

Seit der Verfügbarkeit des Option Packs für Windows-NT haben Administratoren die Möglichkeit, zwei leistungsfähige Skriptsprachen für Windows-NT 4.0 einzusetzen. Dabei handelt es sich um Jscript, das eine Untermenge der Sprachelemente von Java bereitstellt und Vbscript, das wiederum an Visual-Basic angelehnt ist. Beide Skriptsprachen laufen innerhalb einer auf Active-X basierenden Umgebung deren gemeinsame Basis der Windows-Scripting-Host (WSH) ist. Vbscript und Jscript sind dabei nicht die einzigen beiden Sprachen, die innerhalb der WSH-Umgebung laufen. Auch das von Unix bekannte Perl und andere Skriptsprachen sind bereits für den Windows-Scripting-Host verfügbar.

## DIRK PELZER

Der Windows-Scripting-Host stellt dem Administrator eine objektorientierte Programmierumgebung zur Verfügung, mit deren Hilfe er zahlreiche Funktionen, die von der Auswertung von Dateien über die Manipulation der Registry bis hin zum Verwalten von Benutzerkonten reichen. Je nach Anwendungsgebiet sind aber gegebenenfalls noch Zusatzkomponenten erforderlich. Für die Benutzerverwaltung wird beispielsweise noch das Active-Directory-Service-Interface benötigt, das von Microsoft erhältlich ist.

### Grundlagen der objektorientierten Programmierung

Der im Windows-Scripting-Host realisierte Ansatz basiert darauf, daß es zunächst Objekte gibt, auf die wiederum Methoden und Eigenschaften angewendet werden. Ein Standardobjekt von WSH vorhanden ist beispielsweise »Wscript« und eine Methode, die darauf angewendet werden kann, ist »Version«. Um die Methode auf das Objekt anzuwenden, wird eine Punktnotation verwendet. Dabei stellt das Objekt das erste Element dar. Anschließend folgt ein Punkt und danach die jeweilige Methode. Das Kommando `strVersion = Wscript.Version` weist beispielsweise der Variablen `strVersion` den Versions-String der Windows-Scripting-Host-Umgebung zu, die gerade ausgeführt wird. Informationen über Objekte und Methoden von WSH kann der Administrator entweder dem Microsoft Platform-SDK oder der Microsoft Web-Seite unter [www.microsoft.com/management/wshobj.htm](http://www.microsoft.com/management/wshobj.htm) entnehmen.

Mit Vbscript und Jscript geschriebene Skripte können entweder in einer Kommandozeilenumgebung oder in grafischer Form innerhalb von Fenstern

ausgeführt werden. Die Unterscheidung, in welcher Umgebung die Ausführung stattfindet, trifft der Administrator über das Kommando mit dem er das Skript aufruft. Mit dem Kommando `cscript skript.vbs` wird das angegebene Skript in einer Kommandozeilenumgebung ausgeführt. Wenn er statt dessen `wscript skript.vbs` eingibt, erfolgen Ausgaben, die das Skript macht beispielsweise innerhalb von Message-Boxen, die mit OK bestätigt werden müssen.

### NT-Administration am Beispiel

Die beste Methode, um sich in die Programmierung mit dem Windows-Scripting-Host einzuarbeiten, ist das Arbeiten mit Beispielen aus der Praxis. Denkbar wäre beispielsweise eine Aufgabenstellung, bei der für die Ermittlung der Speicherplatzauslastung eines Servers ein Skript geschrieben werden soll, das für ein angegebenes Verzeichnis ermittelt, wieviel Speicherplatz in Megabyte jeweils darunter belegt ist. Die Größen aller Unterverzeichnisse sollen dabei mit berücksichtigt werden. Das ganze soll mit dem Windows-Scripting-Host und Vbscript realisiert werden.

Um die Aufgabe lösen zu können, ist es sinnvoll, diese zunächst einmal in überschaubare Teile zu untergliedern. Dazu ist eine Aufteilung in ein Hauptprogramm und Funktionen, sowie Unterprogramme denkbar. Im dargestellten Fall soll sich das Hauptprogramm lediglich um die Aufnahme des übergebenen Kommandozeilenparameters in Form des Verzeichnisnamens kümmern, für das die Speicherplatzausnutzung ermittelt werden soll. Wenn kein Argument übergeben wird, so soll sich das Skript mit einer Fehlermeldung beenden. Die Ausgabe der Meldung ist dabei über eine Subroutine

realisiert. Das Ermitteln der Verzeichnisgröße soll in einer eigenen Funktion geschehen, die wiederum vom Hauptprogramm aus aufgerufen wird. Listing 1 zeigt, wie die Funktionalität realisiert werden könnte.

