ROZDZIAŁ 6

Host skryptów systemu Windows (WSH, Windows Scripting Host)

WSH – jego zadaniem jest tworzenie skryptów w różnych językach np. Visual Basic Script, PerlScript, Python, Java Script, TCL(Tool Command Language), Rexx. WSH w systemie Windows 7 instalowany jest automatycznie, poza tym ładowane do niego są również dwa dodatkowe mechanizmy skryptów: VBScript a także Jscript. My będziemy korzystać z Visual Basic. Mechanizm hosta skryptów daje nam możliwość tworzenia skryptów znacznie bardziej złożonych niż w wierszu poleceń. Ponad to skrypty, które obsługiwane są przez WSH bazują na architekturze ActiveX co znaczy, iż można je uruchamiać poprzez przeglądarkę Internet Explorer po wcześniejszym umieszczeniu ich na stronie HTML. Istnieje opcja, że będą też uruchamiane po stronie serwera w środowisku sieci Intranet czy Internet.
W celu napisania skryptów nie jest konieczny żaden konkretny edytor, wystarczy wykorzystać dowolny edytor tekstowy np. Notatnik, a później zapisywać pliki z konkretnym rozszerzeniem dla VBasicScript - *.vbs, dla JScript - *.js.
Istnieją dwie wersje Hosta skryptów systemu Windows: wersja pracująca w trybie graficznym (wscript.exe), która posiada arkusz właściwości dla systemu Windows do ustawiania właściwości skryptów, a także wersja działająca w wierszu poleceń (cscript.exe) posiadająca przełączniki do ustawiania właściwości skryptów. Jesteśmy w stanie uruchamiać wybraną z tych dwóch wersji, pisząc w wierszu poleceń wscript.exe lub cscript.exe.
Windows Scripting Host udostępnia obiekty, z których można korzystać w celu uzyskania dostępu do różnych komponentów Windows. Jądrem modelu obiektowego Windows Scripting Hosta jest obiekt o nazwie WScript. WScript zawsze istnieje i jest dostępny. Obiekt WScript pozwala uzyskać informacje odnośnie obecnie wykonywanego skryptu, a także o samym Windows Scripting Hoście. Obiekt ten tworzy inne obiekty, których używa skrypt WSH. Do tworzenia kolejnych obiektów stosuje się metodę CreateObject.

Uruchamianie skryptów poprzez linię poleceń

CScript

Składnia **CScript i WScript** jest następująca

CScript nazwa_skryptu.rozszerzenie [opcja...] [argumenty...]
WScript nazwa_skryptu.rozszerzenie [opcja...] [argumenty...]

//B – tryb wsadowy, który pomija wyświetlanie błędów i monitów w skryptach
//D – włączane jest debugowanie aktywne
//E:aparat - wykonywany jest skrypt przy wykorzystaniu aparatu
//H:CScript – ustawia program CScript.exe jako domyślny host skryptów
//H:WScript – ustawia WScript.exe jako domyślny host skryptów
//I - tryb interaktywny (domyślny, przeciwieństwo trybu //B)
//Job:xxxx – wykonywane jest zadanie WSF
//Logo – wyświetlane jest logo (domyślnie)
//Nologo –logo nie jest wyświetlane: podczas wykonywania skryptu nie będzie wyświetlany transparent

//S - zapisuje bieżące opcje wiersza poleceń dla tego użytkownika
//T:nn - limit czasu w sekundach: maksymalny dozwolony czas wykonywania skryptu
//X - wykonywanie skryptu w debugerze
//U – używany jest standard Unicode dla przekierowań We/Wy z konsoli
Uwaga : Parametry hosta zawsze poprzedzaj dwoma ukośnikami //. Parametry skryptu jednym ukośnikiem /.

