez minden szervernél azonos a ServerID, azaz az azonosító szám kivételével. Ennek minden szerver esetében egyedinek kell lennie. A program a LanstoreServer.cfg fülön megadott formájú konfigurációs fájlt készít minden szerver számára, a telepítés során. Az egyedi azonosító számokat a 7.1 pontban ismertetett adatbázis fájl tartalmazza.



4. ábra Server Manager – LanStoreServer.cfg fül

A telepítési folyamat közben a Server Manager létrehoz egy LanstoreServer.cfg szöveges fájlt, majd átmásolja a beállításokat, úgy, hogy a ServerID értékét „[-ID-]” értékről az adatbázisban megadott azonosítóra cseréli.

8 Az LSWebManager program

Az LSWebManager és az LSServerManager azonos kódbázisra épül, de nem Windows Form, hanem web-es alkalmazás. Szintén Front-End program, a szerverek kezelését a háttérben szintén a ServerManager osztály látja el, mint a LSServerManager esetében. Előnye, hogy futtatásához a felhasználói oldalon csak egy böngészőre van szükség, és nem kell a szerverekkel egy közös alhálózatban tartózkodnunk. Hátránya, hogy használatához szükség van web és alkalmazás szerverre, valamint csak a Domain Controller felügyelete alá tartozó számítógépek kezelhetőek. A fejlesztéshez Microsoft IIS (Internet Information Services) 6.0-ás változatát és ASP .NET 2.0-ás verzióját alkalmaztam.

8.1 ASP .NET

Mikor 2001-ben megjelent az ASP.NET 1.0, a platform egyik legfontosabb célja az volt, hogy elfedje a programozók elől a felhasznált HTTP protokoll részleteit és szerver oldali kontrollok segítségével a webalkalmazások fejlesztését a lehető legközelebb vigye a jól ismert vastag kliens fejlesztéshez. Az időközben a Web 2.0 megjelenésével jelentkező igények gyökeresen új szemléletet vezettek be a webprogramozásban, amelynek követése számos új technológia (XMLHttpRequest, JavaScript, DHTML, DOM, XML, JSON) elsajátítását igényli. A Microsoft webfejlesztői platformja úgy igyekszik mindezzel lépést tartani, hogy a programozók továbbra is a már ismert környezetükben, az ismert eszközöket és módszereket használva felelhessenek meg az új kihívásoknak.

Már az ASP.NET legelső verziója új alapokra helyezte az adatkezeléssel kapcsolatos funkciók webes környezetben történő megvalósítását azáltal, hogy a vezérlőelemek bevezetésével levette a fejlesztők válláról a HTML kód előállításának terheit. A 2.0 verzió egyik legfontosabb célja, hogy az adatok megjelenítésével és szerkesztésével kapcsolatos kódok előállításához szükséges időt a lehető legrövidebbre csökkentse, miközben rugalmas, tetszőlegesen testre szabható, deklaratív leírásokon alapuló rendszert ad a webfejlesztő kezébe. [6]

