Politechnika Łódzka

Instytut Informatyki Stosowanej

Programowanie w języku VBS

Dr inż. Piotr Urbanek

Do czego służą skrypty?

- Do regularnego wykonywania serii zadań administracji systemem
- Do wykonywania serii zadań administracji systemem na wielu komputerach
- Do scalania i przetwarzania danych wynikowych pobranych z komputera
- Do uruchamiania zadań, gdy nie ma nikogo, kto mógłby podjąć interakcję z interfejsem graficznym
- Do powtarzania wielokrotnie tych samych akcji podczas każdego uruchomienia zadania

Rozszerzenia WSH

- VBScript – Visual Basic, Scripting Edition. Język skryptowy dostępny domyślnie w systemie Microsoft Windows.
- WSH – Windows Script Host. Środowisko uruchamiania skryptów.
- WMI – Windows Management Instrumentation. Technologia dostarczająca zasoby służące do zarządzania systemami operacyjnymi Windows przy pomocy skryptów.
- ADSI – Active Directory Service Interfaces. Technologia dostarczająca zasoby do zarządzania Active Directory oraz innymi usługami katalogowymi przy pomocy skryptów.

VBScript

Visual Basic Scripting Edition (VBScript) należy do rodziny języków programowania Visual Basic. VBScript jest okrojoną wersją języka VBA (Visual Basic for Applications). VBA jest składnikiem pakietu Visual Basic, oraz wielu aplikacji firmy Microsoft (np. Office) i innych producentów oprogramowania (np. AutoCAD firmy AutoDesk).

VBScript jest używany przede wszystkim w środowiskach:

- Active Server Pages (ASP).
- ASP to technologia firmy Microsoft, dzięki której można tworzyć skrypty działające po stronie serwera dla Internet Information Server (IIS).
- Windows Script Host (WSH).
- WSH to technologia skryptowa umożliwiająca automatyzację zadań wykonywanych w systemie Windows.
- Windows Script Components (WSC).
- Technologia WSC pozwala na łatwe tworzenie komponentów COM, które mogą być następnie użyte we wszystkich aplikacjach wykorzystujących takie komponenty.
- Microsoft Internet Explorer (IE).
- W środowisku IE, VBScript służy do pisania skryptów działających po stronie klienta.

Typy danych

VBScript posiada jeden typ danych - Variant.

Typ Variant to specjalny rodzaj typu danych który może przechowywać różne rodzaje informacji, w zależności od sposobu wykorzystania. Ponieważ Variant jest jedynym typem danych wykorzystywanym w VBScript, jest on zwracany przez wszystkie funkcje VBScript.

Deklarowanie zmiennych

Zmienne deklaruje się jawnie przy użyciu instrukcji Dim, Public, Private np.

Dim Stopnie_Celsjusza

W jednej instrukcji możemy zadeklarować kilka zmiennych oddzielając je przecinkami np.

Dim Góra, Dół, Lewo, Prawo

Podtypy danych typu Variant

Nazwa typu	Opis typu
Empty	Zadeklarowana, ale nie zainicjalizowana zmienna.
Null	Nieprawidłowe dane.
Boolean	Wartość logiczna (True lub False).
Byte	8-bitowy numeryczny typ danych. Zakres od 0 do 255
Integer	16-bitowy numeryczny typ danych. Zakres od -32 768 do 32 767
Currency	Wartość walutowa. Liczba z ustaloną kropką dziesiętną. Zakres od -922 337 203 685 477.5808 Do 922 337 203 685 477.5807
Long	32-bitowy numeryczny typ danych. Zakres od -2 147 483 648 do 2 147 483 647
Single	Wartość zmiennoprzecinkowa pojedynczej precyzji. Zakres ujemne:-3.402823E38 do -1.401298E-45 dodatnie: 1.401298E-45 do 3.402823E38
Double	Wartość zmiennoprzecinkowa podwójnej precyzji. Zakres ujemne:-0.797693134862 32E308 do -4.940656458412 47E-324 dodatnie: 4.940656458412 47E-324 do 1.797693134862 32E308
Date	Wartość daty lub czasu. Zakres od 1/1/100 do 31/12/9999
String	Łańcuch znaków zmiennej długości od 0 do 2 miliardów znaków.
Object	Adres obiektu.
Error	Błędna wartość liczby.

