Programação VBScript

REDES DE COMPUTADORES

Prof. Dr. João Paulo Aramuni

UNIVERSIDADE FUMEC

O que é o VBScript?

* **VBScript** (acrônimo de Microsoft Visual Basic Scripting Edition) é um sub-sistema do Visual Basic usado em Active Server Pages e em **Windows Scripting Hosts** como uma linguagem de aplicação universal (general-purpose).

O que é o VBScript?

* Quando combinado a tecnologias como a instrumentação de gerenciamento do Windows (WMI) e as interfaces de serviço do Active Directory (ADSI), o Microsoft® Visual Basic® Scripting Edition (VBScript) torna-se uma linguagem de scripts poderosa.

O que é o VBScript?

* Usando o VBScript juntamente com essas tecnologias, você pode escrever um script de aproximadamente 10.000 linhas, um script completo com tratamento de erros, sub-rotinas e outras construções de programação avançadas.

* Por sua vez, esse script fornece a você controle completo sobre todos os aspectos do seu ambiente de computação.

O que é o VBScript?

* No entanto, o que torna o VBScript uma ferramenta tão útil para administradores de rede é o fato de que não é preciso criar soluções tão elaboradas e complicadas. Reconhecidamente, scripts podem ser usados para criar uma solução de gerenciamento empresarial abrangente.

O que é o VBScript?

* Porém, talvez o mais importante seja o fato de que os scripts também podem ser usados do seguinte modo: um administrador de rede pode gastar alguns minutos digitando algumas linhas de código no Bloco de Notas e criar instantaneamente uma solução personalizada para um problema específico.

Variáveis

* O VBScript faz tipagem dinâmica de variáveis, dessa forma não precisamos ficar explicitando o tipo de nossas variáveis. Para declará-las, basta escrever o nome da variável: X, Y, A, B....

* Caso seja necessário indicar o tipo das variáveis, use no topo de seus scripts:

Option Explicit

Variáveis

* O VBScript, assim como o VB clássico, utiliza a instrução **Dim** para declarar variáveis e reservar espaço na memória para as mesmas.

* Exemplos:
  * **Dim** Nomes (9) ' Declara um array de 10 elementos
  * **Dim** Nomes () ' Declara um array dinâmico
  * **Dim** num, num2' Declara duas variáveis.

Variáveis

* Para setarmos o conteúdo de um objeto não primitivo, não basta utilizarmos o operador "=" como estamos habituados a utilizar em variáveis de tipos primitivos (inteiro, string, etc). Exemplo: A = B + C;

* Neste caso, devemos usar o **SET**.
  * Exemplo: set A = WScript.CreateObject("Wscript.Network")
  * Sempre que houver o comando de instânciamento CreateObject estaremos tratando um tipo não primitivo de variável.

Comandos condicionais

‘ Se a condição for atendida....então....

If CONDICAO Then

    Comandos...

End if

‘ Negação...

If not CONDICAO then

    Comandos...

End if

Comandos condicionais

‘ Se a condição for atendida....então....senão....

If CONDICAO Then

Comandos...

else

Outros comandos...

End if

Comandos condicionais aninhados

‘ Se a condição for atendida....então....senão si....

If CONDICAO Then

Comandos...

elseif CONDICAO Then

Outros comandos...

else

Outros comandos...

End if

Comandos condicionais aninhados

```
‘ Se a condição for atendida....então....senão si....
If CONDICAO Then
    Comandos...
    elseif CONDICAO Then
    Outros comandos...
    else
        If CONDICAO Then
            Outros comandos...
        End If
End If
```

Loops

* Quando queremos executar mais de uma vez o mesmo comando, precisamos utilizar um loop. Trabalharemos apenas com loops finitos, que são os mais utilizados.

Loops

* For

```
For i = 1 To 10
    WScript.Echo "Contador: " & i
Next

For i=10 to 1 step -1
    WScript.Echo "Contador Regressivo: " & i
Next
```

Loops

* While

```
Do While contador <= 10
        WScript.Echo (contador)
        contador = contador +1
Loop
```

Loops

* While

```
Do
        WScript.Echo (contador)
        contador = contador +1
Loop While contador <= 10
```

Loops

* While

```
While contador <= 10
        WScript.Echo (contador)
        contador = contador +1
Wend
```

Funções

* Em VBScript temos dois tipos de procedures:

Sub procedure

Function procedure

* As procedures do tipo Sub não retornam nenhum valor
* As procedures do tipo Function retornam valor

Funções

```
SUB minhaFuncao(x, y)
        comandos...
end SUB
```

Para chamar a função:

```
call minhaFuncao(a, b)
```

Funções

Function minhaFuncao(x , y)

Comandos...

minhaFuncao = Algum valor de retorno...pode ser do tipo inteiro ou string por exemplo, não é necessário explicitar o tipo do retorno, ex:

minhaFuncao = 0

ou **minhaFuncao** = "Olá Mundo"

End Function

Para chamar a função:

minhaFuncao()

Funções

* Ok mas.... Porquê precisamos de funções?