In der ersten Zeile des Skriptes wird über das Kommando »Option Explicit« festgelegt, daß alle Variablen, die innerhalb des Skriptes verwendet werden, explizit deklariert werden müssen. Anders als bei einer Programmiersprache wie C, bei der die Variablendeklaration vorgeschrieben ist, würde Vbscript bei der Ausführung eines Skriptes einfach eine Variable, die versehentlich falsch geschrieben wurde implizit als neue Variable deklarieren und das Skript weiter ausführen. Da dieses Verhalten zu nur schwer lokalisierbaren Fehlerbildern führen kann, empfiehlt sich ausdrücklich die explizite Variablendeklaration mit „Option explicit“. Im nächsten Schritt werden die beiden Variablen strDirectory und oArgs deklariert. StrDirectory wird im Hauptprogramm den Namen des Verzeichnisses aufnehmen, für das die Größe ermittelt werden soll und der beim Aufruf des Skriptes übergeben wird. Zur Übergabe von Kommandozeilenparametern kommt das WshArguments-Objekt zum Einsatz, welches allerdings nicht direkt zur Verfügung steht, sondern nur die Eigenschaft Wscript.Arguments. In der Praxis sieht das so aus, daß man mit Hilfe des Set-Kommandos der Variablen oArgs eine Referenz auf die Eigenschaft Wscript.Arguments übergibt. Über weitere Eigenschaften kann der Administrator dann mit Hilfe dieser Referenz die benötigten Parameter auslesen und weiter verarbeiten. Eine Eigenschaft im Zusammenhang mit Argumenten, die sehr nützlich ist, wird auch im Beispielskript verwendet. Es handelt sich dabei um die Eigenschaft Count. Diese liefert einen Integerwert zurück, der die Anzahl der übergebenen Kommandozeilenparameter enthält. Damit kann der Programmierer feststellen, ob und wenn ja, wie viele Parameter beim Aufruf übergeben wurden. Genau das prüft auch das Beispielskript ab und ruft für den Fall, daß kein Parameter übergeben wurde, ein Unterprogramm mit dem Namen Usage auf, welches eine Fehlermeldung ausgibt und die Syntax des Skriptes erklärt. Anschließend wird das Skript über Wscript.Quit verlassen und dabei in Klammern ein Fehlercode zurückgegeben, der beispielsweise einem anderen Skript oder einer Batchroutine als

Errorlevel zur Verfügung steht. Für den Fall, daß ein Argument wurde, wird der Variablen strDirectory mit Hilfe des Ausdrucks strDirectory = oArgs(0) das erste Argument übergeben. Genau wie bei Arrays unter Visual Basic oder Vbscript üblich, findet sich das erste Element an Position 0, das zweite an Position 1 und so weiter. Über die Variable oArgs ist der Programmierer demzufolge in der Lage auf die Array-Elemente des WshArguments-Objektes zuzugreifen. Die letzte Anweisung der Hauptroutine des Skriptes gibt als Ergebnis die Größe des angegebenen Verzeichnisses aus. Die Größe wird dabei mit Hilfe der Funktion ShowFolderSize ermittelt, wobei die Variable strDirectory als Parameter mit übergeben wird.

In der Funktion ShowFolderSize findet die eigentliche Ermittlung der Verzeichnisgrößen statt. Der Schlüssel für den Zugriff auf Dateien und Verzeichnisse ist der Einsatz des FileSystemObject (FSO) Objekt-Modells. Dieses gibt dem Skriptprogrammierer umfassende Möglichkeiten, um Dateien und Verzeichnisse anzulegen, zu verändern zu verschieben, zu löschen, oder einfach festzustellen, ob die angegebene Datei, beziehungsweise das Verzeichnis existiert. Um mit dem Objekt-Modell des FSO arbeiten zu können, sind prinzipiell drei Schritte notwendig. Zunächst muß der Programmierer mit Hilfe der CreateObject-Methode ein FileSystemObject erzeugen. Anschließend wendet er die geeignete Methode auf das Objekt an und kann dann auf die Eigenschaften des Objektes zugreifen. Genauso ist das auch im Beispiel realisiert. Mit Set FileSystemObject = CreateObject("Scripting.FileSystemObject") wird zunächst ein neues Objekt mit dem Namen FileSystemObject erzeugt. Im nächsten Schritt wird über Set Folder = FileSystemObject.GetFolder(strDirectory) die Methode GetFolder auf das Objekt angewendet und dabei der Name des gewünschten Verzeichnisses als Parameter übergeben. Dann wird der Variablen intFolderSize über die Eigenschaft Size die Größe des angegebenen Verzeichnisses übergeben. Um die Größe nicht zu unübersichtlich zu gestalten, erfolgt mittels FormatNumber noch die Umwandlung des Ergebnisses in eine Dezimalzahl mit einer Nachkommastelle. Danach wird die Zahl noch zusammen mit verschiedenen anderen Parametern in einen String

