SKRIPTOVACÍ TECHNIKY MICROSOFTU

Martin Molhanec

ČVUT-FEL, K-313, Technická 2, 166 27 PRAHA 6, Dejvice, ČR

Abstrakt

Cílem příspěvku je seznámit účastníky konference s vybranými skriptovacími technologiemi firmy Microsoft. Jedná se o následující skriptovací technologie: VBScript, JScript, Windows Scripting Host (WSH), HTML Application (HTA). Příspěvek poskytuje přehled o všech hlavních možnostech těchto technologií, které je možné uplatnit v různých programovacích prostředích a která umožňují objektově orientovanou výstavbu komponent. Příspěvek je určen spíše praktickým uživatelům, správcům a programátorům v prostředí Windows firmy Microsoft, kteří se chtějí seznámit s těmito technologiemi.

Klíčová slova: Informační systémy, Microsoft, skriptovací jazyky, VBScript, JScript, Windows Script Host (WSH), ADSI, WMI, HTA

Úvod

Autor tohoto příspěvku si klade za úkol seznámit účastníky semináře *Tvorba software* se skriptovacími technologiemi firmy *Microsoft*. Tento příspěvek částečně navazuje na podobný příspěvek autora přednesený v roce 1999 na konferenci *Objekty'99*. Nicméně od minulého roku došlo ve vývoji skriptovacích technologií firmy Microsoft k dalšímu rozvoji. Protože nikoliv všichni účastníci tohoto semináře se zúčastnili i semináře již výše zmíněného, zahrnu v tomto příspěvku i aktualizované informace, které byly obsahem předešlého příspěvku.

Skriptovací technologie firmy Microsoft jsou pevně sepjaty s jazykem *Basic*, koneckonců tento jazyk stál u počátku vzniku firmy Microsoft. Pominu nyní všechny předešlé implementací jazyka *Basic* firmou *Microsoft* a zastavím se až u jazyka *Visual Basic (VB)*. Přestože programovací jazyk *Visual Basic* je objektově orientovaný již od své první verze, teprve jeho poslední verze mohou náročného objektově orientovaného programátora uspokojit. Odlehčenou verzí tohoto programovací jazyka je *Visual Basic for Applications (VBA)*, který je určen být interním programovacím jazykem aplikací firmy Microsoft. Ještě více odlehčenou verzí je pak *Visual Basic Scripting Edition (VBScript)*, který byl nejprve určen coby programovací jazyk *Microsoft Internet Exploreru (MSIE)* a *Internet Information Serveru (IIS)*. Později firma Microsoft objevila, že se tento programovací jazyk výborně hodí jako jazyk pro psaní jednoduchých příkazových souborů a tím se konečně mnozí správci a uživatelé operačního systému Windows dočkali lepšího nástroje nežli je stařičký dávkový jazyk (soubory typu .BAT) příkazového procesoru těchto systémů. Programové prostředí, které toto umožňuje se nazývá *Windows Scripting Host (WSH)* a je součástí Windows 98 a vyšších verzí operačních systémů typu Windows. Dá se ovšem nainstalovat i na starší verze Windows i Windows NT.

Jazyk VBScript je veskrze velice jednoduchý. Jeho síla spočívá v možnosti spolupracovat s *ActiveX* objekty. Bez této možnosti by byl prakticky nepoužitelný. V objektech ActiveX je skryta síla objektové technologie firmy Microsoft. Jedná se o

technologii firmou Microsoft nazývanou *COM (Component Object Model)*. Tato technologie byla rozšířena pod názvem *DCOM (Distributed COM)* pro distribuované objekty. Poslední verze této technologie se nazývá *COM+*. Vzhledem ke skutečnosti, že všechny aplikace Microsoft Office i MSIE jsou současně *OLE Automation* objekty, je možné je ovládat nejen z VB či VBA, ale dokonce i z VBScriptu.

Velice zajímavá skutečnost je ta, že samotná skriptovací engine je ve formě ActiveX objektu nebo snad lépe ActiveX komponenty. Dokonce i samotný interpreter jazyka VBScript je ve formě ActiveX komponenty. Díky této skutečnosti není VBScript jediným jazykem, který je možné se skriptovací engine použít. Dalšími jazyky jsou *JScript* (jazyk firmy Microsoft kompatibilní s jazykem *JavaScript* firmy *Netscape*), který byl ovšem poněkud firmou Microsoft zanedbáván a jazyk *Perl* (firmy *ActiveWare*). Další jazyky, například *Tcl* nebo *Python* mohou v brzké době přibýt.

V tomto příspěvku se budu zabývat některými skriptovými a současně objektovými technologiemi Microsoftu, které nejsou zatím ještě příliš známé mezi odbornou veřejností a doufám, že můj příspěvek přispěje k jejich širší znalosti.