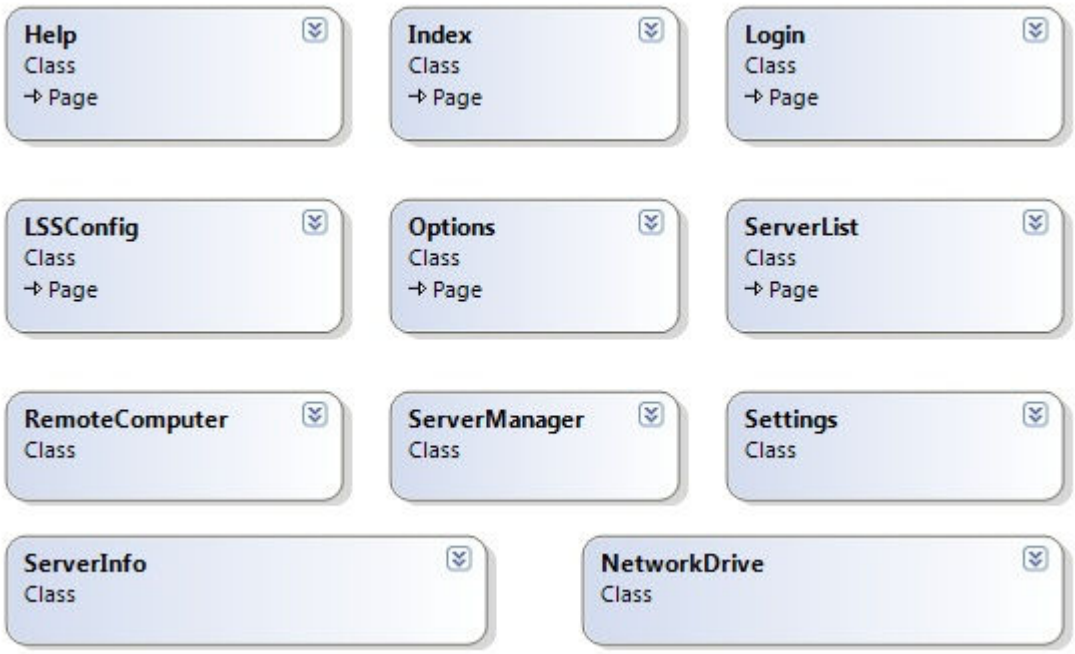
8.2 A program telepítése

A fejlesztéshez a Microsoft Visual Studio 2005-ös verzióját használtam. Jobb egérgombbal kattintsunk A Solution explorer ablakban az LSWebManager projectre, majd válasszuk a „Publish…” opciót. A megjelenő ablakban adjuk meg a célmappát, majd kattintsunk a „Publish” gombra. A webszerver C:\Inetpub\wwwroot mappájában hozzunk létre egy tetszőleges nevű mappát, majd másoljuk át a fájlokat ebbe az új mappába.

Szükség van még egy konfigurációs állományra, a LSWebManager.settings.txt nevű szöveges fájlra. Ezt alapértelmezésben a program a C:\ meghajtó gyökerében keresi. A fájl megtalálható a project mappában. Továbbra is szükséges az elérhető szerverek neve (8.8 pont) és a futtatható állományokat tartalmazó mappa, valamint a LanstoreServer.cfg konfigurációs fájl. (8.7 pont)

Nyissuk meg a szerver Internet Information Services (IIS) Manager alkalmazását, és keressük meg a „Web Sites” mappában az alkalmazásunkat. Kattintsunk a mappára jobb egérgombbal és válasszuk a „Properties” opciót. A megjelenő ablakban válasszuk ki az „ASP .NET” fület, majd az „ASP .NET version” értékét állítsuk 2.x-re. Ha nincs ilyen lehetőség telepítsük a .NET Framework 2-es verzióját.

8.3 Osztály diagram



5. ábra LSWebManager – Osztály diagram

8.4 Biztonság

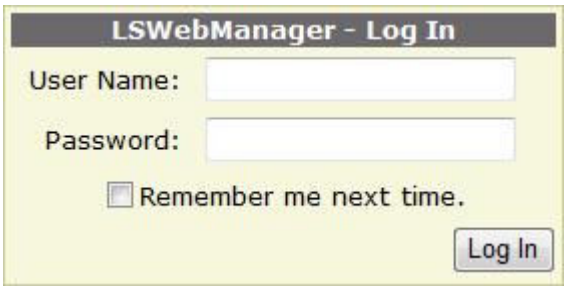
Ahhoz, hogy kihasználjuk a web-es felület nyújtotta előnyöket, hozzáférhetővé kell tenni a programot az alhálózaton kívül is. Ez komoly biztonsági kockázatot jelent, ezért szükség van legalább egy minimális védelemre. Ezt egy bejelentkező ablakkal oldottam, meg, amely csak a megfelelő felhasználói név és jelszó megadása után engedélyezi a program használatát.

