Ismerkedés a Hyper-V programozási lehetőségeivel

Virtualizációs technológiák és alkalmazásaik (VIMIAV89) házi feladat

Nagy Dániel Dávid

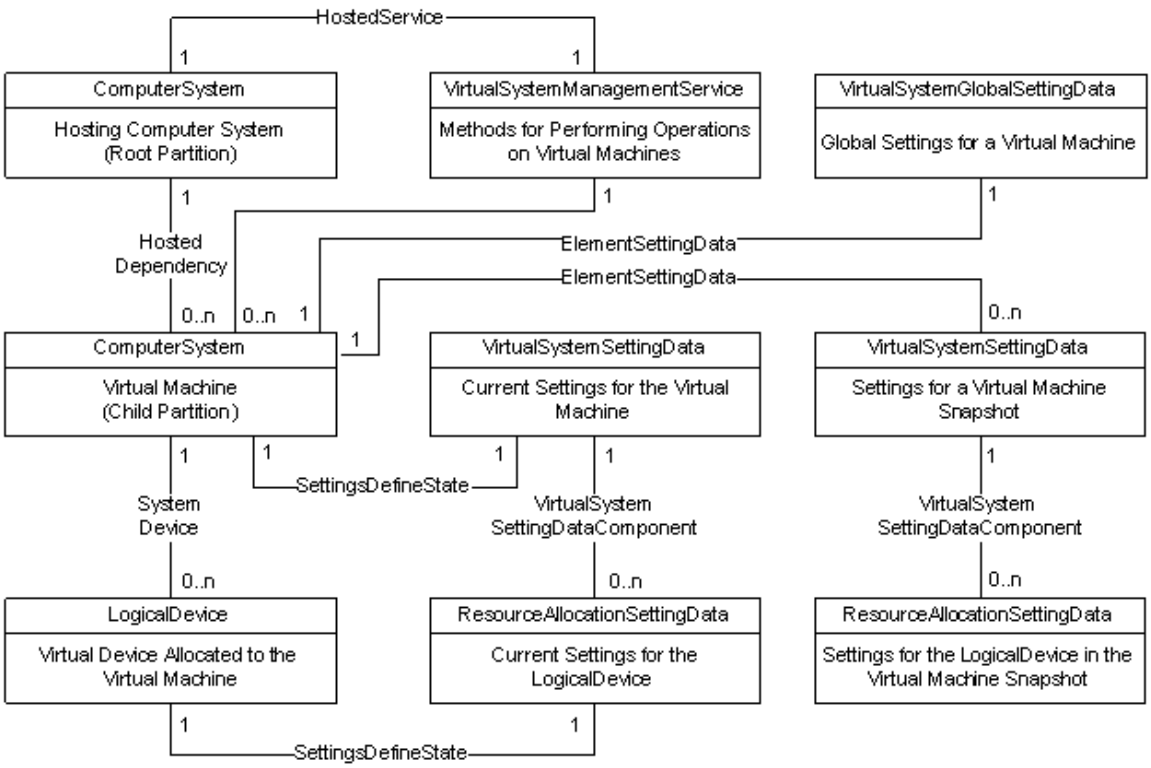
2011.12.07.

Bevezetés, a feladat áttekintése

A Hyper-V a Microsoft második szerveroldali virtualizációs platformja, ami a Virtual Server 2005 leváltására jelent meg, nem sokkal a Windows Server 2008 kiadása után. Az újdonságok részletes, mindenre kiterjedő ismertetése túlmutat e beszámoló keretein, elmondható azonban, hogy a rendszer programozási interfésze (Application Programming Interface, API) is komoly változásokon ment keresztül.

A Virtual Server 2005 funkcióit a Microsoft előző generációs komponenstechnológiája, a Component Object Model (COM) segítségével szólíthattuk meg saját alkalmazásainkból, illetve szkriptjeinkből. Bár a COM sok területen bevált, helyenként máig is használt technológia, amely több programozási nyelvet, illetve elosztott működést – a Distributed COM (DCOM) formájában – is támogat, a modern Microsoft környezetek, mint amilyen a .NET platform, illetve a kifejezetten adminisztrációs feladatok automatizálására szánt PowerShell, csak nagyon nehézkesen teszik lehetővé használatát. Különösen igaz ez a Virtual Server 2005 interfészére, amelynek mind biztonsági, mind szálmodellje eltér a COM világában általánostól.