verpackt der dann wiederum an die Funktion ShowFolderSize übergeben das Ergebnis an die Hauptroutine zurück liefert.

#### Aufgabe sinnvoll erweitern

Nun ist die Anzeige der Größe eines Verzeichnisses natürlich nicht besonders spannend, aber sie veranschaulicht doch, wie relativ einfach der Umgang mit Dateien und Verzeichnissen unter Vbscript ist. Eine mögliche Erweiterung des Skripts wäre beispielsweise die Aufgabe, neben der Größe eines Verzeichnisses auch gleichzeitig die aller Unterverzeichnisse mit aufzulisten. Dazu sind gegenüber dem ursprünglichen Code nur einige kleinere Modifikationen notwendig, wie in Listing 2 ersichtlich ist. Dort wird mit Hilfe der booleschen Variable booSubFolders gesteuert, ob die Unterverzeichnisse des angegebenen Hauptverzeichnisses mit ausgegeben werden sollen oder nicht. Dazu wird im Skript lediglich geprüft, ob beim Aufruf mehr als ein Parameter übergeben wurde. Ist das der Fall, so wird booSubFolders auf TRUE gesetzt und in der Funktion ShowFolderSize eine zusätzliche For-Schleife ausgeführt, die ausgehend vom angegebenen Verzeichnis alle darunter liegenden ermittelt und auch für diese die aktuelle Größe angibt. Dabei handelt es sich nicht um eine gewöhnliche For-Schleife, sondern um eine, die für jedes Element einer Gruppe einen Reihe von Befehlen ausführt. Bei der Gruppe kann es sich dabei entweder um ein Array oder eine Collection handeln, welches wiederum eine spezielles Objekt darstellt. Im gezeigten Fall handelt es sich dabei um das Folders-Collection-Objekt, das beispielsweise eine Liste aller Unterverzeichnisse für ein angegebenes Directory ermittelt.

#### Zahlreiche weitere Funktionen verfügbar

Das FileSystemObject-Objekt-Modell kennt darüber hinaus noch zahlreiche weitere nützliche Funktionen. So kann der sich Administrator beispielsweise mit der Drives-Collection eine Liste der an das jeweilige System angeschlossenen physikalischen und logischen Laufwerke anzeigen lassen.

Auch das Löschen, Kopieren oder Erstellen von Dateien oder Verzeichnissen wird über Methoden wie zum Beispiel DeleteFile, CopyFile oder CreateTextFile unterstützt. Die generelle Vorgehensweise ist dabei, wie in den Beispielen gezeigt stets so, daß der Programmierer zunächst die CreateObject-Methode benutzt, um ein FileSystemObject-Objekt anzulegen. Anschließend wendet er die geeignete Methode auf das erzeugte Objekt an und kann dann auf die Objekteigenschaften zugreifen. Set fso = CreateObject("Scripting.FileSystemObject") legt beispielsweise das Objekt mit dem Namen fso an. Über Set fsoMethod = fso.GetFiles("C:\TEMP\MeineDatei.txt") wird daraufhin die Methode GetFiles auf das Objekt fso angewendet. Schließlich kann man mit Hilfe von intFileSize = fsoMethod.Size zum Beispiel die Dateigröße von MeineDatei.txt ermitteln und der Variablen intFileSize zuordnen.