1. Novinky v implementaci scriptovací engine a jazyků VBScript a JScript

Jazyky VBScript a JScript, které tvoří základ skriptovací technologie firmy Microsoft se neustále vyvíjejí. Jako pozitivní skutečnost považuji fakt, že týmy, které mají vývoj těchto jazyků na starosti, poměrně pružně reagují na požadavky uživatelů a oba dva jazyky a skriptovací engine neustále vylepšují.

1.1 VBScript a JScript verze 5.0 a 5.1

Vývoj skriptovacích jazyků VBScript a JScript probíhal v poslední době poměrně rychle. V současné době je poslední produkční verzí verze 5.1. Verze 5.1 se od verze 5.0 liší zejména přítomností WSH 2.0 a WSC 1.1 a navíc je standardní verzí, která se instaluje s operačním systémem Windows 2000. Vlastní skriptovací jazyky se podstatně v těchto dvou verzích neliší.

1.1.1 VBScript verze 5.1

Tato verze přinesla některé novinky, které byly programátory kladně přijaty. Jednou z nejvýznamnějších novinek je možnost definovat v jazyce VBScript nové uživatelské třídy pomocí příkazu CLASS. Ukažme si takovou jednoduchou definici třídy Osoba, která má atributy Jmeno a Prijmeni, read property Plne_Jmeno a metodu Hello.

```
Class Osoba
        Public Jmeno, Prijmeni
        Property Get Plne_Jmeno
                Plne_Jmeno = Jmeno & " " & Prijmeni
        End Property

        Sub Hello
                Wscript.Echo "Hello " & Plne_Jmeno
        End Sub
End Class
```

Tuto třídu je možné použít například takto. Nejprve vytvoříme nový objekt naší třídy Osoba, naplníme jeho atributy Jmeno a Prijmení. Nakonec přečteme read property Plne_Jmeno a zavoláme metodu Hello. Jak je patrné, není vytváření a používaní uživatelem definovaných tříd v jazyce VBScript nic složitého.

```
Dim moje_osoba
Set moje_osoba = new Osoba
moje_osoba.Jmeno = "Martin"
moje_osoba.Prijmeni = "Molhanec"
Wscript.Echo moje_osoba.Plne_Jmeno
moje_osoba.Hello
```

Mezi další užitečná vylepšení jazyka VBScript patří zavedení příkazu WITH, který nám podobně jako v jazyce VB nebo VBA ulehčí spoustu zbytečného psaní. Ukažme si, jak ho využijeme v naší ukázce práce s uživatelskými třídami a objekty.

```
Dim moje_osoba
Set moje_osoba = new Osoba
With moje_osoba
        .Jmeno = "Martin"
        .Prijmeni = "Molhanec"
        Wscript.Echo .Plne_Jmeno
        .Hello
End With
```

Další důležitou novinkou v poslední verzi jazyka VBScript je objekt pro užívání regulárních výrazů (regular expressions). Zajímavé na implementaci regulárních výrazů v jazyce VBScript je právě skutečnost, že jsou implementovány čistě objektově. A opět bude nejlepší si uvést jednu malou ukázku. Náš prográmek bude v zadaném řetězci hledat výskyt slova řetězce Objekty a podle toho, zdali ho nalezne nebo ne, tuto skutečnost vypíše. Další výhoda této koncepce je ta, že samotnou třídu RegExp, která je realizovaná jako samostatná ActiveX komponenta, je možné využít i z jiných programovacích jazyků, které jsou ActiveX kompatibilní.

```
dim Muj_RegExp
Set Muj_RegExp = new RegExp
Muj_RegExp.pattern = ".Objekty."
if Muj_RegExp.Test("konference Objekty'99")then
        Wscript.Echo ("Retezec Objekty nalezen !")
else
        Wscript.Echo ("Retezec Objekty nenalezen !")
end if
```

Dalším velice podstatným vylepšením, které bylo žádáno mnoha programátory bylo přidání funkcí Eval a Execute, které umožňují za běhu programu vyhodnocovat výrazy nebo příkazy jazyka VBScript. Funkce EVAL je určená pro vyhodnocování výrazů a funkce EXECUTE pro vyhodnocování celých příkazů v lokálním jmenném prostoru. Pokud chceme příkaz vyhodnocovat v globálním jmenném prostoru, musíme použít funkci EXECUTEGLOBAL. Opět si uvedeme malou ukázku použití všech těchto funkcí.

```
' ukazka funkce EVAL
Wscript.Echo "evaluating: 1+1=" & Eval("1+1")
' ukazka funkce EXECUTE
s = "executing ""Wscript.Echo"" statement !"
Execute("Wscript.Echo s")
' ukazka funkce EXECUTEGLOBAL
ClassDemoDef = "Class Demo"       &       vbCrLf & _
                    "  Sub Echo"&       vbCrLf & _
                    "       Wscript.Echo ""Oh.. miracle !!!"" " & vbCrLf & _
                    "  End Sub"  &       vbCrLf & _
                    "End Class"
ExecuteGlobal ClassDemoDef
Set moje_demo = new Demo
moje_demo.Echo
```