A Hyper-V interfésze nem csupán a modern Microsoft menedzsment környezetbe illeszkedik jobban, de a más platform használatában jártasaknak is ismerősebb felépítéssel rendelkezik, mint elődje. A rendszer megszólítása ugyanis COM helyett a Windows Management Instrumentation (WMI) technológiával történik, objektummodellje pedig a platformfüggetlen Common Information Model (CIM) kiterjesztése (1. ábra). Parancssori hívásokat alkalmazásfejlesztés vagy szkript írás nélkül is kezdeményezhetünk a *wmic.exe* eszközzel, összetettebb feladatokat pedig a modernebb, .NET keretrendszerre építő PowerShell segítségével valósíthatunk meg, ami kifejezetten WMI-specifikus parancsokkal (*cmdlet*-ekkel) is rendelkezik – mint amilyen például a *Get-Wmiobject*. Mi több, önálló .NET alkalmazást is készíthetünk a Hyper-V kezelésére, mivel a keretrendszer külön kiterjesztések telepítése nélkül is támogatja a WMI lekérdezések, illetve metódushívások végrehajtását.



1. ábra: Virtual System Management Service [1]

Látható tehát, hogy a Hyper-V gazdag programozási lehetőségekkel rendelkezik, számos eszköz közül választhatunk. A továbbiakban ismertetem, melyik konkrét megközelítésben mélyültem el behatóbban, illetve milyen alkalmazást készítettem a segítségével. Végül összegzem a fejlesztés tapasztalatait.

A feladat pontosítása, célok

A Virtual Server 2005-öt gyakran érte kritika kezelőfelületének sajátos megvalósítása miatt: a már bemutatott COM interfészen kívül ugyanis kizárólag HTML alapú, webes adminisztrációs felület volt elérhető, mi több, a rendszer telepítése megkövetelte a Microsoft webszervere, az Internet Information Services (IIS) meglétét a host számítógépen e felület futtatása céljából – ami dedikált gazdagép esetén kifejezetten erőforráspazarló megközelítésnek bizonyulhat.

Vitathatatlan előnye volt ugyanakkor a webadmin felületnek, hogy nem Windows alapú eszközről is elérhetővé tehettük – ennek jelentősége, ha akkoriban nem is volt számottevő, napjainkban abban áll, hogy akár egy egyszerű tablet, vagy nagyobb kijelzőjű okostelefon képernyőjéről is

elvégezhetőek a fontosabb, sürgősebb adminisztrációs feladatok, mint amilyen például egy tartalék vendéggép felébresztése, vagy korábbi pillanatképre való visszaállás.

Házi feladatom céljául tehát egy egyszerű, a legfontosabb feladatok elvégzésére alkalmas webadmin felület elkészítését tűztem ki, ami az alábbi funkciók ellátására alkalmas:

- Virtuális gépek listázása, állapotuk áttekintése.
- Egyszerű állapotváltások indítása, mint amilyen például a felfüggesztés, bekapcsolás stb.
- Pillanatkép készítése és visszaállítása.

További extrafunkcionális követelmény a modern webes szabványok követése és a kompakt oldalelrendezés kialakítása, hogy a felület tabletekről is kényelmesen használható legyen.