Ograniczenia w deklarowaniu zmiennych

Nazwa zmiennej w języku VBScript:

- może zwierać do 255 znaków;
- musi zaczynać się od znaku alfabetu;
- nie może zawierać spacji;
- nie może zawierać żadnych znaków specjalnych (tj. niealfabetycznych, nienumerycznych) z wyjątkiem znaku podkreślenia;
- musi być unikatowa w obrębie swojego zakresu.

Zasięg zmiennych

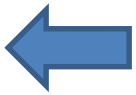
Zmienna poziomu procedury – zmienna zadeklarowana wewnątrz procedury, zmienna lokalna

Zmienna poziomu skryptu – zmienna zadeklarowana na zewnątrz procedury. Jest widoczna w całym skrypcie i pełni rolę zmiennej globalnej.

Deklarowanie tablic

Dim Tablica(10) - wektor 10 elementów.
Dim Tablica2D(5, 10) - tablica zawierająca 5 wierszy i 10 kolumn

Odwoływanie się do elementów tablicy
Tablica(0)=123
Tablica(1)=234
Tablica(2)=345
...
Tablica(10)=901



Indeksacja tablic od „0”

Redefinicja rozmiaru tablicy

ReDim Tablica(25)
...
ReDim Preserve Tablica(30)

Deklarowanie tablic dynamicznych

Dim Tablica()

ReDim Inna_Tablica()

Deklaracja stałych

Const Napis = "To jest łańcuch znaków"
Const Wiek = 33
Const Data_urodzin = #6-1-97#

Pierwszeństwo wykonywania operatorów

Najpierw wykonywane są operacje **arytmetyczne**, następnie operacje **porównania**, a **na** końcu operacje **logiczne**. Operatory porównania mają jednakowy priorytet tzn. są wykonywane od lewej do prawej, w kolejności występowania w wyrażeniu. Operatory arytmetyczne i logiczne są wykonywane w następującej kolejności:

W przypadku gdy **mnożenie** i **dzielenie** występują razem w wyrażeniu, wykonywane są w kolejności występowania od lewej do prawej. Podobna sytuacja ma miejsce gdy w wyrażeniu występują dodawanie i odejmowanie.
Operator **łączenia łańcuchów** znaków nie jest operatorem arytmetycznym, ale w pierwszeństwie operatorów występuje po operatorach arytmetycznych, a przed operatorami porównania. Operator **Is** jest operatorem równości referencji obiektowych. Nie porównuje on obiektów, ani ich wartości; sprawdza jedynie czy adresy (odwołania) dwóch obiektów dotyczą tego samego obiektu.

Operator	Opis
^	potęgowanie
-	unarna negacja
*	mnożenie
/	dzielenie
\	dzielenie całkowite (bez reszty)
Mod	modulo
+	dodawanie, konkatenacja łańcuchów znaków (nie zaleca się wykonywania konkatenacji za pomocą znaku +)
-	odejmowanie
&	konkatenacja łańcuchów znaków

Operator	Opis
=	równość, operacja przypisania
<>	różny od
<	mniejszy niż
>	większy
<=	mniejszy niż lub równy
>=	większy niż lub równy
Is	równość referencji obiektowych

Operator	Opis
Not	logiczna lub bitowa negacja
And	logiczna lub bitowa koniunkcja
Or	logiczna lub bitowa alternatywa
Xor	logiczna lub bitowa alternatywa wyłączająca
Eqv	logiczna lub bitowa ekwiwalencja
Imp	logiczna lub bitowa implikacja