Mezi další vylepšení jazyka VBScript verze 5.0 pak dále patří zavedení ukazatelů na funkce (function pointers), které umožňují za běhu programu dynamicky přidělovat obslužné funkce k událostem vyskytujícím se v systému. Mezi zbývající vylepšení pak patří podpora DCOM.

1.1.2 Jscript Version 5.1

Vývoj jazyka JScript je oproti vývoji jazyka VBScript možná trochu opožděn. Důvodem může být skutečnost, že firma Microsoft se snaží jazyk JScript rozvíjet v souladu s ECMAScripts standardem. Na druhé straně byl jazyk JScript od počátku do určité míry mocnějším nástrojem nežli jazyk VBScript, takže je možné také konstatovat, že oba jazyka jsou nyní co se týče svých možností zhruba rovnocenné. Jazyk JScript byl proto ve verzi 5.1 mimo podpory DCOM vylepšen především o příkaz TRY-CATCH-THROW určený pro ošetření výjimek (Exception Handling). Tento příkaz je použit tak, jak je možné očekávat u jazyka, který je podobný jazyku C. A opět si uvedeme jednu velice jednoduchou ukázku.

```
function TryCatchDemo(x)
{
  try {
        throw x;
  }
  catch(e) {
    return("Chyba: " + e);
  }
}
WScript.Echo(TryCatchDemo(0));
WScript.Echo(TryCatchDemo(1));
```

Oba výše uvedené jazyky byly dále vylepšeny o zvýšení rychlosti provádění kódu, možnost lepšího ladění aplikací a možnost zakódování zdrojového textu.

1.2 VBScript a JScript verze 5.5

Verze jazyků VBScript a JScript jsou ve verzi 5.5 v současné době ve formě beta verzí, které jsou uvolněny pro veřejné testování. Cílem verze 5.5 je docílit zejména shodnost jazyka JScript se standardem ECMA-262. Přestože je beta verze volně ke stažení není bohužel k dispozici ucelený dokument popisující všechna její vylepšení oproti stávající produkční verzi 5.1. Nicméně o několika vylepšeních jsou informace k dispozici.

- *Regulární výrazy*
  Regulární výrazy byly vylepšeny o *Replacement Functions*. Jedná se o možnost, aby pro každou nalezenou shodu, byla zavolána funkce, která provede transformaci nalezeného řetězce, jejíž výsledek bude vložen jako náhrada řetězce původního. Ukážeme si jednoduchý příklad. V tomto příkladu bude výskyt slova *Martin* nahrazen slovy *Martin Molhanec*. Navíc naše funkce *transform*, která bude provádět transformaci nalezeného řetězce, vypíše i pozici, kde došlo k nalezení hledaného řetězce.

```
// Ukazka vylepsene Replace Function ve verzi 5.5
// ********************************************
var re = /Martin/g;      // regularniho vyraz, ktery hledame
// retezec, ktery bude nahrazovan
var s  = "Tento clanek napsal Martin a Martin ho bude i prednaset"
// provedu transformaci
s = s.replace(re,transform);
// a zobrazim vysledek
WScript.echo(s)
//definice transformacni funkce
function transform(matchedstring,submatch,matchpos,source)
{

      WScript.echo("Match at " + submatch);
      return matchedstring + " Molhanec";
}
```

Další vylepšení jazyka JScript verze 5.5 se týkají rozšíření formátovacích funkcí a vylepšení v objektu Error. Vylepšení jazyka VBScript verze 5.5 se týkají vylepšení objektu RegExps, tak aby měl stejné funkce jako v jazyce JScript.

2. Windows Script Host (WSH)

Windows Scripting Host je prostředí pro běh skriptů v jazycích kompatibilních se skriptovací engine firmy Microsoft, tedy především v jazycích VBScript nebo JScript. WSH přinesl konečně možnost správci či uživatelovi operačních systémů firmy Microsoft psát jednoduché příkazové soubory v jazycích vyšší úrovně. Vzhledem ke koncepci skriptovací engine, pro kterou je samotný interpreter jazyka VBScript zásuvným modulem (ActiveX komponentou), je možné ve WSH využívat i další kompatibilní interpretační komponenty od třetích firem. Například interpreter jazyka Perl, Python nebo Tcl.