Az elkészült alkalmazás

Belső felépítés, implementációs részletek

Bár, mint említettem, WMI hívások indítása lényegesen kézreállóbb .NET környezetben, mint COM hívásoké, az elérendő objektummodell továbbra sem tekinthető fordításidőben kötöttnek – legalábbis a keretrendszer által beépített formában nem, hiszen az általános WMI kezelésre lett felkészítve, alkalmazásspecifikus kiterjesztéseket értelemszerűen nem ismer. Így a statikus tipizálásra építő fejlesztőkörnyezet-szolgáltatásokról – mint amilyen az objektumtagok nevének automatikus kiegészítése, fordításidejű ellenőrzés stb. – jobbára le kell mondanunk, illetve sok ismétlődő kód is keletkezhet az alkalmazásban, ha minden egyes funkció maga állítja össze a WMI kéréseket. Első lépésként tehát egy, a WMI API-ra építő, típusbiztos, .NET-es interfészt szerettem volna keresni, illetve készíteni.

A .NET alapú, nyílt forráskódú projekteket gyűjtő oldalon, a Codeplexen találkozhatunk is egy ilyen osztálykönyvtárral, ennek fejlesztése azonban, sajnálatos módon, elkészülte előtt leállt [2]. Így, bár az indulásban hatalmas segítséget jelentett – ugyanis a CIM Hyper-V környezetben használatos struktúráinak C# osztályokra és enumokra történő leképezése, illetve néhány egyszerűbb lekérdezés megvalósítása megtalálható a jelenlegi változatban is – néhány funkciót magam kellett elkészítsek, vagy kijavítsak, hogy az osztálykönyvtár a kitűzött célok elérésére alkalmas legyen:[1]

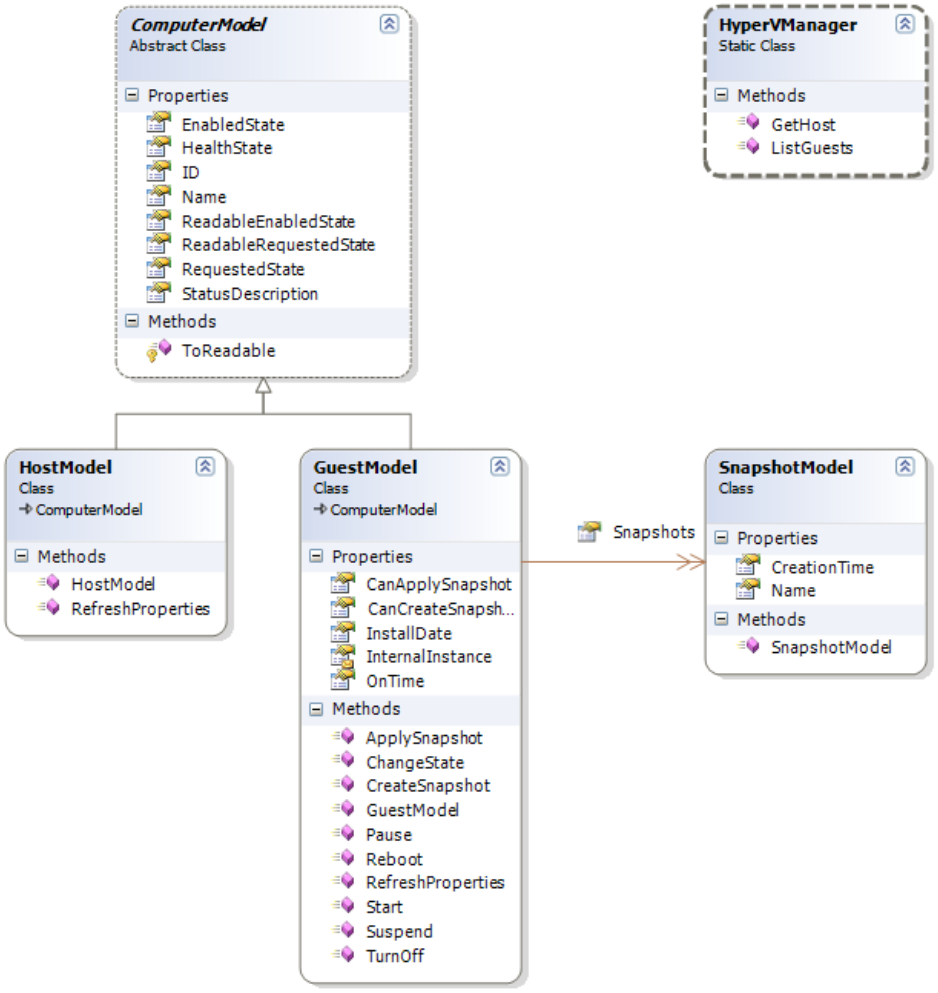
[1] A kiegészítéseket a 2010. októberében megjelent 0.0.2.0 változathoz készítettem. Bár azóta töltöttek fel patcheket, új kiadást nem rendeltek hozzájuk, és a változási naplók alapján a módosítások inkább

