eBook Gratuit

APPRENEZ asp-classic

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#asp-classic

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: asp-classic

It is an unofficial and free asp-classic ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official asp-classic.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec asp-classic

Remarques

Active Server Pages (ASP), également appelé ASP classique ou ASP classique, était le premier moteur de script côté serveur de Microsoft pour les pages Web générées dynamiquement. L'introduction de ASP.NET a conduit à utiliser le terme Classic ASP pour la technologie d'origine.

Le langage de script côté serveur par défaut pour ASP est VBScript. Les pages générées sont destinées à être visualisées dans un navigateur, elles utilisent généralement le balisage HTML et le style CSS.

[1] ASP n'est pas installé par défaut sur ces versions d'IIS. Vous devez entrer dans les fonctionnalités du gestionnaire de serveur et ajouter ASP.
Voir ASP classique non installé par défaut sur IIS 7.0 et versions ultérieures

Versions

IIS	ASPIC	Libéré
3.0	1.0	1996-12-01
4.0	2.0	1997-09-01
5.0	3.0	2000-11-01
6,0	3.0	2003-01-01
7.0	3.0 [1]	2008-01-01
7.5	3.0 [1]	2009-01-01
8.0	3.0 [1]	2012-01-01

Examples

Bonjour le monde

```
<!doctype html>
<html>
  <head>
    <title>Example Page</title>
  </head>
  <body>
<%
  'This is where the ASP code begins
  'ASP will generate the HTML that is passed to the browser
```

```
  'A single quote denotes a comment, so these lines are not executed
  'Since this will be HTML, we included the html and body tags
  'for Classic ASP we use Response.Write() to output our text
  'like this

  Response.Write ("Hello world")

  'Now we will end the ASP block and close our body and html tags
%>
  </body>
</html>
```

Lorsque la réponse est envoyée du serveur au navigateur, la sortie sera comme ceci:

```
<!doctype html>
<html>
  <head>
    <title>Example Page</title>
  </head>
  <body>
 Hello world
  </body>
</html>
```

Structure d'une page ASP simple

```
<%@ Language="VBScript" CodePage = 65001 %>
<%
Option Explicit
Response.Charset = "UTF-8"
Response.CodePage = 65001
%>
<!doctype html>
<html>
  <head>
    <title>My First Classic ASP Page</title>
  </head>

  <body>
    <%="Hello World"%>
  </body>
</html>
```

Ceci est un exemple très basique d'une page ASP classique qui renvoie l'expression "Hello World" au navigateur avec le reste du code HTML standard. Les portions HTML sont statiques, c'est-à-dire que le serveur les envoie au navigateur tel quel. Les parties délimitées par `<% %>` sont ce que le serveur traitera réellement avant de l'envoyer au client.

Notez que la syntaxe `<%="stuff"%>` est un raccourci pour `<%Response.Write "stuff"%>` .

Lire Commencer avec asp-classic en ligne: https://riptutorial.com/fr/asp-classic/topic/1094/commencer-avec-asp-classic

Chapitre 2: Connexion à une base de données

Introduction

Classic ASP utilise une technologie appelée ActiveX Data Objects pour accéder à des sources de données externes. La bibliothèque ADODB fournit trois objets principaux à cette fin, `ADODB.Connection` , `ADODB.Command` et `ADODB.Recordset` .

Examples

Remplir une liste déroulante de la base de données

(Caveat emptor: il y a beaucoup de programmeurs qui entrent dans des connotations absolues s'ils rencontrent le code qui utilise des jeux d'enregistrements au lieu de commandes et de procédures stockées.)

```
<%
dim rs, sql
dim SelectedUser
SelectedUser = request.form("user")
if IsNumeric(SelectedUser) then
    SelectedUser = CLng(SelectedUser)
else
    SelectedUser = 0
end if
%>
...
<p>Select a user: <select name="user" size="1">
<%
sql = "SELECT id, displayname FROM users WHERE active = 1 ORDER BY displayname"
set rs = server.createobject("ADODB.Recordset")
rs.open sql,"[connection string stuff goes here]",1,2
do until rs.eof
    response.write "<option value='" & rs("id") & "'"
    if rs("id") = SelectedUser then response.write " selected"
    response.write ">" & rs("displayname") & "</option>" & vbCrLf
    rs.Movenext '<- VERY VERY IMPORTANT!
loop
rs.close
set rs = nothing
%>
</select></p>
...
```

Lire Connexion à une base de données en ligne: https://riptutorial.com/fr/asp-classic/topic/4991/connexion-a-une-base-de-donnees

Chapitre 3: En boucle

Examples

Pour boucle

Dans ASP classique, nous pouvons spécifier une boucle for avec le mot-clé *for* . Avec l'instruction for, nous avons besoin de l'instruction *suivante* qui incrémentera le compteur.

```
For i = 0 To 10
    Response.Write("Index: " & i)
Next
```

Le mot-clé *step* peut être utilisé pour modifier la manière dont l'instruction *suivante* modifiera le compteur.

```
For i = 10 To 1 Step -1
    'VBS Comment
Next
```

Pour quitter une boucle for, utilisez l'instruction *Exit For*

```
For i = 0 To 10
    Response.Write("Index: " & i)
    If i=7 Then Exit For 'Exit loop after we write index 7
Next
```

Nous pouvons également utiliser une boucle `For...Each` pour effectuer une boucle à travers une série d'éléments définis dans une collection. Par exemple:

```
Dim farm, animal
farm = New Array("Dog", "Cat", "Horse", "Cow")
Response.Write("Old MacDonald had a Farm, ")
For Each animal In farm
    Response.Write("and on that farm he had a " & animal & ".<br />")
Next
```

Do Loop

Do while est très similaire à for loop mais ceci est généralement utilisé si nos répétitions de boucle sont inconnues.