Operatory porównania

```
if value="0" then
   msgbox value
elseif value="1" then
   msgbox value
elseif value="2" then
   msgbox value
else
   msgbox "value out of range!"
end if
```

```
select case wyrażenie_testujące
case wartość1
    instrukcja 1
    instrukcja 2
    instrukcja n
case wartość2
    instrukcja 1
    instrukcja 2
    instrukcja n
case wartośćn
    instrukcja 1
    instrukcja 2
    instrukcja n
 case else
    instrukcja 1
    instrukcja 2
    instrukcja n
end select
```

Pętla Do … Loop

```
Sub WarunekDoWhile()
 Dim licznik,x
 licznik = 0
 x = 20
 Do While x > 10
  x = x - 1
  licznik = licznik + 1
 Loop
 MsgBox "Pętla wykonała " & licznik & „_
przebiegów."
End Sub
```

Pętla Do Until

```
Sub WarunekDoUntil()
 Dim licznik,x
 licznik = 0
 x = 20
 Do Until x = 10
  x = x - 1
  licznik = licznik + 1
 Loop
 MsgBox "Pętla wykonała " & licznik & „_
przebiegów."
End Sub
```

Instrukcja Exit Do

```
Sub Przyklad_ExitDo()
  Dim licznik, i
    licznik = 0
    i = 9
    Do Until  i = 10
      i = i - 1
      licznik = licznik + 1
      If i < 10 then exit do
    loop
    msgbox "pętla wykonała " & licznik & " powtórzeń."
End Sub
```

While...Wend

Wykonuje instrukcję (grupę instrukcji)
dopóki warunek jest równy True.

```
While warunek
 instrukcja
 instrukcja
 .
 .
 .
Wend
```

```
Dim licznik
licznik = 0
While licznik < 10
 licznik = licznik + 1
 msgbox "Petla wykonana: " & licznik
Wend
```

For...Next

Pętli For...Next używamy aby wykonać pewien blok instrukcji określoną liczbę razy. Pętle for używają zmiennej licznika której wartość jest zwiększana lub zmniejszana po każdym wykonaniu pętli.

```
dim j,wartosc
 for j=2 to 10 step 2
   wartosc = wartosc + j
 next
 msgbox "suma wynosi " & wartosc
```

```
dim x,wartosc
 for x=16 to 8 step -2
   wartosc = wartosc + x
 next
 msgbox "suma wynosi " & wartosc
```

For Each...Next

Instrukcja For **Each...Next** jest bardzo podobna do instrukcji For...Next. Zamiast jednak wykonywać instrukcje określoną ilość razy, pętla **For Each...Next** powtarza operacje dla każdego elementu kolekcji obiektów, lub dla każdego elementu tablicy. Jest ona szczególnie przydatna, jeżeli nie znamy liczby elementów kolekcji (tablicy).

```
dim tab(2)
tab(0)=4
tab(1)=2
tab(2)=23
for each element in tab
msgbox "Element nr: " & element
next
```

Kategoria	Funkcje
Tablice	Array, IsArray, LBound, UBound
Konwersja	Abs Asc, AscB, AscW Chr, ChrB, ChrW CBool, CByte, CCur, CDate CDbl, CInt, CLng, CSng, CStr DateSerial,DateValue Hex, Oct, Fix, Int, Sgn TimeSerial, TimeValue
Data/Czas	Date, Time, Now DateAdd, DateDiff, DatePart DateSerial,DateValue Month, MonthName Weekday, WeekdayName Year, Day, Hour, Minute, Second TimeSerial, TimeValue

Wyrażenia	Eval
Formatowanie łańcuchów	FormatCurrency, FormatDateTime FormatNumber, FormatPercent
Wejście/Wyjście	InputBox, MsgBox, LoadPicture
Matematyka	Atn, Cos, Sin, Tan Exp, Log, Sqr, Rnd
Inne	RGB
Zaokrąglanie	Abs, Sgn, Fix, Int, Round
Silnik skryptów	ScriptEngine ScriptEngineBuildVersion ScriptEngineMajorVersion ScriptEngineMinorVersion