- A *CIM_EnabledLogicalElement* osztály *EnabledState* mezője által felvehető értékeket gyűjtő *EnabledStateOption* enum kizárólag az általános értékeket tartalmazta, a Hyper-V specifikus állapotokat nem – ezekre az állapotok értelmezéséhez, illetve az állapotváltáshoz szükség volt, így a dokumentáció alapján [3] kiegészítettem az enumerációt.
- Az állapotváltást lehetővé tevő *RequestStateChange* művelet nem került implementálásra, így azt pótoltam. A saját implementációm nem általánosan, csak kifejezetten a Hyper-V rendszerben történő alkalmazást célozva készült el, így paraméterként kizárólag a Hyper-V által engedélyezett célállapotokat [4] fogadja, (ehhez felvettem egy új enum típust, a *StateChangeOption*-t), illetve a visszatérési értéket nem kezeli (lévén az Hyper-V esetén szükségtelen, az eredmény ugyanis rögtön a hívás után kiolvasható az érintett objektum *RequestedState*, ill. *EnabledState* mezőiből).
- A *Guest* osztály *Snapshots* tulajdonsága – amely az elérhető pillanatképeket listázza –, bár implementálásra került, a lényegi részét jelentő WMI lekérdezés nem járt eredménnyel – így a lekérdezést javítottam.
- A pillanatképre való visszaállás, bár implementálásra került, az általa hívott, pillanatképet lekérdező *GetSnapshotByUniqueName* függvény implementációja hibás volt, így azt teljesen újraírtam.

Az így kiegészített, illetve javított osztálykönyvtár formájában már rendelkezésemre állt egy, C# környezetben kényelmesen hívható, típusbiztos interfész a Hyper-V fontosabb funkcióihoz. (A mellékelt forráskód *System.Management.HyperV* projectje.)

Erre építve készítettem el az előző szakaszban specifikált webadmin felület alkalmazáslogikáját, megjelenítéstechnológia-független formában, a *HyperVAdminBLL* projektben – ez egy általános .NET osztálykönyvtár, ami statikus függvények, és egyszerű belső felépítésű, felületi adatkötést származtatott tulajdonságok definiálásával támogató Model osztályok formájában teszi elérhetővé a kitűzött funkcionalitást (2. ábra).

kísérletezgetéseknek, mint valós újításoknak mutatkoznak – például a legutóbbi, idén júliusi frissítés arra irányult, hogy a *ManagementObject* tulajdonságait a C# dynamic kulcsszóval érjék el, indexerek helyett – ennek jelentősége, az osztálykönyvtár használhatósága szempontjából, csekély, stabilitási problémákat viszont okozhat, mivel bevallottan nem tesztelték behatóan.



2. ábra: A *HyperVAdminBLL* osztálydiagramja

Végül implementáltam a konkrét web réteget is, ASP.NET MVC 4 platformon – ez a *HyperVAdminWeb*.[2,3] Az ASP.NET MVC a Microsoft újgenerációs vékonykliens platformja, ami a Model-View-Controller mintát követve szolgálja ki a kéréseket, és állítja elő a HTML oldalakat. Az elkészült webalkalmazás kódja rendkívül egyszerű, köszönhetően annak, hogy a *System.Management.HyperV* könyvtárral közvetlenül nem dolgozik (sem függvényhívások, sem osztályelérések formájában), csak a *HyperVAdminBLL* metódusait hívja, illetve annak objektumait jeleníti meg.