8.4.1 Authentikáció

Az authentikáció kikapcsolható a Web.config konfigurációs állomány módosításával. A következő kódrészlet törlése, vagy „ki-kommentezése” után azonosítás nélkül érhető el a program valamennyi funkciója:

```
<authorization>
     <deny users="?" />
</authorization>
```

Amennyiben az authentikáció aktív, az alább látható bejelentkező ablak fogad minket. Az ablak az ASP .NET keretrendszer része, kódja a Login.aspx fájlban található.



6. LanStore Web Manager – Bejelentkező ablak

8.4.2 Felhasználók

A felhasználók neve és jelszava is a Web.config állományban található, a következő formában:

```
<user name="name" password="9dd4e461268c8034f5c8564e155c67a6" />
```

Minden sor egy felhasználó nevét és jelszavát tartalmazza, a jelszót MD5 algoritmussal kell kódolni.

8.5 Szerverek kezelése

A Home oldal szolgál a szerverek irányítására. Felépítése nagyon hasonlít az LSServerManager Management fülének felépítésére. Az oldal felső része tartalmazza a kezelőgombokat, alatta a szerverlista, majd egy TextBox következik, mely a ServerManager osztály Consol (6.1 pont) funkcióját tölti be.

Minden szerver egyedileg kezelhető, de lehetőség van egyszerre több példányon azonos műveletet elvégezni. A lista első oszlopában válasszuk ki azokat a szervereket, melyeken az adott műveletet el szeretnénk végezni, majd kattintsunk a felső sorban található műveleti gombra. Az egyes gombok funkcióját, valamint a lista oszlopainak jelentését a következő alpontokban ismertetem

7. ábra LanStore Web Manager - Home

8.5.1 Funkciógombok

Change Selection – Kijelölések felcserélése (kiválasztott <=> nem kiválasztott)

Refresh – Szerver futásának és telepítésének ellenőrzése

Install – Szerver telepítése

Uninstall – Szerver törlése

Start – Szerver indítása

Stop – Szerver leállítása

8.5.2 Szerverek listája és állapota

Az egyes oszlopok jelentése:

ID - A szerver azonosító száma

Name - Szerver neve, vagy IP címe

Installed –Telepítés állapota (ismeretlen állapotú szerverek esetében kérdőjel)

Running – Futás állapota (ismeretlen állapotú szerverek esetében kérdőjel)

8.6 Beállítások

A beállítások a LSWebManager.settings.txt nevű konfigurációs fájlban tárolódnak. A fájlt a program a C:\ meghajtó gyökerében keresi. Az elérési útvonalat a Settings osztály MANAGER_CONFIG_FILE_PATH adattagja tartalmazza. Az osztály első hivatkozásakor, a konstruktorból hívódik meg a Refresh metódus, amely inicializálja a publikus adattagokat. A fájl az Options oldalon, szabadon szerkeszthető. Módosítás után a változtatásokat a „Save” gombra kattintva véglegesíthetjük. Lehetőség van egysoros kommentekkel rövid leírást adni az egyes mezők szerepéről, az ilyen leíró sorokat # (kettőskereszt) jellel kezdjük.

