Kostenloses eBook

LERNEN asp-classic

Free unaffiliated eBook created from
Stack Overflow contributors.

#asp-classic

Inhaltsverzeichnis

Über

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Kapitel 1: Erste Schritte mit asp-classic

Bemerkungen

Active Server Pages (ASP), auch bekannt als Classic ASP oder ASP Classic, war Microsofts erste serverseitige Skript-Engine für dynamisch generierte Webseiten. Die Einführung von ASP.NET führte zur Verwendung des Begriffs "Classic ASP" für die ursprüngliche Technologie.

Die serverseitige Standard-Skriptsprache für ASP ist VBScript. Die generierten Seiten sollen in einem Browser angezeigt werden, so dass sie normalerweise HTML-Markup und CSS-Stil verwenden.

[1] ASP wird auf diesen Versionen von IIS nicht standardmäßig installiert. Sie müssen die Server-Manager-Funktionen aufrufen und ASP hinzufügen.
Siehe Klassisches ASP, das standardmäßig nicht unter IIS 7.0 und höher installiert ist

Versionen

IIS	ASP	Veröffentlicht
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

Hallo Welt

```
<!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>
```

Wenn eine Antwort vom Server an den Browser gesendet wird, sieht die Ausgabe folgendermaßen aus:

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

Struktur einer einfachen ASP-Seite

```
<%@ 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>
```

Dies ist ein sehr einfaches Beispiel für eine klassische ASP-Seite, die den Ausdruck "Hello World" zusammen mit dem restlichen Standard-HTML-Code an den Browser zurückgibt. Die HTML-Teile sind statisch, dh der Server sendet sie unverändert an den Browser. Die durch `<% %>` begrenzten Teile werden vom Server tatsächlich verarbeitet, bevor er an den Client gesendet wird.

Beachten Sie, dass die `<%="stuff"%>` -Syntax eine Abkürzung für `<%Response.Write "stuff"%>` .

Erste Schritte mit asp-classic online lesen: https://riptutorial.com/de/asp-classic/topic/1094/erste-schritte-mit-asp-classic

Kapitel 2: Looping

Examples

Für Schleife

Im klassischen ASP können wir eine for-Schleife mit dem *for*- Schlüsselwort angeben. Mit der for-Anweisung benötigen wir die *nächste* Anweisung, die den Zähler erhöht.

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

Mit dem Schlüsselwort *step* kann geändert werden, wie die *nächste* Anweisung den Zähler ändert.

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

Um eine for-Schleife zu beenden, verwenden Sie die *Exit For*- Anweisung

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

Wir können auch eine `For...Each` Schleife verwenden, um eine Schleife durch eine Reihe definierter Elemente in einer Auflistung durchzuführen. Zum Beispiel:

```
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
```

Machen Sie eine Schleife

Do while ist der for-Schleife sehr ähnlich, wird jedoch im Allgemeinen verwendet, wenn die Wiederholung der Schleife unbekannt ist.

Tun während:

```
'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
```

Machen bis:

```
'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
```

Das Beenden einer Do-Schleife ähnelt einer for-Schleife, verwendet jedoch nur die *Exit Do*-Anweisung.

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

Looping online lesen: https://riptutorial.com/de/asp-classic/topic/5663/looping

Kapitel 3: Variablen

Examples

Deklarieren

Das Erstellen von Variablen in VBScript kann mit der Anweisung Dim, Public oder Private erfolgen. Am besten setzen Sie am Anfang des Skripts "Option Explicit", wodurch Sie gezwungen werden, eine Variable explizit zu definieren.

Sie können eine Variable folgendermaßen deklarieren:

```
Option Explicit
Dim firstName
```

Oder Sie können mehrere Variablen wie folgt:

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

Wenn Sie keine explizite Option haben, können Sie Variablen wie folgt erstellen:

```
firstName="Foo"
```

Dies wird **NICHT** empfohlen, da während der Laufzeitphase Ihres Skripts merkwürdige Ergebnisse auftreten können. Dies geschieht, wenn später bei der Wiederverwendung der Variablen ein Tippfehler gemacht wird.