[2] A házi feladat beadására szolgáló portál szabta meglehetősen szűk méretkorlát miatt a *HyperVAdminWeb* függőségeit kielégítő *packages* könyvtár tartalmát ürítenem kellett beadás előtt. Ezért a fordítás előfeltétele, hogy a *HyperVAdminWeb* könyvtárban található *packages.config* definiálta függőségek letöltésre kerüljenek a NuGet parancssori eszköz segítségével.

[3] A házi feladat írásakor az ASP.NET MVC 4 csak preview változatban volt elérhető, így előfordulhat, hogy a végleges változattal történő fordítás és futtatás kisebb módosításokat igényel majd, amennyiben változnak az interfészek a keretrendszerben.

Az egyszerűségen túl további előnye e köztes rétegnek, hogy amennyiben a *MOCK_HYPER_V* szimbólumot definiálva fordítjuk le, a hívásokat el sem juttatja a *System.Management.HyperV* felé, csak – többé-kevésbé – emulálja annak működését, így a webes felület olyan számítógépen is fejleszthető és kipróbálható, amelyen Hyper-V nem érhető el.

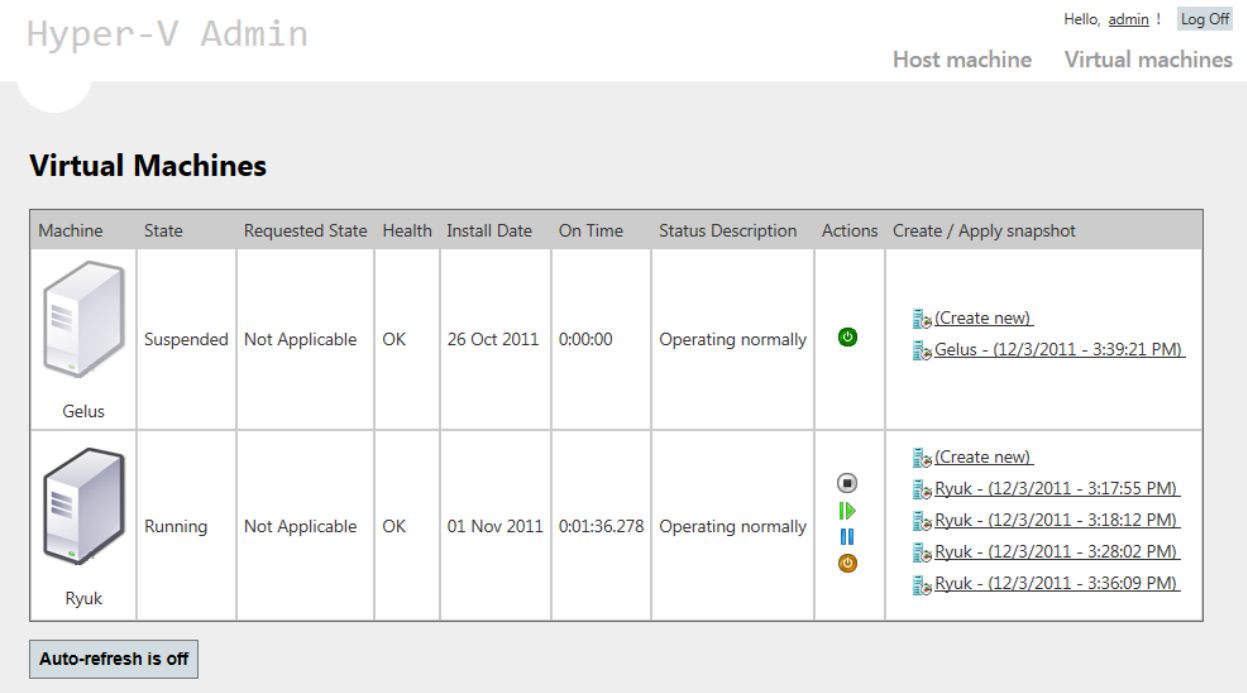
Telepítési információk

A felület a már említett IIS webszerveren hosztolandó [5]. Fontos, hogy a weboldal létrehozásakor olyan Application Poolt adjunk meg, amely a .NET Framework 4.0 változatát használja, illetve olyan felhasználó nevében fut, akinek van joga a Hyper-V rendszerben elvégzendő műveletekhez. Erősen ajánlott továbbá a HTTP alapú elérés letiltása, és a HTTPS elérés megkövetelése, biztonsági okokból. A webalkalmazás – a felhasználó-kezeléshez – egy egyszerű adatbázist is tartalmaz, ennek karbantartása azonban SQL Compact Edition 4 segítségével történik, így adatbázisszerverre nincs szükség.