Zalety WSH

• Uzyskujemy bezpośredni dostęp do sterowania rejestrem
• Dokonujemy dowolnych operacji na systemach plików,
• Obiekty FileSystemObject oferują zbiory funkcji umożliwiających kopiowanie i przenoszenie zbiorów, modyfikowanie ich zawartości, zakładanie katalogów, folderów, odczytywanie ilości wolnego czy zajętego miejsca na dysku.
• Reprezentowany plik Obiekt File. Udostępniane są pola zawierające podstawowe informacje o pliku jak data jego utworzenia, ostatniej modyfikacji, skrócona nazwa, pełna ścieżka dostępu czy rozmiar pliku. Kolekcja plików to obiekt Files.
• Manipulując łańcuchami znakowymi uzyskujemy np. usuwanie znaków z końca łańcuchów, oddzielanie ciągów znaków, zmianę wielkości liter na duże itp.
• Napisany skrypt jest od razu gotowy do użycia, bez konieczności długotrwałego kompilowania.

Reguły VBScript

• Małe i duże litery nie są odróżniane
• W czasie wykonywania skryptu znaki ukryte za wyjątkiem końca wiersza są ignorowane
• Jeden wiersz to jedna instrukcja
• Użycie znaku (_)poprzedzonego spacją kontynuuje instrukcję w następnym wierszu
• Użycie znaku (:) umożliwia w jednym wierszu użyciu kilku instrukcji
• Komentarze umieszczamy po znaku (') lub za pomocą słowa kluczowego (Rem)

Ważniejsze metody i pola obiektu FileSystemObject

Metoda	Opis
BuildPath (ścieżka, nazwa)	Dodawana jest nazwa pliku do ścieżki, gwarantując przy tym odpowiednie wykorzystanie znaku separatora ścieżki
CopyFile plikŹródłowy, plikDocelowy, zastąpić	Kopiuje plikŹródłowy zapisując go jako plikDocelowy. Jeśli plikDocelowy już istnieje oraz jeśli argument zastąpić ma wartość TRUE, to kopiowany plik zastąpi istniejący plik o tej samej nazwie. Przy podawaniu nazw można używać znaków specjalnych(np. *,?,)
CopyFolder katalogŹródlowy, katalogDocelowy, zastąpić	Kopiuje katalogŹródłowy zapisując go jako katalogDocelowy. Jeśli katalogDocelowy już istnieje oraz jeśli argument zastąpić ma wartość TRUE, to kopiowany katalog zastąpi istniejący katalog o tej samej nazwie.

CreateFolder (nazwa)	Tworzony jest katalog o podanej nazwie
CreateTextFile (nazwa, zastąpić, unicode)	Tworzy plik tekstowy i zwraca egzemplarz obiektu TextStream skojarzony z tym plikiem. Argument zastąpić jest opcjonalną wartością logiczną określającą, czy należy zastąpić istniejącą już wersję pliku. Domyślenie argument ten ma wartość FALSE. Argument unicode jest opcjonalną wartością logiczną określającą, czy plik ma być zapisany w kodzie Unicode (TRUE), czy w kodzie ASCII(FALSE)
DeleteFile (ścieżka, wymuś)	Usuwa plik określony przy użyciu ścieżki. Na końcu ścieżki może być znak wieloznaczny. Wymuś jest argumentem opcjonalnym, przypisując mu wartość TRUE, wymuszamy usunięcie plików
DeleteFolder (ścieżka, wymuś)	Usuwa katalog określony przy użyciu ścieżki. Na końcu ścieżki może być znak wieloznaczny. Wymuś jest argumentem opcjonalnym, przypisując mu wartość TRUE, wymuszamy usunięcie katalogu
DriveExists (ścieżka)	Zwracana jest wartość logiczna informującą, czy istnieje napęd określony przez podaną ścieżkę
FileExists (ścieżka)	Zwracana jest wartość logiczna informującą, czy istnieje plik określony przez podaną ścieżkę
FolderExists (ścieżka)	Zwracana jest wartość logiczna informującą, czy istnieje katalog określony przez podaną ścieżkę
GetAbsolutePathName (określenieŚcieżki)	Zwracana jest pełną ścieżka na podstawie podanej specyfikacji np.: jeśli argument będzie miał wartość ”c:” , zostanie zwrócona pełna ścieżka dostępu do katalogu bieżącego na dysku C. Jeśli natomiast argument będzie miał wartość ”c:..”, to zostanie zwrócona ścieżka do katalogu nadrzędnego względem aktualnego katalogu.
GetBaseName (ścieżka)	Zwraca samą nazwę (bez rozszerzenia) pliku określonego za pomocą podanej ścieżki.
GetDrive (ścieżka)	Zwraca egzemplarz obiektu **Drive** reprezentujący napęd, na którym znajduje się plik określony za pomocą podanej ścieżki.
GetDriveName (ścieżka)	Zwraca łańcuch znaków zawierający nazwę napędu dla określonej ścieżki.
GetExtensionName (ścieżka)	Zwraca rozszerzenie pliku określonego za pomocą podanej ścieżki.
GetFile (ścieżka)	Zwraca egzemplarz obiektu **File** reprezentujący plik określony przez podaną ścieżkę.
GetFileName (ścieżka)	Zwraca nazwę pliku lub katalogu. A zatem wywołanie **GetFileName** („ c:\Inetpub\wwwroot\default.asp”) zwróci łańcuch znaków” default.asp” .