Um ein Array zu erstellen, deklarieren Sie es einfach mit wie vielen Elementen im Parameter:

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

Dadurch wird ein Array mit drei Elementen erstellt

Um Array-Elemente festzulegen, verwenden Sie einfach die Variable mit dem Index als Parameter:

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

VBScript unterstützt auch mehrdimensionale Arrays:

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

Variablentypen

VBScript ist eine schwach typisierte Sprache. Variablen sind alle vom Typ variant , obwohl sie normalerweise einen implizierten Untertyp haben , der die Daten angibt, die sie enthalten.

Dies bedeutet, dass Ihre Variable, egal wie Sie sie nennen, einen beliebigen Wert enthalten kann:

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

Beachten Sie, dass der obige Code perfekt ist, obwohl das Mischen Ihrer Variablen eine erstaunlich schlechte Praxis ist.

Der String-Untertyp wird immer mit Sprachmarken `" "` zugewiesen. Im Gegensatz zu JavaScript und anderen Sprachen bietet der Apostroph nicht die gleiche Funktionalität.

Zahlen in VBScript können ein beliebiges Zahlenformat enthalten, haben jedoch einen bestimmten Untertyp, basierend auf ihrem Wert und ob sie einen Dezimalpunkt enthalten oder nicht.

Datumsangaben verwenden die `# #` Bezeichner. Beachten Sie, dass Formate für einen numerischen Datumsstil (z. B. 01/01/2016) ein amerikanisches Datumsformat beibehalten. `#05/06/2016#` 05/06/2016 `#05/06/2016#` ist also der 6. Mai und nicht der 5. Juni. Dies kann durch Verwendung eines `#dd-mmm-yyyy#` wie im obigen Beispiel `#dd-mmm-yyyy#` .

Boolesche Variablen enthalten `True` oder `False` Werte.

Wie zuvor erläutert, werden Arrays mit einer Reihe von Klammern dimensioniert, um die Anzahl der Elemente und Ränge (Zeilen und Spalten) zu definieren, zum Beispiel:

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

Alle Elemente in Arrays sind vom Typ variante, sodass jedes einzelne Element von einem beliebigen Subtyp sein kann. Dies ist sehr wichtig, wenn Sie Aufgaben wie das Lesen von Daten aus einem Datensatz oder einem anderen Objekt ausführen müssen. In diesen Fällen können Daten direkt einer Variablen zugewiesen werden, z. B. wenn sie von einem Datensatz zurückgegeben werden.

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

Ein letzter Typ, für den eine Erklärung erforderlich ist, ist der `Object` . Objekte sind im Wesentlichen Zeiger auf den Speicherplatz des Objekts selbst. Objekttypen müssen `Set` ...

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

Variablen online lesen: https://riptutorial.com/de/asp-classic/topic/3195/variablen

Kapitel 4: Verbindung zu einer Datenbank herstellen

Einführung

Classic ASP verwendet eine Technologie, die als ActiveX- Datenobjekte bezeichnet wird, wenn Zugriff auf externe Datenquellen erforderlich ist. Die ADODB-Bibliothek stellt zu diesem Zweck drei Hauptobjekte bereit: `ADODB.Connection` , `ADODB.Command` und `ADODB.Recordset` .

Examples

Dropdown aus der Datenbank ausfüllen

(Einschränkungshilfe: Es gibt viele, viele Programmierer, die absolute Konsequenzen suchen, wenn sie auf Code treffen, der Recordsets anstelle von Befehlen und gespeicherten Prozeduren verwendet.)

```
<%
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>
...
```

Verbindung zu einer Datenbank herstellen online lesen: https://riptutorial.com/de/asp-classic/topic/4991/verbindung-zu-einer-datenbank-herstellen

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit asp-classic	Community, David Starkey, Lankymart, Martha, RamenChef, SearchAndResQ
2	Looping	Jake, Paul
3	Variablen	feetwet, Jake, John Odom, Lankymart, Paul
4	Verbindung zu einer Datenbank herstellen	Lankymart, Martha