A rendszer használata előtt a webalkalmazás gyökerében található *Web.config* állomány *HyperVAdminWeb.Properties.Settings* szekciójának *EnableRegistration* beállítását állítsuk *True*-ra, hogy új felhasználókat hozhassunk létre. Ebben az állásban a *https://{a webalkalmazás gyökér URL-e}/Account/Register* címet meglátogatva bárki regisztrálhat a rendszerbe, ilyenkor tehát a weboldalt semmiképp se tegyük kívülről elérhetővé. *False* állásban a regisztráció minden esetben meghiúsul (még akkor is, ha valaki ismeri a regisztrációs oldal imént említett címét – a menüben ugyanis egyáltalán nem szerepel), így ebben az esetben már lehetővé tehetjük a külső elérést, regisztrált felhasználóink számára.

Az alkalmazás felülete

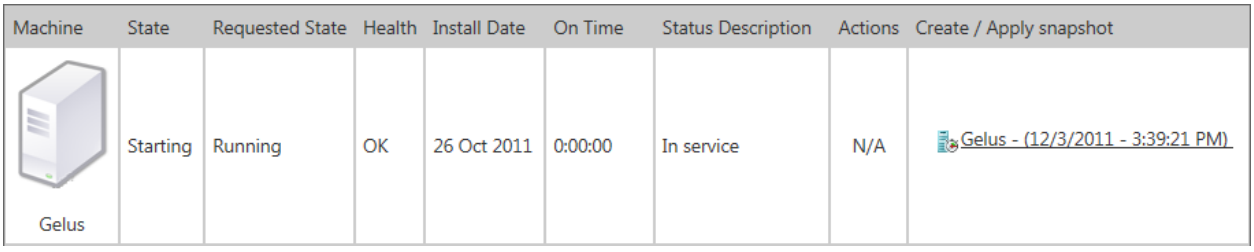
Az alkalmazás tesztelése Windows Server 2008 R2 rendszeren történt, amelyre két virtuális gépet telepítettem. Bejelentkezés után néhány alapinformációt láthatunk a gazda gépről, majd a *Virtual Machines* menüpontot választva megjelenik a virtuális gépek listája, a 3. ábrán látható formában.



3. ábra: A virtuális gépeket kezelő felület

Az első oszlop a virtuális gép nevét tartalmazza, illetve egy, a gép állapotát reprezentáló ikont, ahol halványabban a megállított, erősebb színnel a futó gépeket jelöli a felület. A további szöveges oszlopok az állapot részletesebb kifejtését tartalmazzák.

Az Actions oszlop az állapotátmenetet kiváltó műveleteket kínálja fel – innen tudjuk a gépeket bekapcsolni, felfüggeszteni stb. A parancs kiadását követően, ha az átmenet nem hajtódik végre azonnal (például a bekapcsolás jellemzően időigényesebb folyamat), a Requested State oszlopban jelenik meg a célállapot. Ilyenkor a műveletek – az érintett gépnél – ideiglenesen letiltódnak (4. ábra).



4. ábra: Virtuális gép, közvetlenül bekapcsolás után, annak befejeződése előtt

Webes felület lévén az alkalmazás a célállapot eléréséről nem tud közvetlenül értesülni, csak időszakos lekérdezések (polling) útján, amelyet a táblázat alatt látható *Auto-refresh* gomb megnyomásával kapcsolhatunk be – természetesen az is elegendő, ha magunk frissítjük időnként az oldalt.