GetFolder (ścieżka)	Zwraca egzemplarz obiektu Folder reprezentujący katalog określony przez podaną ścieżkę.
ParentFolder	Zwraca egzemplarz obiektu reprezentujący Folder nadrzędny w stosunku do podanego folderu.
GetSpecjalFolder (określenie)	Zwraca ścieżkę do określonego katalogu specjalnego. Argument określenie można określić za pomocą następujących stałych: WindowsFolder, SystemFolder oraz TemporaryFolder. Stałym tym odpowiadają wartości: 0,1 oraz 2.
MoveFile plikŹródlowy, plikDocelowy	Przenoszony jest plikŹródlowy i zapisywany jako plikDocelowy.
MoveFolder katalogŻródlowy, katalogDocelowy	Przenoszony jest katalogŻródlowy i zapisywany jako katalogDocelowy
OpenTextFile (nazwa, tryb, utwórz, format)	Zwraca egzemplarz obiektu TextStream skojarzony z plikiem o podanej nazwie. Argument tryb określa, czy plik powinien być otwarty w trybie do odczytu(1), do zapisu (2), czy też w trybie do dopisywania (8). Argument utwórz określa, czy w razie gdyby plik nie istniał, to należy go utworzyć. Jeśli argument format ma wartość -1, to zawartość pliku będzie zapisywana w kodzie Unicode. Jeśli argument ten będzie miał wartość 0, to zawartość pliku zostanie zapisywana w kodzie ASCII. Wartość -2 tego argumentu oznacza, że zawartość pliku powinna być zapisywana w kodzie określonym przez ustawienia systemowe. **Format** opcjonalnie definiuje format w jakim będzie otwarty plik. Domyślnym trybem jest ASCII.

Składnia wybranych instrukcji

• zmienne- przechowują informację i noszą nazwy przyporządkowane w skrypcie. VBScript nie wymaga deklaracji zmiennej w skrypcie przed jej użyciem, chyba że używamy instrukcji Option Explicit. Wartość zmiennych może ulegać zmianom w trakcie wykonywania skryptu. Zmienne użyte w skrypcie obowiązują tylko podczas wykonywania skryptu, a następnie są usuwane z pamięci komputera. Deklaracja zmiennej w skrypcie wygląda następująco: Dim x. Nazwa zmiennej w VBScript: - może zawierać do 255 znaków - musi zaczynać się od znaku alfabetu - nie może zawierać spacji - nie może zawierać żadnych znaków specjalnych za wyjątkiem podkreślenia - musi być unikatowa w obrębie swojego działania

• Set – ustawia wartość zmiennej

• Stałe – to wartości wbudowane w język programowania. Za pomocą instrukcji **Const** można deklarować w skrypcie stałe.