#### Fazit

Die Weichen für die Skriptprogrammierung unter Windows-NT 4.0, Windows-2000 und Windows-98 sind von Microsoft in Richtung Windows-Scripting-Host und Vbscript beziehungsweise Jscript gestellt worden. Das ist auch gut so, denn die bislang von den Redmondern für Administratoren zur Verfügung gestellten Automatisierungsmechanismen für die genannten Betriebssysteme waren mehr als dürftig. Mit WSH können nunmehr beinahe alle erdenklichen Systemadministrationsaufgaben bewältigt werden und das vorgestellte FileSystemObject-Modell stellt nur einen kleinen Teil der Funktionalitäten dar. Über das Active-Directory-Service-Interface beispielsweise, welches ebenfalls mit WSH nutzbar ist, erhält der Systemverwalter die Möglichkeit die Administration von Benutzerkonten vollständig und mit relativ geringem Aufwand skriptgesteuert zu bewältigen, was eine enorme Arbeitserleichterung darstellt. Das gute dabei ist, daß die Funktionalitäten sowohl für Windows-NT 4.0 und Windows-2000, aber auch für Novell Netware und LDAP-kompatible Directory-Services zur Verfügung stehen. In einer der nächsten Ausgaben wird sich Network Computing speziell diesem Thema widmen und die Grundlagen vermitteln.

## NT herumkommandieren mit dem Windows-Scripting-Host

Objekte und Collections des FileSystemObject-Objekt-Modells

| Objekt/Collection | Beschreibung  |
|-------------------|---|
| Drives            | Collection, die Liste der an das System angeschlossenen physikalischen oder logischen Laufwerke unabhängig vom Typ ermittelt.   |
| Drive             | Objekt, über dessen Methoden sich Laufwerksinformationen ermitteln lassen, wie zum Beispiel   |
| Folders           | Collection, die eine Liste aller Unterverzeichnisse eines Verzeichnisses liefert.   |
| Folder            | Objekt mit Methoden und Eigenschaften, die das Erstellen, Löschen oder Verschieben von Verzeichnissen erlauben. Darüber hinaus werden Abfragen über Verzeichnisnamen, Pfade und verschiedene andere Eigenschaften unterstützt.  |
| Files             | Collection, die eine Liste aller Dateien eines Verzeichnisses zurück liefert.   |
| File              | Objekt mit Methoden und Eigenschaften, die das Erstellen, Löschen oder Verschieben von Dateien erlauben. Ebenso werden die Abfragen über Dateinamen, Pfade und verschiedene andere Eigenschaften unterstützt.   |
| FileSystemObject  | Das Hauptobjekt der Gruppe, welches Methoden und Eigenschaften zum Anlegen, Löschen, Abfragen von Laufwerken, Verzeichnissen und Dateien zur Verfügung stellt. Viele der Methoden, die mit diesem Objekt verknüpft sind, existieren bereits in identischer Form für andere Objekte. Um dem Programmierer jedoch die Arbeit zu erleichtern, wurden die entsprechenden Methoden einfach dupliziert. |
| TextStream        | Objekt, welches das Einlesen und Schreiben von Textdateien erlaubt.   |

```

C:\WTSRV\System32\cmd.exe

C:\>dir E:\USF\DIRK\Obs\GetDirSize\GetDirSizeNW\Listing2.obs C:\WTSRV
Microsoft (R) Windows Scripting Host Version 5.9 für Windows
Copyright (C) Microsoft Corporation 1996-1997. Alle Rechte vorbehalten..

Das Verzeichnis C:\WTSRV hat folgende Größe: C:\WTSRV = 449,3 MB

C:\>dir E:\USF\DIRK\Obs\GetDirSize\GetDirSizeNW\Listing2.obs C:\WTSRV /s
Microsoft (R) Windows Scripting Host Version 5.9 für Windows
Copyright (C) Microsoft Corporation 1996-1997. Alle Rechte vorbehalten..

Das Verzeichniss C:\WTSRV und dessen Unterverzeichnisse haben folgende Größen:
C:\WTSRV = 449,3 MB
C:\WTSRV\system32 = 186,9 MB
C:\WTSRV\system = 7,3 MB
C:\WTSRV\repair = 0,4 MB
C:\WTSRV\inf = 12,1 MB
C:\WTSRV\help = 7,2 MB
C:\WTSRV\fonts = 9,7 MB
C:\WTSRV\Config = 0,0 MB
C:\WTSRV\Cursors = 0,2 MB
C:\WTSRV\Media = 1,8 MB
C:\WTSRV\Application Compatibility Scripts = 0,3 MB
C:\WTSRV\Profiles = 16,6 MB
C:\WTSRV\ShellNew = 0,1 MB
C:\WTSRV\DOWNLOADED PROGRAM FILES = 0,1 MB
C:\WTSRV\Temporary Internet Files = 0,0 MB
C:\WTSRV\COOKIES = 0,0 MB
C:\WTSRV\History = 0,0 MB
C:\WTSRV\msdownload.tmp = 0,0 MB
C:\WTSRV\Tasks = 0,0 MB
C:\WTSRV\GetRoot = 0,0 MB
C:\WTSRV\Java = 7,4 MB
C:\WTSRV\Web = 0,1 MB
C:\WTSRV\Forms = 0,1 MB
C:\WTSRV\SendTo = 0,0 MB
C:\WTSRV\twain_32 = 0,2 MB
C:\WTSRV\pcTemp = 0,0 MB
C:\WTSRV\HELP.DAT = 0,3 MB
C:\WTSRV\SvcServicePackUninstall$ = 46,2 MB
C:\WTSRV\Offline Web Pages = 0,0 MB
C:\WTSRV\BooksOnline = 17,4 MB
C:\WTSRV\Samples = 0,0 MB
    
```