Pokud chceme nějaký programový kód určený pro WSH spustit, je nutné použít následující jednoduché programy.

Wscript.exe - který spouští náš kód v grafickém prostředí Windows
Cscript.exe - který spouští náš kód v prostředí textové konsole Windows

Další možností, jak program určený pro WSH spustit je prostě naň dvojitě kliknout ve standardním programu *Windows Explorer* nebo použít v DOSovém okénku příkaz *Start*.

Vlastní WSH tvoří pouze tenkou vrstvu, která jednak poskytne náš programový kód skriptovací engine a jednak vytváří jednoduché programové objektově orientované prostředí, které umožňuje našemu kódu přistupovat k následujícím zdrojům.

- Pomocí užití funkcí CreateObject a GetObject, přistup k libovolným ActiveX komponentám.
- Výpis zpráv na obrazovku a vstup příkazů od uživatele.
- Mapování síťových zdrojů.
- Připojení tiskáren.
- Přístup a modifikace proměnných prostředí.
- Čtení a modifikace údajů v registry.

Absolutně podstatná je první výše uvedená funkce! Umožňuje pomocí WSH ovládat i například tak složitou aplikaci jako je například Microsoft Excel. Koneckonců jednoduchá ukázka kompletního prográmku následuje.

```
' Excel Sample
Dim objXL
Set objXL = WScript.CreateObject("Excel.Application")
objXL.Visible = TRUE
objXL.WorkBooks.Add
objXL.Cells(1, 1).Value = "Text v prvni bunce"
objXL.Cells(2, 1).Value = "a ve druhe"
objXL.Cells(3, 1).Value = "a nakonec ve treti"
```

Člověk se neubrání obdivu. Tak jednoduchý prográmek a funguje! Vzhledem k tomu, že samotné WSH poskytuje pouze omezený počet zdrojů, které jsou po jeho nainstalování k dispozici, musí programátor WSH hledat další vhodné ActiveX komponenty, které jsou pro jeho záměry vhodné. Například ActiveX komponenta *SysInfo* zpřístupní některé informace o vybavení našeho počítače. Její použití je jednoduché. Následující prográmek vypíše informace o počtu a typu našeho procesoru, velikosti a volném místě na našem disku C: a typu a verzi našeho operačního systému.

```
Dim oSysInfo
Set oSysInfo = CreateObject("SystemInfoControl.MSysInfo")
Wscript.Echo oSysInfo.processorCount & " x " & oSysInfo.ProcessorType
oSysInfo.Drive = "c"
Wscript.Echo oSysInfo.TotalDiskSpace & " | " & oSysInfo.AvailableDiskSpace
Wscript.Echo oSysInfo.OSPlatform & " + " & oSysInfo.OSBuild
```

Poměrně nepříjemná je skutečnost, že WSH poskytuje minimální *GUI (Graphical User Interface)*, který bychom mohli použít při tvorbě našich aplikací. Ale i zde si šikovní programátoři pomohli a to využitím vlastností *Microsoft Internet Exploreru (MSIE)*. Klasický prográmek, který vytiskne na obrazovku uživatele slova *Hello, world !!!*, je uveden na následujícím příkladu. Nutno ovšem podotknout, že spolupráce WSH a MSIE se používá především pro vstup dat, kde se s výhodou využívají možnosti formulářů v HTML stránkách.

```
Dim IE
On Error Resume Next
Set IE = CreateObject("InternetExplorer.Application")
With ie
        .height=370
        .width=500
        .menubar=0
        .toolbar=0
        .navigate "About:Blank"
        .visible=1
        Do while .Busy
                ' wait for page to load
        Loop
        With ie.Document
                .Write "<html><body>"
                .Write "Hello, world !!!"
                .Write "</html></body>"
        End With
End With
```

2.1 WSH ve verzi 2.0

Poslední produkční verze WSH je verze 2.0, která byla připravována pro Windows 2000 a přinesla několik významných rozšíření této technologie. Jedná se o následující novinky.

- Podpora INCLUDE příkazu.
- Podpora užití několika skriptových jazyků v jednom programovém souboru.
- Podpora užití *Type Library.*
- XML kompatibilní formát zdrojového souboru.
- Podpora několika Job v jednom souboru.
- Podpora Drag & Drop v úrovni operačního systému.
- Podpora ladění (debugging).
- Podpora pozdržení (pause) při vykonávání skriptu.
- Podpora Stdin, Stdout a Stderr.
- Ovládání aplikací pomocí příkazu SendKeys.