Łańcuchy znaków	Asc, AscB, AscW Chr, ChrB, ChrW Filter, InStr, InStrB, InStrRev Join, Split Len, LenB LCase, UCase Left, LeftB, Mid, MidB, Right, RightB Replace, Space StrComp, String, StrReverse LTrim, RTrim, Trim
Typy zmiennych	IsArray, IsDate, IsEmpty IsNull, IsNumeric, IsObject TypeName, VarType

Wyświetlanie informacji na ekranie

MsgBox "text",64,"tytuł"

0 - OK
1 - OK Anuluj
2 - Przerwij Ponów Próbę Zignoruj
3 - Tak Nie Anuluj
4 - Tak Nie
5 - Ponów Próbę Anuluj
16 - Błąd krytyczny
32 - Pytanie
48 - Ostrzeżenie
64 - Informacja

```
Dim komunikat     //deklaracja zmiennej
Set komunikat = Script.CreateObject("WScript.Shell")
komunikat.popup "text"
```

Tworzenie folderu z datą wraz z kopiowaniem do niego plików

```
StrMonth = Month(Date)
If Len(strMonth) = 1 Then
strMonth = "0" & strMonth
End If
StrDay = Day(Date)
If Len(strDay) = 1 Then
strDay = "0" & strDay
End If
StrYear = Year(Date)
' NAZWA TWORZONEGO FOLDERU + DATA SYSTEMOWA
strFolderName = "X:\ścieżka\nazwa folderu" & strDay & "." & strMonth & "." & StrYear
Set objFSO = CreateObject("Scripting.FileSystemObject")
' SPRAWDŹ CZY FOLDER ISTNIEJE, JEŚLI TAK-KOPIUJ PLIKI
if objFSO.FolderExists(strFolderName) then
objFSO.CopyFile "X:\ścieżka\nazwa folderu\xxx.txt", strFolderName & "\"
objFSO.CopyFile "X:\ścieżka\nazwa folderu\x.txt", strFolderName & "\"
' SPRAWDŹ CZY FOLDER ISTNIEJE, JEŚLI NIE-UTWÓRZ FOLDER I KOPIUJ PLIKI
else
Set objFolder = objFSO.CreateFolder(strFolderName)
objFSO.CopyFile "X:\ścieżka\nazwa folderu\xxx.txt", strFolderName & "\"
objFSO.CopyFile "X:\ścieżka\nazwa folderu\x.txt", strFolderName & "\"
end if
```

Mapowanie udziału do komputera w domenie

```
Option Explicit
On error resume next
Dim objNetwork
Dim objShell
Dim appShell
Dim userLink
Set objNetwork = CreateObject("Wscript.Network")
Set objShell = CreateObject("Wscript.Shell")
Set appShell = CreateObject("Shell.Application")
userDesktop = objShell.SpecialFolders("Desktop")
'MAPOWANIE DYSKU
objNetwork.MapNetworkDrive
"h:","\\nazwa_servera\nazwa_udziału", true
appShell.NameSpace("h:").Self.Name = "Podmapowany"
'TWORZENIE SKRóTU NA PULPICIE DO PODMAPOWANEGO DYSKU
set userLink = objShell.CreateShortcut(userDesktop &
"\Podmapowany.lnk")
userLink.TargetPath = "h:"
userLink.Save
```

Obiekty standardowe w WSH

- Obiekt WScript
- Obiekt WshArguments
- Obiekt WshEnvironment
- Obiekt WshNetwork
- Obiekt WshShell

Wybrane rozszerzenia WSH

- Obsługa strumieni
  - Strumień StdOut
  - Strumień StdIn
- Obsługa plików i folderów
- Inne zastosowania WSH

Obiekt WScript

Własności WScript przechowują dane o wykonaniu i parametrach uruchomienia skryptu. Metody WScript pozwalają na sterowanie wykonaniem skryptu oraz na zarządzanie obiektami.

Metody WScript:
CreateObject tworzy obiekt określony nazwą.