Az utolsó oszlop a virtuális gép pillanatképeit listázza, illetve felkínálja új pillanatkép létrehozásának lehetőségét is. Egy adott pillanatképre kattintva aktiválhatjuk azt, vagyis visszaállíthatjuk a virtuális gépet abba az állapotba, amelyben a pillanatkép készítésekor volt. A pillanatképekkel kapcsolatos műveletek aszinkron módon lettek megvalósítva, vagyis ilyenkor a teljes felület nem frissül (sőt, az automatikus frissülés is kikapcsolódik, ha engedélyezve volt), csak egy AJAX hívás indul el a szerver felé. Ennek oka, hogy az állapotváltással ellentétben, a pillanatkép-műveletek blokkolnak, nem térnek vissza azonnal a hívóhoz, így azokat hagyományos weboldal-lekérések formájában nem vehettem igénybe, mivel túl hosszú időre hagyták volna a böngészőt válasz nélkül. A jelenlegi megoldás mind a kérés kiadása, mind a válasz megérkezte után visszajelzést ad a felületen, JavaScript segítségével (5. ábra).

5. ábra: Bár a pillanatkép-visszaállítás hosszan tartó művelet, a parancs kiadásának hatása a felületen azonnal megjelenik

Összefoglalás

A Hyper-V programozási interfésze, köszönhetően annak, hogy a .NET környezetbe nehézkesen illeszkedő COM helyett WMI alapokon nyugszik, lényegesen könnyebben használható a modern alkalmazásfejlesztői platformokon, mint elődjéé, a Virtual Server 2005-é. Nem rendelkezik azonban beépített webes felülettel, ami bár általános esetben előrelépés abból a szempontból, hogy nem teszi a telepítés előfeltételévé webszerver meglétét és konfigurálását, a távoli elérés jóval rugalmasabb, ha tetszőleges böngészőből elvégezhető.

Egy ilyen felület elkészítése történhet közvetlenül a WMI interfészek hívásával is, ekkor azonban, az erős típusosság hiányában, a fejlesztőkörnyezet számos szolgáltatásáról le kell mondanunk, valamint a felület és az alkalmazáslogika túlzott összefonódásának veszélye is fennáll. Mind a karbantarthatóság, mind a kényelmes fejlesztés szempontjából praktikus tehát egy általános, erősen típusos, WMI-ra épülő interfész kialakítása, amelyen keresztül az alkalmazásunk a Hyper-V menedzsmentjét végzi, és bár a nyílt forráskódú közösségnek voltak is ilyen irányú kezdeményezései, még a legígéretesebb is messze áll a

befejezettől. Jó kiindulási alapot adott azonban egy saját megoldás kidolgozásához, amelyre építve az alapfelszereltségből hiányzó webadmin felületet elkészíthettem.

A továbbfejlesztés lehetséges irányait legfőképp az általános .NET interfész jövője határozza meg – hiszen a webes felület csak annak szolgáltatáskészletét tudja felkínálni, legalábbis közvetlen WMI hívások indítása nélkül. Terveim között szerepel módosításaim – vagy akár a teljes elkészült rendszer – közzététele a nyílt forráskódú .NET projekteket gyűjtő Codeplex portálon, ami remélhetőleg nem csak a hasonló alkalmazások elkészítését segíti majd elő, de talán az általános interfész fejlesztésének is új lendületet adhat.

Hivatkozások

[1] Virtual System Management Service:
http://msdn.microsoft.com/en-us/library/cc136990(v=vs.85).aspx

[2] HyperV C# Library:
http://hypervlib.codeplex.com/

[3] CIM_EnabledLogicalElement class:
http://msdn.microsoft.com/en-us/library/cc136818(VS.85).aspx

[4] RequestStateChange Method of the Msvm_ComputerSystem Class:
http://msdn.microsoft.com/en-us/library/cc723874(v=vs.85).aspx

[5] Create a Web Site (IIS 7):
http://technet.microsoft.com/en-us/library/cc772350(WS.10).aspx