Všechna výše uvedená vylepšení byla vytvořena na základě připomínek prvních uživatelů WSH a jejich podrobné vysvětlení by přesáhlo rozsah tohoto příspěvku. Přes všechny možnosti, které WSH poskytuje, je však zatím poměrně málo užíváno. Důvodem je

snad skutečnost, že není zatím běžnou součástí OS Windows, protože do starších verzích se musí explicitně doinstalovat. Nicméně lze očekávat, že s přechodem na Windows 2000 jehož je WSH verze 2.0 standardní součástí se tato situace podstatně změní.

Přestože je možnost použití WSH pro správce systémů Windows velikým přínosem, byla podrobena kritice skutečnost, že pro přístup k mnoha systémovým informacím operačního systému a síťového prostředí bylo nutné používat ActiveX komponenty od třetích stran. Firma Microsoft se proto rozhodla činnost správců podpořit zavedením dvou nových komponent.

2.2 Active Directory System Interface (ADSI)

Komponenta ADSI je určena jako abstraktní vrstva, která pro správce sítí umožňuje transparentní přístup k síťovým zdrojům. Nezávisle na typu síťového prostředí umožňuje správci takové činnosti jako vytváření nového uživatele, jeho přidání do skupiny nebo nastavení jeho práv k síťovým zdrojům. Přestože ADSI je standardní součástí až operačního systému Windows 2000, je možné ho také nainstalovat například pro Windows 98 nebo Windows NT. Přestože, jak je z jeho názvu patrné bylo vytvořeno zřejmě s primárním účelem podpory technologie *Active Directory*, která ve Windows 2000 nahrazuje starší technologii *Microsoft Domain*. Je možné ADSI používat pro přístup k síťovým zdrojům firmy Novell nebo podle standardu *LDAP*. Následující tabulka nám ukazuje, jak pomocí ADSI získat objekt představující *uživatele* z různých síťových zdrojů.

Síťový zdroj	**Ukázka použití příkazu**
Netware Bindery	Set oUser = GetObject(”NWCOMPAT://Server/User”)
Netware NDS	Set oUser = GetObject(”NDS://Tree/O=Company/OU=Dep/CN=User”)
Windows NT 4.0	Set oUser = GetObject(”WinNT://Domain/User”)
Active Directory nebo LDAP	Set oUser = GetObject(”LDAP://Server.Company.Com/C=Country/O=Company/OU=Dep/CN=User”)

Nyní si ukážeme, jak pomocí ADSI získat seznam uživatelů (users) skupiny users domain K313. Program není nikterak složitý, nejprve pomocí *GetObject* získáme objekt představující celou skupinu. V kolekci (*collection*) *Members* jsou pak objekty jednotlivých uživatelů (*user*). Pomocí cyklu tuto kolekci zpracujeme a tak získáme objekty jednotlivých uživatelů, z jejichž *property name* získáme logovací jméno uživatele.

```
' Ukazka vypisu uzivatelu dane skupinu v dane domain
' ==================================================
Domain          = "K313"   'nastavim domain
Group           = "Users"   ' nastavim skupinu
' nejprve si vyzvednu objekt skupiny
Set GroupObj = GetObject("WinNT://" & Domain & "/" & Group)
' pro kazdeho clena skupiny ziskam objekt uzivatele
' a vypisi jeho jmeno
For each UserObj in GroupObj.Members
   List = List & UserObj.Name & vbCrLf
Next
' vsechna jmena nyni zobrazim
Wscript.Echo List
```

Podobným způsobem je možné získat z ADSI i další jiné údaje. Je pochopitelně možné pomocí ADSI nejen údaje číst, ale i zapisovat nebo modifikovat. Například pomocí následujícího skriptu vytvoříme nového uživatele *Novak* v domain *K313*.

```
' Ukazka pridani noveho uzivatele do domain
' ==========================================
NewUser = "Novak"       ' jmeno noveho uzivatele
Domain  = "K313"        ' jmeno domain
' ziskame objekt domain
Set oDomain = GetObject("WinNT://" & Domain)
' vytvorime uzivatele
Set oUser = oDomain.Create("User",NewUser)
' nastavime mu heslo
oUser.AccountRestrictions.SetPassword("bflmpsvz")
' a informaci ulozime
oUser.SetInfo
```

Je vidět, že správce sítě si může v jazyce VBScript či JScript, nebo třeba i v jazyce Perl či některém dalším, velice snadno vytvářet vlastní skripty pro práci s uživatelskými konty. Například může snadno vytvořit skript, který na základě jmen studentů uložených v excelovském souboru, pro ně začátkem semestru vytvoří hromadným způsobem konta v domain a koncem semestru je opět smaže.