Faire pendant:

```
'Continues until i is greater than 10
Do While i <= 10
    i = i + 1
Loop
```

```
'Or we can write it so the first loop always executes unconditionally:
'Ends after our first loop as we failed this condition on our previous loop
Do
    i = i + 1
Loop While i <= 10
```

Faire jusqu'à ce que:

```
'Ends once i equates to 10
Do Until i = 10
    i = i + 1
Loop

'Or we can write it so the first loop always executes unconditionally:
'Ends after our first loop as we failed this condition on our previous loop
Do
    i = i + 1
Loop Until i=10
```

Quitter une boucle Do est similaire à une boucle for mais simplement utiliser l'instruction *Exit Do.*

```
'Exits after i equates to 10
Do Until i = 10
    i = i + 1
    If i = 7 Then Exit Do
Loop
```

Lire En boucle en ligne: https://riptutorial.com/fr/asp-classic/topic/5663/en-boucle

Chapitre 4: Les variables

Examples

Déclarer

La création de variables dans VBScript peut être effectuée à l'aide de l'instruction Dim, Public ou Private. Il est recommandé de mettre en haut du script "Option Explicit" qui vous oblige à définir explicitement une variable.

Vous pouvez déclarer une variable comme ceci:

```
Option Explicit
Dim firstName
```

Ou vous pouvez plusieurs variables comme ceci:

```
Option Explicit
Dim firstName, middleName, lastName
```

Si vous n'avez pas l'option instruction explicite, vous pouvez créer des variables comme ceci:

```
firstName="Foo"
```

Ceci n'est **PAS** recommandé car des résultats étranges peuvent se produire pendant la phase d'exécution de votre script. Cela se produit si une faute de frappe est faite ultérieurement lors de la réutilisation de la variable.

Pour créer un tableau, déclarez-le simplement avec le nombre d'éléments dans le paramètre:

```
Option Explicit
Dim nameList(2)
```

Cela crée un tableau avec trois éléments

Pour définir des éléments de tableau, utilisez simplement la variable avec l'index comme paramètre comme suit:

```
nameList(0) = "John"
```

VBScript prend également en charge les tableaux multidimensionnels:

```
Option Explicit
Dim gridFactors(2, 4)
```

Types de variables

VBScript est un langage faiblement typé. les variables sont toutes de type variant , mais elles ont généralement un sous- type implicite indiquant les données qu'elles contiennent.

Cela signifie que votre variable, peu importe comment vous l'appelez, peut contenir n'importe quelle valeur:

```
Dim foo
foo = "Hello, World!"
foo = 123.45
foo = #01-Jan-2016 01:00:00#
foo = True
```

Notez que ce qui précède est un code parfaitement valide, bien que mélanger vos variables comme cela est une pratique étonnamment médiocre.

Le sous-type de chaîne est toujours attribué en utilisant les marques de parole `" "` . Contrairement à JavaScript et à d'autres langages, l'apostrophe ne fournit pas les mêmes fonctionnalités.

Les nombres dans VBScript peuvent inclure n'importe quel format de nombre, mais ont un sous-type particulier basé sur leur valeur et s'ils contiennent ou non un point décimal.

Les dates utilisent les spécificateurs `# #` . Sachez que les formats pour un style de date numérique (par exemple, 01/01/2016) conservent un format de date américain, donc `#05/06/2016#` est le 6 mai et non le 5 juin. Cela peut être vérifié en utilisant un format `#dd-mmm-yyyy#` , comme dans l'exemple ci-dessus.

Les variables booléennes contiennent des valeurs `True` ou `False` .

Comme expliqué précédemment, les tableaux sont dimensionnés à l'aide d'un ensemble de parenthèses pour définir le nombre d'éléments et de rangs (lignes et colonnes), par exemple:

```
Dim myArray(3, 4)
```

Tous les éléments des tableaux sont de type variant, ce qui permet à chaque élément d'être de n'importe quel sous-type. Ceci est très important lorsque vous devez effectuer des tâches telles que la lecture de données à partir d'un jeu d'enregistrements ou d'un autre objet. Dans ces cas, les données peuvent être directement affectées à une variable, par exemple, lorsqu'elles sont renvoyées d'un jeu d'enregistrements ...

```
Dim myData
....
myData = rsMyRecordset.GetRows()
....
Response.Write(myData(3,2))
```

Un type final nécessitant des explications est le type d' `Object` . Les objets sont essentiellement des pointeurs vers l'emplacement mémoire de l'objet lui-même. Les types d'objet doivent être `Set` ...

```
Dim myObj
```

```
Set myObj = Server.CreateObject("ADODB.ecordSet")
```

Lire Les variables en ligne: https://riptutorial.com/fr/asp-classic/topic/3195/les-variables

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec asp-classic	Community, David Starkey, Lankymart, Martha, RamenChef, SearchAndResQ
2	Connexion à une base de données	Lankymart, Martha
3	En boucle	Jake, Paul
4	Les variables	feetwet, Jake, John Odom, Lankymart, Paul