* De acordo com as boas práticas de programação, devemos exportar a lógica de negócio de nossos programas para funções separadas.

* Assim não misturamos o conteúdo de nossos programas com o conteúdo de funções que podem, posteriormente, serem reaproveitadas por outros programas.

Funções Nativas

* Para não termos que reinventar a roda, o VBS nos disponibiliza algumas funções nativas, referentes à funcionalidades que usamos com frequência, para não termos que ficar sempre criando novos métodos para a mesma finalidade.

Funções Nativas

* Podemos chamar essas funções diretamente, pois elas já foram desenvolvidas e estão disponíveis para serem chamadas por qualquer vbscript.

* Documentação:

* https://msdn.microsoft.com/en-us/library/3ca8tfek(v=vs.84).aspx

Funções Nativas

* Exemplos:
* **LCase** (lowercase) – retorna a string em letras minúsculas.
* **UCase** (uppercase) – retorna string em letras maiúsculas.
* **StrReverse** – retorna a string invertida. (JOAO -> OAOJ)
* **Len** (length) – retorna o tamanho da string. (JOAO -> 4)
* **Mid** – retorna um intervalo de caracteres da string.
* **Replace** – substitui uma palavra ou letra pela outra caso a encontre na sua string.

* Obs: Utilize a instrução **Dim** para definir suas variáveis ao usar funções nativas. Estas funções geralmente esperam variáveis tipadas. Nestes casos tipamos as variáveis para evitar erros de conversão.

Funções Nativas

```
Option Explicit

Dim msg1, msg2

msg1 = "Primeira Mensagem de Teste"
msg2 = "Segunda Mensagem de Teste"

WScript.Echo "Tudo em minusculo: " & LCase(msg1)

WScript.Echo "Tudo em maisculo: " & UCase(msg2)

WScript.Echo "Invertido: " & StrReverse(msg1)

WScript.Echo "Numero de caracteres da segunda string: " & Len(msg2)

WScript.Echo "Primeiros 5 caracteres da primeira string: " & Mid(msg1, 1, 5)

WScript.Echo "Substitua Teste por Erro: " & Replace(msg2,"Teste","Erro")
```

Desafio

* Agora que você já conhece a função **StrReverse**, utilize-a para verificar se uma determinada palavra é palíndromo.

* Exemplo: ARARA, OCO, RIR, SOCOS, REVER...

Exercícios

* exer01_msgbox.vbs
* exer02_imc.vbs
* exer03_cont.vbs
* exer04_avg.vbs
* exer05_milhao.vbs
* exer06_random.vbs
* exer07_mountyHall.vbs
* exer08_byVal_byRef.vbs
* exer09_fatorial.vbs
* **exer10_fibonacci.vbs**

Desafio

* A **Sequência de Fibonacci** é uma sequência de números inteiros, começando normalmente por 0 e 1, na qual, cada termo subsequente (*número de Fibonacci*) corresponde a soma dos dois anteriores.

* A sequência recebeu o nome do matemático italiano Leonardo de Pisa, mais conhecido por Fibonacci (contração do italiano *filius Bonacci*), que descreveu, no ano de 1202, o crescimento de uma população de coelhos, a partir desta. Tal sequência já era no entanto, conhecida na antiguidade.

Desafio

* Escreva um script vb que possua uma função que receba um número inteiro do usuário e calcule o número de Fibonacci correspondente na sequencia.

* Dica: Pesquisa como é feito o cálculo em outras linguagens, como linguagem C, e tente traduzir a lógica para o vbs.

* Existem dois modos de se fazer, o modo Iterativo (utilizando loop), e o modo recursivo (utilizando empilhamento de memória, sem loop).

* O modo recursivo vale **1 ponto extra !!!**

Desafio

* Tente agora escreva um script vb que tenha a mesma função que o anterior, porém que imprima na tela os números de Fibonacci de 1 até a posição 16 da sequência.

* Solução: **exer11_fibonacciSeq.vbs**

Obrigado.

Contato: joaopauloaramuni@gmail.com

UNIVERSIDADE FUMEC

50 Anos