Instrukcje warunkowe

If...Then to podstawowa instrukcja warunkowa. Po słowie kluczowym If następuje warunek (wyrażenie logiczne) i jeżeli warunek jest spełniony, to wykonywany jest zestawów instrukcji podanych po słowie Then (przy czym zestaw ten może zawierać tylko jedną instrukcję).

```
If warunek
then instrukcja
Else instrukcja
End if
```

Np.:

```
If PewnaData < Now Then
X = X+1 '// zwiększamy pewną bliżej nieznaną zmienną
MsgBox ''Data z przeszłości''
End If
```

lub:

```
If PewnaData < Now Then MsgBox ''Data z przeszłości''
```

Widzimy, że w przypadku większej liczby instrukcji do wykonania przy spełnionym warunku, należy konstrukcję zakończyć przez End If.

If...Then...Else to uogólnienie poprzedniej konstrukcji, jeżeli warunek nie jest spełniony, to wykonywana jest instrukcja (instrukcje) po słowie Else.
Np.:

```
If Wiek < 18 Then
MsgBox ''Przykro mi, ale nie możesz obejrzeć tego filmu.''
Else MsgBox ''Zapraszamy do kina !''
End If
```

W przypadku zagnieżdżonych instrukcji warunkowych, używamy słowa **ElseIf**.

Np.:

```
If RokStudiow = 2 Then
Stypendium = 130
ElseIf RokStudiow = 3 Then
Stypendium = 150
ElseIf RokStudiow = 4 Then
Stypendium = 170
Else
Stypendium = 200
End If
```

Pętle

Instrukcja For ... Next
Używana jest w tych sytuacjach, gdy przed rozpoczęciem pętli wiadomo, ile razy należy wykonać umieszczony w niej kod. Instrukcja wykonuje umieszczony w niej kod określoną ilość razy, inkrementując jednoczesnie specjalną zmienną nazywaną licznikiem pętli.

```
For licznik_pętli =wartość_początkowa to wartość_końcowa Step krok
     Blok kodu
Next
```

Licznik_pętli - jest zmienną numeryczną określającą, ile razy zawartość pętli została już wykonana. Na samym początku wykonywania pętli zmiennej licznik_pętli przypisywana jest wartość_posząтkowa. Podczas kolejnego wykonywania pętli zmienna ta będzie miała wartoś_początkowa + krok. Zakładając, że wartość_początkowa jest większa od zera, pętla będzie wykonywana aż do momentu, gdy wartość licznika_pętli stanie się większa od wartości_końcowej. Po każdym wykonaniu pętli, zmienna licznik_pętli jest powiększana o krok .W przypadku prostych pętli, instrukcja **For ... Next** jest wygodniejsza od instrukcji DoWhile...Loop, bowiem obsługuję inicjalizację i inkrementację licznika pętli. Wartość kroku może być liczbą ujemną. Także w takim przypadku liczba ta jest dodawana do licznika_pętli, ale wykonanie pętli kończy się w momencie, gdy wartość licznika_pętli będzie mniejsza od wartości_końcowej. Określanie wartości kroku jest opcjonalne. Jeśli zarówno słowo kluczowe Step, jak i wartość kroku nie zostaną podane, to automatycznie przyjmuje się, iż krok ma wartość 1. Jeśli wartość_posząтkowa jest większa od wartości_końcowej, a krok jest liczbą dodatnią, to blok kodu umieszczony wewnątrz pętli w ogóle nie zostanie wykonany. Do natychmiastowego przerwania pętli For...Next może posłużyć instrukcja Exit For, która natychmiast opuszcza pętlę.

Przykład:

```
For n = 1 to 5
MsgBox” To jest nasza pętla pętla”
Next
```

MsgBox zostanie wyświetlony na ekranie 5 razy.