## Listing 1

Option Explicit

Dim strDirectory

Dim oArgs

Set oArgs = Wscript.Arguments

If oArgs.Count &lt; 1 Then

Call Usage

wscript.quit(1)

End If

strDirectory = oArgs(0)

Wscript.echo "Das Verzeichnis " & strDirectory & "  
hat folgende Größe: " &  
ShowFolderSize(strDirectory)

Sub Usage

    wscript.echo "Fehler - Es wurde kein Argument  
angegeben"

    wscript.echo "Syntax: GetDirSize  
<LW:>\<Verzeichnis>"

End Sub

Function ShowFolderSize (strDirectory)

Dim FileSystemObject, Folder

Dim intFolderSize

Dim strFolderSizeInfo

    Set FileSystemObject =  
    CreateObject("Scripting.FileSystemObject")

Set Folder =

FileSystemObject.GetFolder(strDirectory)

intFolderSize = Folder.Size

    intFolderSize = FormatNumber(intFolderSize /  
1048576, 1)

    strFolderSizeInfo = strDirectory & " = " &  
intFolderSize & " MB" & vbCRLF

ShowFolderSize = strFolderSizeInfo

End Function

## Listing 2

Option Explicit

Dim strDirectory

Dim oArgs

Dim booSubFolders

booSubFolders = FALSE

Set oArgs = Wscript.Arguments

If oArgs.Count &lt; 1 Then

Call Usage

wscript.quit(1)

End If

strDirectory = oArgs(0)

If oArgs.Count = 1 Then

    Wscript.echo "Das Verzeichnis " & strDirectory &  
" hat folgende Größe: " &  
ShowFolderSize(strDirectory, booSubFolders)

Else

booSubFolders = TRUE

    Wscript.echo "Das Verzeichniss " & strDirectory  
& " und dessen Unterverzeichnisse haben folgende  
Größen:" & vbCrLf & ShowFolderSize(strDirectory,  
booSubFolders)

End If

Sub Usage

    wscript.echo "Fehler - Es wurde kein Argument  
angegeben"

    wscript.echo "Syntax: GetDirSize  
<LW:>\<Verzeichnis> [/s]"

End Sub

Function ShowFolderSize (strDirectory,  
booSubFolders)

    Dim FileSystemObject, Folder, sFolder,  
Subfolders

Dim intFolderSize

## NT herumkommandieren mit dem Windows-Scripting-Host

```
Dim strFolderSizeInfo

Set FileSystemObject =
CreateObject("Scripting.FileSystemObject")
Set Folder =
FileSystemObject.GetFolder(strDirectory)
intFolderSize = Folder.Size
intFolderSize = FormatNumber(intFolderSize /
1048576, 1)
strFolderSizeInfo = strDirectory & " = " &
intFolderSize & " MB" & vbCRLF

If booSubFolders Then
Set SubFolders = Folder.SubFolders
For Each sFolder in SubFolders
intFolderSize = sFolder.Size
intFolderSize = FormatNumber(intFolderSize /
1048576, 1)
strFolderSizeInfo = strFolderSizeInfo &
sFolder & " = " & intFolderSize & " MB" & vbCRLF
Next
End If
ShowFolderSize = strFolderSizeInfo

End Function
```

### Zur Person

DIPL. ING. DIRK PELZER arbeitet als freier Consultant und Journalist in München. Er betreibt ein Storage Labor für verschiedene namhafte Fachzeitschriften. Zudem beschäftigt er sich mit Speichernetzen und Hochverfügbarkeit.