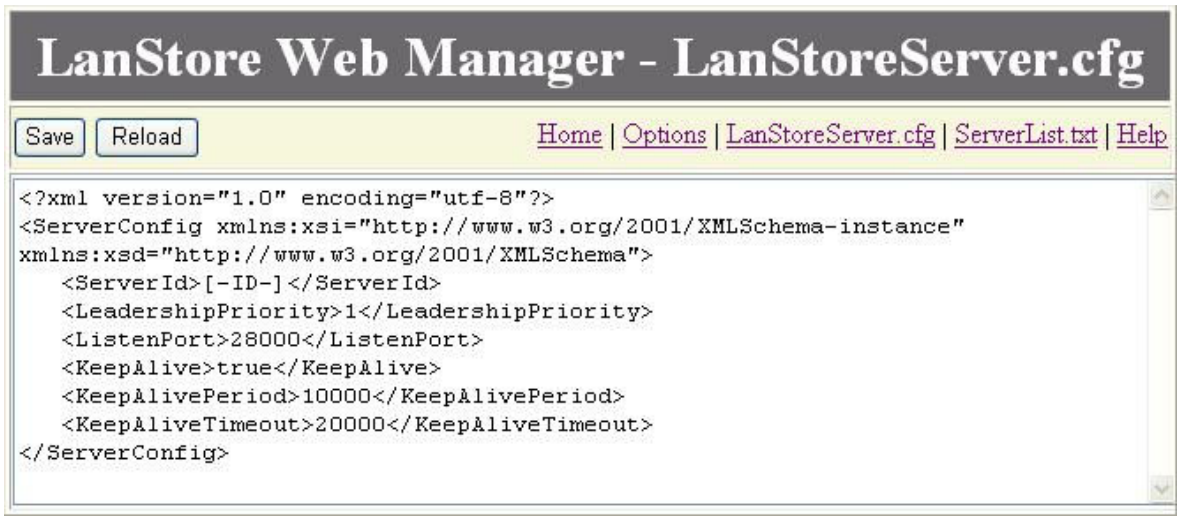
8. ábra LanStore Web Manager - Options

Az LSWebManager.settings.txt konfigurációs fájl mezőinek leírása	
SERVER_LIST_FILE_PATH	A szerver lista (8.8) elérési útvonala
SERVER_CONFIG_FILE_PATH	A LanstoreServer.cfg (8.7) elérési útvonala
SERVER_SOURCE_FOLDER	A szervert tartalmazó, helyi mappa útvonala
SERVER_REMOTE_USER	Megfelelő (RPC hívások) jogosultságokkal rendelkező felhasználó neve
SERVER_REMOTE_PASSWD	Megfelelő (RPC hívások) jogosultságokkal rendelkező felhasználó jelszava
SERVER_WORKING_DIR	A szervert tartalmazó mappa elérési útvonala a távoli számítógépen
SERVER_PROCESS_NAME	A szerver folyamat neve
SERVER_COMMAND_LINE	A szerver indítását végző utasítás
MAP_NETWORK_DRIVE_AS	A felcsatolt hálózati meghajtó betűjele
MAP_NETWORK_DRIVE_WITH_AUTH	„true” érték esetén a program, a távoli gép megosztását, felhasználói név és jelszó alkalmazásával csatlakoztatja
MAP_NETWORK_DRIVE_USER	A megosztás csatolásához használt név
MAP_NETWORK_DRIVE_PASSWD	A megosztás csatolásához használt jelszó
REMOTE_SHARED_FOLDER	Megosztott könyvtár neve a távoli gépen
REMOTE_INSTALL_FOLDER	A telepítési alkönyvtár elérési útvonala a megosztáson belül

1. táblázat Az LSWebManager.settings.txt konfigurációs fájl mezői

8.7 LanstoreServer.cfg konfigurációs fájl

A fájl felépítése és az egyedi azonosító előállítása megegyezik a 7.5 pontban ismertetettel. Ebben az esetben azonban szükség van egy mintafájlra is, melyhez a webszerver írási és olvasási jogokkal rendelkezik. A fájl a LanstoreServer.cfg oldalon, szabadon szerkeszthető. Módosítás után a változtatásokat a „Save” gombra kattintva véglegesíthetjük.



9. ábra LanStore Web Manager - LanstoreServer.cfg

8.8 Szerverlista

Az LSServerManager programhoz hasonlóan szükség van az elérhető szerverek nevére, vagy IP címére. Az adatbázis ebben az esetben is egy szöveges állomány, de felépítése egyszerűbb, mint az előző esetben. A fájl minden sora egy szerver adatait tartalmazza. Minden sor két mezőt tartalmaz, vesszővel elválasztva. Az egyes mezők jelentése sorrendben a következő:

1. Szerver neve, vagy IP címe
2. A szerver azonosító száma (ID)

A lista a ServerList.txt oldalon, szabadon szerkeszthető. Módosítás után a változtatásokat a „Save” gombra kattintva véglegesíthetjük.