Pętla **Do – Loop ”**wykonaj dopóki“ wykonuje zawarte w niej instrukcje tak długo, aż zostanie spełniony określony warunek. Tę pętlę można przerwać poleceniem Exit Do. Składnia tej pętli może być różna w zależności od tego, gdzie i w jaki sposób sprawdzany jest warunek pętli.

```
Do While warunek
Blok instrukcji do wykonania
Loop
```

Przykład - wyświetla całą zawartość pliku autoexec.bat.:

```
Set M = CreateObject(„ Scripting.FileSystemObject” )
Set Dat = M.OpenTextFile(„ c:autoexec.bat” )
Do
n = n + 1
Wiersz = Dat.ReadLine
Wscript.echo” Wiersz nr” & n &” :” & Wiersz
Loop Until Dat.AtEndOfStream = True
```

Przykładowe skrypty

Jako, że będziemy używać VBScriptu, który jest językiem programowania, jako student drugiego roku informatyki znasz już podstawy programowania zatem pisanie prostych skryptów w WSH nie powinno Ci sprawić problemu.

• Skrypt ten o nazwie wyswietl.vbs uruchamia Exploratora.exe

```
Set o = CreateObject(„ Wscript.shell” )
o.Run(„ Explorer.exe” )
```

• Skrypt poniższy petla.vbs wyświetli cztery razy wyrazy” Labolatorium hurra“

```
dim i
For i=1 to 4
Wscript.Echo” Labolatorium hura”
Next
```

• Skrypt przenies.vbs powoduje przeniesienie pliku dom.doc do katalogu archiwum

```
Set obiekt = CreateObject(„ Scripting.FileSystemObject” )
obiekt.MoveFile” C:\zapis\Nowy folder\dom.doc” ,” D:\archiwum”
```

• Skrypt poniższy o nazwie linijki.vbs zlicza linijki, w których jest jakiś znak (litera, cyfra, kropka, itp.). Podczas uruchamiania skryptu należy podać nazwę pliku, w którym będą zliczane znaki.

```
dim plik,otwarty,nazwa,argumenty,pom
set argumenty=Wscript.arguments
if argumenty.Count then
nazwa=argumenty.Item(0)
else
nazwa=InputBox(„ Podaj nazwe pliku:” )
if nazwa=„ „ then Wscript.Quit(1)
end if
set plik=CreateObject(„ Scripting.FileSystemObject” )
if not plik.FileExists(nazwa) then
MsgBox” Plik nie istnieje:” &nazwa,20
Wscript.Quit(2)
end if
set otwarty=plik.OpenTextFile(nazwa,1)
linijki=0
while not otwarty.AtEndOfStream
pom=otwarty.ReadLine
if Len(pom) <>„ 0” then
linijki=linijki+1
end if
Wend
otwarty.Close
MsgBox” Liczba linijek w pliku:” &nazwa&” wynosi” &linijki,75
REM znak & łączy łańcuchy
```

• **Usuwa plik z dysku:**

```
Dim fso
Set fso = CreateObject(„ Scripting.FileSystemObject” )
fso.DeleteFile(„ c:plik.txt” )
```

• **Usuwa folder z dysku:**

```
Dim fso
Set fso = CreateObject(„ Scripting.FileSystemObject” )
fso.DeleteFolder(„ c:folder” )
```

• **Tworzy plik na dysku:**

```
Dim fso, MyFile
Set fso = CreateObject(„ Scripting.FileSystemObject” )
Set MyFile = fso.CreateTextFile(„ C:plik.txt” ,True)
```

• **Tworzenie pliku tekstowego.**

```
MyFile.WriteLine(„ To jest plik.txt z zawartym textem” )
MyFile.WriteLine(„ by v1rhuz5” )
MyFile.Close
```

• **Tworzy folder na dysku:**

```
Dim fso, f
Set fso = CreateObject(„ Scripting.FileSystemObject” )
Set f = fso.CreateFolder(„ c:folder” )
```

• **Kopiuje plik:**

```
Dim fso
Set fso = CreateObject(„ Scripting.FileSystemObject” )
fso.CopyFile” c:plik.txt” ,” c:my documents”
```