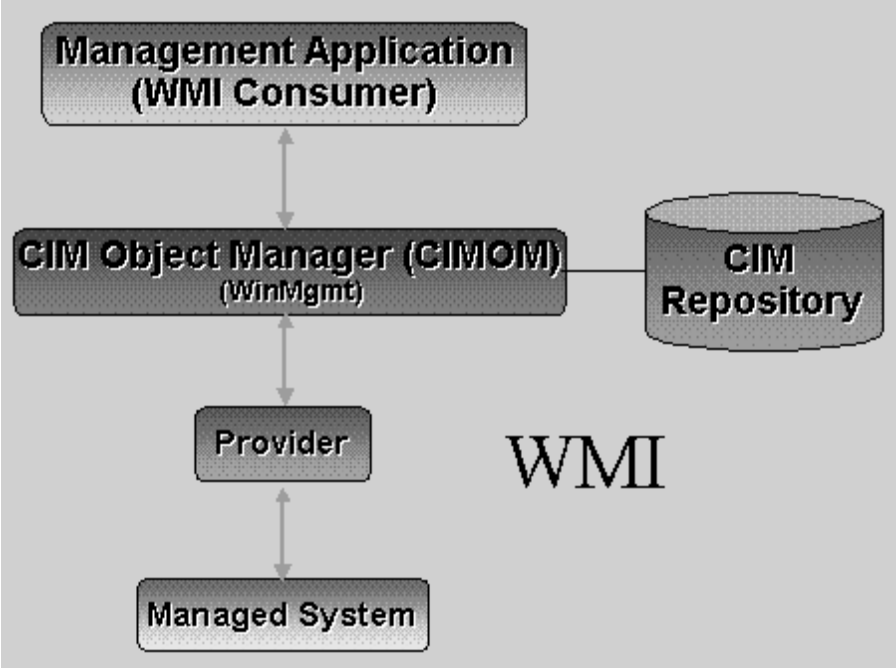
2.3 Windows Management Instrumentation (WMI)

Komponenta *WMI* je určena pro standardní přístup k systémovým informacím, jako je například typ video adapteru nebo množství paměti pracovní stanice. *WMI* je založeno na *Web-based Enterprise Mangement (WBEM)* iniciativě a *Common Information Model (CIM)* standardizační iniciativy *Desktop Management Task Force (DMTF)* a bylo firmou Microsoft implementováno pro operační systém Windows 2000, nicméně je dostupné i pro operační systémy Windows 95, Windows 98 a Windows NT 4.0. Mezi hlavní přednosti WMI patří:

- *Jednotné rozhraní pro skriptovací jazyky*
- *Podpora vzdálené administrace*
- *Navigační přístup k WMI objektům*

- *Schopnost dotazovacího přístupu k WMI objektům*
- *Možnost práce s událostmi*

Základní architektura WMI je znázorněna na následujícím obrázku.



Na obrázku je vidět ze kterých komponent se architektura WMI skládá. *Provider* je agent, který komunikuje na straně jedné s částí systému, která má být spravována (operační systém, služby, aplikace, device driver, disky, atp.) a na straně druhé s modulem *CIM object manager (CIMOM, WinMgmt)*. Informace dodané agentem *Provider* jsou mapovány *CIMOM* do příslušných tříd a objektů, které jsou dostupné skrze WMI rozhraní. *CIMOM* navíc pracuje s databází *CIM Repository,* ve které jsou uložena statická data o systému a dále poskytuje objekty, které obsahují několik kategorií informaci.

- *Properties*
- *Methods*
- *Events*
- *Associations*

Kategorie *Properties, Methods* a *Events* nepotřebují zvláštní výklad. Zajímavější je poslední kategorie. *Associations* jsou speciálním druhem tříd, které udávají vztahy mezi jednotlivými třídami WMI navzájem. Součástí standardní instalace WMI pro Windows 2000 jsou dále následující standardní *Providers*.

- *Win32 Provider*
  Poskytuje informace o operačním systému, počítači, periferních zařízeních, souborovém systému a bezpečnostním systému
- *WDM Provider*
  Poskytuje low-level informace o vstupních zařízeních, diskových zařízeních, síti a komunikačních portech.

- *Event Log Provider*
  Umožňuje přístup k logovacímu subsystému Windows NT.
- *Registry Provider*
  Umožňuje přístup k *Registry*.
- *Performance Counter Provider*
  Umožňuje přístup k čítačům zatížení. Pouze pro Windows 2000.
- *Active Directory Provider*
  Funguje jako brána ke službám ADSI.
- *Windows Installer Provider*
  Umožňuje přístup k instalátoru Windows. Poskytuje informace o všech instalovaných aplikacích.
- *SNMP Provider*
  Funguje jako brána ke službám SNMP.
- *View Provider*
  Umožňuje vytvářet nové třídy a pohledy ze tříd stávajících.