10. ábra LanStore Web Manager - ServerList.txt

9 Felmerült problémák és megoldásuk

9.1 Windows Workgroup csoporttagság esetén hiba lép fel a szerver indításakor

Ennek egyik oka lehet, hogy a Windows XP Professional alap beállítása szerint, minden RPC hívást Guest jogosultságokkal próbál futtatni. A hiba javítható, ha a következő kulcsot 1-ről 0-ra állítjuk a regisztrációs adatbázisban [7]:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\
    ForceGuest
```

9.2 Az LSWebManager a felhasználó azonosítása után hibát dob

Pontos okát nem sikerült kiderítenem, de a hiba javítható. Futtassuk a programot a Visual Studio segítségével, majd a project mappában létrejött App_Data könyvtárat és tartalmát másoljuk a webszerver LSWebManager-t tartalmazó mappájába.

10 Elért eredmények

Mindkét általam készített program rendelkezik előnyökkel és hátrányokkal, de úgy érzem jelentősen megkönnyítik a későbbi fejlesztők munkáját. Az LSServerManager kitűnően alkalmazható Windows Workgroup környezetben, amilyen jelenleg az egyetemi CISCO laborban található. A LSWebManager igaz több erőforrást, webszervert igényel, de megfelelően konfigurálva lehetővé teszi, hogy a fejlesztők akár otthonról is telepítsék és kipróbálják a szerver komponenst.

A diplomadolgozatomban igyekeztem részletesebb leírást adni a programok használatáról, remélem, munkámmal megkönnyítettem a projekten dolgozó következő generációk számára a LanStore továbbfejlesztését!

Irodalomjegyzék

[1] Roszik György Attila. *A LanStore elosztott tárolást támogató rendszer továbbfejlesztése.* Diplomadolgozat 2006.

[2] Csepregi Péter Sándor. *Elosztott tárolást támogató rendszer fejlesztése.* Diplomadolgozat 2006.

[3] Sebesi Sándor. *Verziókezelés megvalósítása a LanStore rendszerben*. Diplomamunka 2006.

[4] Wikipedia. *Remote procedure call*
http://en.wikipedia.org/wiki/Rpc

[5] Wikipedia. *Windows Management Instrumentation*
http://en.wikipedia.org/wiki/Windows_Management_Instrumentation

[6] Magyarországi Web Konferencia. *Microsoft és az AJAX – ASP.NET alkalmazások AJAX-osítása*
http://web.conf.hu/2007/program/i/aspnetajax

[7] Google Groups. *microsoft.public.scripting.wsh*
http://groups.google.com/group/microsoft.public.scripting.wsh/msg/dc3a8490b8e68a62

Programkód megírásához felhasznált irodalom

1. Microsoft TechNet. *A WMI használata a .NET programokból.*
2. MSDN. *Microsoft Win32 and COM Development*
http://msdn.microsoft.com/en-us/library/aa139672.aspx
3. MSDN. *ASP.NET Developer Center*
http://msdn.microsoft.com/en-us/asp.net/default.aspx
4. aejw.com *NetworkDrive*
http://www.aejw.com/
5. SharpDevelop. *RemoteComputer*

Nyilatkozat

Alulírott Buczkó Tamás László, programtervező matematikus szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem Informatikai Tanszékcsoport Szoftverfejlesztés tanszékén készítettem, programtervező matematikus diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy diplomamunkámat a Szegedi Tudományegyetem könyvtárában, a kölcsönözhető könyvek között helyezik el.

Szeged, 2008. május 11.

Aláírás

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek Bilicki Vilmosnak, áldozatos munkájáért.

Mellékletek

1. DVD - A

   Windows Server 2003 - Microsoft Virtual PC (windows-dc-iis)

   Forráskód

2. DVD – B

   Windows XP - Microsoft Virtual PC (wgroup-server-1)

   Windows XP - Microsoft Virtual PC (domain-server-1)