• **Kopiuje folder:**

```
Dim fso
Set fso = CreateObject(„ Scripting.FileSystemObject” )
fso.CopyFolder” c:plik” ,” d:”
```

• **Obliczenie silni (!) liczby naturalnej.**

Czy wiecie co to jest silnia? Nie? No to śpieszę z wyjaśnieniem: silnia liczby naturalnej (czyli całkowitej i dodatniej) n - to iloczyn kolejnych liczb naturalnych od 1 do n. Dodatkowo przyjmujemy, że 0! = 1. Możemy to zapisać następująco:

n! = 1*2*...*(n-1)*n, n>0

0! = 1

Kod programu wygląda następująco:

```
Option Explicit
Const strNazwaMakra As String =„ Obliczanie silni”

Sub Silnia()
'Deklaracje zmiennych lokalnych
Dim dblSilnia As Double
Dim strLiczba As String
Dim lngLiczba As Long
Dim i As Long

'Pętla Do...Loop, wykonywana dopóty użytkownik poda prawidłowe dane
Do
'Wyświetlenie okienka do pobrania danych.
strLiczba = InputBox(„Podaj liczbę dodatnią, mniejszą od stu”,
strNazwaMakra)

'Jeżeli użytkownik nie podał żadnego ciągu
'lub wcisnął przycisk” Cancel” kończymy pracę programu.
If strLiczba = vbNullString Then End

'Wychodzimy z pętli Do...Loop, jeżeli liczba jest prawidłowa
If Not IsNumeric(strLiczba) Then
MsgBox ”To nie jest liczba.” , vbInformation, strNazwaMakra
Else
If Val(strLiczba) < 0 Or Val(strLiczba) > 100 Or _
(Val(strLiczba) <> CLng(Val(strLiczba))) Then _
MsgBox ”Nieprawidłowa liczba.” , vbInformation, strNazwaMakra
Else Exit Do
End If
Loop

'Zamieniamy ciąg (typ String) na liczbę typu Long (całkowitą)
lngLiczba = CLng(Val(strLiczba))

'ustawienie wartości początkowej zmiennej dblSilnia
dblSilnia = 1

'Jeżeli liczba jest większa od zera, obliczamy silnię w pętli.
'Pytanie za sto punktów:
'Dlaczego napisałem pętlę For i = 2 ... a nie
'For i = 1 ..., tak jak jest w definicji silni?
If lngLiczba > 0 Then
For i = 2 To lngLiczba
dblSilnia = dblSilnia * i
Next i
End If

'wyświetlenie wyniku.
MsgBox ”Wynik:” & lngLiczba &”! =„ & dblSilnia, vbInformation
```

Zadania do samodzielnego wykonania

Przećwicz powyższe skrypty.

1. Napisz, krótki skrypt (stosując oczywiście WSH), który wyświetli np. trzy unikalne nazwy w celu nadania ich innym tworzonym plikom.
2. Napisz skrypt, który obliczy i wyświetli wynik wyrażenia np. 1000+5^3*8/16
3. Napisz skrypt, który wyświetli liczby od cyfry 0 do 25 z odstępem pięciu cyfr.
4. Zmodyfikuj program linijki.vbs tak, aby obok liczby linijek zapisanych wyświetlał liczbę linijek pustych (nie zapisanych)
5. Napisz skrypt, który wyświetli po kolei nazwy podfolderów wraz z ich rozmiarami w katalogu np.: Windows
6. Napisz skrypt, który zliczy wszystkie podfoldery w katalogu np. Program Files. W wyniku ma wyświetlić tylko liczbę podfolderów.
7. Napisz skrypt, którego zadaniem będzie przeniesienie napisanych przez Ciebie skryptów do katalogu WSHKAT na dysku C.
8. Napisz skrypt, którego zadaniem będzie skopiowanie katalogu WSHKAT na dysk USB.
9. Napisz skrypt, który usunie katalog WSHKAT z dysku C.