Ale nejlepší bude uvést opět několik jednoduchých ukázek práce s WMI rozhraním. První ukázka je užitečný skript, který zjistí jaké procesy právě na počítači běží.

```
' ukazka prace s WMI, zjisteni bezicich procesu
' ================================================
' nejprve vytvorime instanci CIMOM
set objService = getobject("winmgmts:")
' a nyni v cyklu pro kazdou instanci typu Win32_process
' zjistujeme jeji jmeno a proces id
for each Process in objService.InstancesOf("Win32_process")
    List = List & Process.Name & vbTab & Process.processid & vbCrLf
next
' a zde vypiseme seznam procesu
Wscript.Echo List
```

Druhá ukázka je také poměrně jednoduchá. Jedná se o zjištění volného místa na všech discích. Na rozdíl od prvé ukázky, kde byl uplatněn normální navigační přístup k pro nás zajímavých objektů, je u druhé ukázky použit přístup dotazovací, velice podobný užití standardního dotazovacího jazyka SQL.

```
' prace s WMI, zjisteni malo mista na disku
'
' nejprve vytvorime kolekci jednotlivych disku
' (musi být na jedné řádce!)
Set DiskSet = GetObject("winmgmts:").ExecQuery("select FreeSpace,Size,Name
from Win32_LogicalDisk where DriveType=3")
' nyni pro kazdy disk zjistujeme volne misto
' a pokud je mensi nezli 50% vypiseme informaci
For each Disk in DiskSet
  If (Disk.FreeSpace/Disk.Size) < 0.50 Then
     List = List & "Drive " + Disk.Name + " is low on space." & vbCrL
  End If
Next
```

```
' a zde je skutecny vypis
WScript.Echo List
```

Možnosti WMI jsou velice bohaté. Umožňují přístup ke každé části počítače a správce počítačů může psát velice snadno užitečné skripty. Nicméně je nutné dodat, že objektový model WMI, podobně jako ADSI, je velice bohatý, zatím nepříliš rozšířený mezi odbornou veřejností, a proto jeho zvládnutí bude vyžadovat určité nezanedbatelné úsilí.

3. HTML Application (HTA)

HTA je poměrně nová technologie, která není příliš známá. Jedná se o aplikace psané zejména ve skriptovacích jazycích VBScript nebo JScript. Na rozdíl však od WSH pracují aplikace v kontextu DHTML, které je využito jako kvalitní grafické uživatelské rozhraní. Výhodou této koncepce je skutečnost, že návrhář vzhledu aplikace má k dispozici všechny možnosti technologie DHTML včetně kaskádních stylů (CSS). Navíc pro návrh atraktivního vzhledu aplikace mohou být využity WYSIWYG editory HTML. Na rozdíl od skriptů, které pracují v internetovém prohlížeči, nejsou na aplikace HTA uplatňována žádná omezení. Je tedy možné, aby HTA aplikace měla úplný přístup k počítači uživatele. Na rozdíl od skriptů, které pracují přímo v internetovém prohlížeči, pracuje HTA aplikace v běžném okénku windows, proto bylo HTML obohaceno o speciální značku *HTA:APPLICATION*, která slouží pro upravení vzhledu okénka, ve kterém HTA pracuje.

Možná čtenáře tohoto příspěvku překvapí, že nejjednodušší HTA aplikace je následující:

```
Hello world...
```

Opravdu není to žádný žert. Stačí výše uvedený text napsat a soubor pojmenovat například: pokus1.hta a spustit. Nicméně výše uvedená aplikace nedodržuje dobrá pravidla psaní HTML kódu a navíc má pouze defaultní chování. Trochu více sofistikovaná aplikace bude mít například následující podobu.

```
<HTML>
<HEAD><TITLE>Moje prvni HTA aplikace</TITLE>
<HTA:APPLICATION  ID="oHTA"
            APPLICATIONNAME="pokus"
            BORDER="thick"
            BORDERSTYLE="complex"
            CAPTION="yes"
            ICON="winupd.ico"
            SINGLEINSTANCE="yes"
            SYSMENU="yes"
            VERSION="1.0"
            WINDOWSTATE="minimize"
>
</HEAD>
<BODY>
Hello, world...
</BODY>
</HTML>
```

Jak je patrné, naše vylepšená aplikace obsahuje celou řadu parametrů u značky *HTA:APPLICATION*. Není účelem tohoto příspěvku zde všechny tyto parametry vysvětlit.Většina z nich je ostatně pochopitelná dle svého názvu. Zmíním se pouze o některých z nich. Parametr *singleinstance* dovoluje aplikaci spustit pouze jedenkrát, při druhém spuštění se otevře okno již dříve spuštěné aplikace. Parametr *windowstate* umožňuje aplikaci spustit například minimalizovaně či maximalizovaně. Pomocí parametru *id* se můžeme z aplikace obracet na HTA aplikaci jako objekt, což nám umožňuje programově přečíst parametry HTA aplikace. Navíc existuje ještě jeden nesmírně důležitý parametr: *commandLine*, který nám zpřístupňuje parametry příkazové řádky, pomocí které byla HTA aplikace spuštěna. Ukážeme si ještě jednu o trochu složitější aplikaci.

```
<HTML><HEAD>
<TITLE>Sample HTA Shell</TITLE>
<HTA:APPLICATION ID="myapp">
<SCRIPT>
var params = myapp.commandLine;
function frwr(s){
document.frames("myframe").document.writeln(s)
}
function myfunc() {
  frwr("parametry pri spusteni:<br>");
  frwr(params);
}
</SCRIPT></HEAD>
<BODY>
<IFRAME ID="myframe"></IFRAME><BR>
<BUTTON onclick="myfunc()">button</BUTTON>
<SCRIPT>
frwr("stiskni tlacitko\<br\>");
</SCRIPT>
</BODY>
</HTML>
```

Pokusím výše uvedou aplikaci popsat. Uživatelské prostředí aplikace je tvoře rámcem (značka *IFRAME*) a tlačítkem (značka *BUTTON*). V rámci je text *„stiskni tlačitko“* (značka *SCRIPT* v části *BODY*). Tento text je do rámce zapsán pomocnou funkcí *frwr*, která je definována v části *HEAD* (značka *SCRIPT* v části *HEAD*). Po stisknutí tlačítka se vyvolá funkce *myfunc* (je definována ve stejném místě jako funkce *frwr*), která do rámce vepíše parametry příkazové řádky při spuštění této aplikace. Trochu komplikované? Asi ano, nicméně pokud tomuto stylu programování přivykneme, je možné poměrně rychle vytvářet aplikace s dobrým grafickým vzhledem.

Závěr

Přestože se mi v tomto příspěvku nepodařilo podchytit celý bohatý svět problematiky skriptovacích technologií firmy Microsoft, snad bude i takový příspěvek pro čtenáře vítaným informačním přínosem. Ve svém příspěvku jsem se nezmínil o problematice a významu následujících skriptovacích technologiích.

- *Windows Script Components*, které umožňují velice jednoduchým způsobem psát pomocí VBScriptu, Jscriptu a XML vlastní ActiveX komponenty.
- *Remote Scripting* technologii, která umožňuje volat z HTML stránky v prohlížeči klienta metody objektů v ASP stránkách na serveru.
- *Microsoft Script Control* technologii, která umožňuje, aby se vaše vlastní aplikace staly ovladatelné ze skriptovacích jazyků.

Některé z výše jmenovaných technologiích již byly zmíněny v mých předchozích příspěvcích na tomto semináři a je pravděpodobné, že k nim opět obrátím svoji pozornost i v budoucnu. Celá problematika je poměrně velice rozsáhlá a vyžaduje určité úsilí při sledování jejího vývoje.

Dle mého názoru si skriptovací technologie firmy Microsoft zaslouží zvýšenou pozornost. Na platformě Windows umožňují totiž velice efektivně zvýšit produktivitu práce zejména správcům počítačových sítí. Jsou také součástí řešení intranetových a Internetových aplikací. Na platformě UNIXu jsou skriptovací technologie jedním ze základních nástrojů pro správu systémů, na platformě MS DOS a Windows byl k dispozici pouze jazyk dávek .bat. Nyní se situace mění, je čas používat skriptování!

Odkazy

Skriptovací technologie u fy Microsoft..... msdn.microsoft.com/scripting
Články o skriptování u fy Microsft ..... msdn.microsft.com/voices/scripting.asp
Články o Jscriptu a JavaScriptu .....www.netpedia.com/features/javascript
HTA Overview.....msdn.microsoft.com/workshop/author/hta/overview/htaoverview.asp
ADSI FAQ..... cswahington.netreac.net/script_repository/faqs.asp?topic=adsifaq
WMI introduction..... msdn.microsoft.com/library/techart/mngwmi.htm
WMI Overview .....msdn.microsoft.com/library/background/html/wmiscript.htm
WSH .....cswashington.netreach.net
Molhanec .....http://martin.feld.cvut.cz/~mmm

Ing. Martin Molhanec, CSc

ČVUT-FEL, K-313
Technická 2
166 27 PRAHA 6, Dejvice
tel.: ++420 (2) 2435 2118
fax: ++420 (2) 2435 3949
email: molhanec@fel.cvut.cz

Molhasoft
Křivenická 406
181 00 PRAHA 8, Čimice
tel.: ++420 (2) 855 05 21
fidonet: 2:420/12.102
email:molhanec@technologist.com