Dynamisches Web und dynamische Linkliste

Franz Fiala

Im Artikel "Web-Generator" (**PCNEWS**-65, Seite 81) wurde eine Möglichkeit zur Abbildung eines Web in einer Datenbank vorgestellt. Ein VBA-Programm generierte alle Seiten des Web, versah die Seiten mit Layoutinformation, verband die Seiten mit externen Dateien und stellte auch die Links zwischen den Seiten her.

Bitte lesen Sie den ersten Teil dieses Artikel gemeinsam mit dem Artikel aus Ausgabe 65 und verwenden Sie die dort verwendeten Bilder,

Der Vorteil dieses "quasi-dynamischen" Webs ist, dass alle abrufbaren Seiten "auf Vorrat" hergestellt werden und der Webserver keine weitere Arbeit hat, als die fertigen Seiten zum Client zu senden. Das Web ist schnell, auch auf langsamen Servern. Ein weiterer Vorteil ist, dass das Web auch ohne Webserver, z.B. auf CD lauffähig ist, weil keine Skripts seitens des Servers abzuarbeiten sind.

Der Nachteil ist, dass Wartungsarbeiten an der Datenbank sich nicht unmittelbar in den Webseite zeigen. Das Verfahren ist daher nur bei seltenem Aktualisierungsbedarf sinnvoll.

Im folgenden Artikel wird gezeigt, wie wenig Aufwand es bedeutet, aus dem ursprünglich "quasi-dynamischen" Web ein dynamisches Web zu erzeugen.

Dynamisches Web

Es gibt mehrere Aspekte, die eine Webseite als dynamisch – beim Aufrufzeitpunkt auf Kundenwunsch von einem Serverskript hergestellt – erscheinen lassen:

1. Festes Layout, Inhalte veränderlich

1.1 Inhalt auf Kundenwunsch (Beispiel: Telefonbuch, Zugfahrplan)

1.2 Inhalt durch Redaktion bestimmt (Beispiel: ORF, Zeitungsverlage)

2. Variables Layout: User kann Erscheinung und teilweise auch die Inhalte selektieren (Beispiel: Portalsysteme)

Bestandteile einer dynamische Webseite

1. Inhalt

1.1 aus Datenbank

1.2 aus externen Daten (Bilder, Texte)

2. Layoutinformation

2.1 Bildschirmlayout (Frames, Tabellen)

2.2 Stil-Information (Textstile, Absatzstile)

3. Navigationsinformation

3.1 Hierarchische Navigation (inhaltsunabhängig)

3.2 Inhaltsbezogene Navigation

Das folgenden Beispiel zeigt die Punkte 1.1, 1.2, 2.2, 3.1.

Variables Bildschirmlayout (2.1) erzeugt zu große Beispielprogramme. Inhaltsbezogene Navigation (3.2) erfordert eine weitergehende Abstimmung mit der Verzeichnisstruktur (Dateiverschieben verschiebt auch alle Links in allen Dateien) und mehr administrativen Overhead (z.B. bei Frontpage realisiert).

Dynamisch erzeugte Seiten gibt es nicht als HTML-Datei. Sie werden vielmehr durch Aufruf eines serverseitigen Skript (ASP oder PHP) erzeugt. Das Skript erhält im allgemeinen eine eindeutige ID als Parameter übergeben und sucht eine Übereinstimmung dieser Nummer mit einem dafür vorgesehenen Feld in der Datenbank. Beispiel: **http://domain.at/dynamic.asp?ID=12345**. Für eine vollständige Übereinstimmung mit dem Artikel Web-Generator wurde statt des Index **12345** der dortige Seitenname (**seite1a.htm**) beibehalten.

Da dynamische Seiten nur auf Webservern ablaufen können, verwenden Sie zum Testen eine der folgenden Adressen:

http://dynamic.iam.at/generator.asp?seite=seite1a.htm
http;//ueb.iam.at/000/webgen/webgen.asp?seite=seite1a.htm

Hinweise zum Entwickeln serverseitiger Skripts und dynamischer Webs

Eigene Entwicklungswerkzeuge

Ein Entwicklungszyklus für serverseitige Skripts dauert ohne eigene Werkzeuge etwas länger, weil nach jeder Überarbeitung das Skript mit Ftp auf den Server geladen werden muss. Mit InterDev aus dem Visual Studio kann man diese Zeitspanne erheblich verkürzen. Außerdem sorgt das Programm dafür, dass die lokale Version und die Serverversion identisch sind.

Installation eines lokalen Webservers

Eine weitere Hilfe für die Entwicklung ist die Installation eines lokalen Webservers. Sowohl Window-98 als auch Windows 2000 haben einen Webserver beigepackt. Bei Windows-98 findet man ihn auf der CD unter **\AddOns\PWS**, unter Windows 2000 muss man unter Systemsteuerung, Software, Windows-Komponenten hinzufügen/entfernen die Internet-Informationsdienste (ca. 19MB) installieren.

In beiden Fällen wird auf dem aktuellen Rechner ein Webserver installiert, den man unter **http://rechnername** ansprechen kann. Das Wurzelverzeichnis des Servers findet man unter **c:\inetpub\wwwroot**. Dieses Verzeichnis können auch alle Teilnehmer der Workgroup "sehen", falls mit der mitinstallierten Konsole das Web freigegeben ist. Für serverlose Offline-Entwicklung ist dieser Server sehr gut geeignet.

Wann kann man Server-Skripts verwenden?

Ob bei Ihrem Web-Content-Provider (im allgemeinen auch der Access-Provider) serverseitige Skripts verwendet werden können, müssen Sie dort erfragen. Üblicherweise sind in Linux-Systemen Perl/PHP-Scripts und in Windows-NT-Systemen ASP-Scripts möglich. Jeder Server kann aber um zusätzliche Skriptsprachen bereichert werden. Beispielsweise sind am Server des CCC für Clubmitglieder ASP-Skripts erlaubt und es ist auch der Zugang über Frontpage möglich. Beim PCNEWS-Server sind für Unterrichtszwecke Perl, PHP und ASP installiert. Clubmitglieder können auch auf diesem Server experimentieren.

Wie meldet man eine Datenbank an?

Ein Web mit Datenbank-Ankopplung ist üblicherweise ein zahlungspflichtiger Zusatzdienst. Am Server wird die Datenbank am ODBC-System angemeldet (Beschreibung im Beitrag von Christian Zahler). Für die Entwicklungsphase am lokalen Webserver erfolgt die Anmeldung genau so. Damit zeigt sich ein großer Vorteil des ODBC: Programme laufen trotz geänderter Arbeitsumgebung mit demselben Kode. Mit gleichzeitig laufendem lokalen Webserver kann man das dynamische Web vollständig lokal testen. Für Übungszwecke gibt es einige Demoverzeichnisse mit angemeldeten Datenbanken auf einem PCNEWS-Übungsserver. Bei Bedarf bitte anfragen.

Dynamisches Web aus nur einer Datei

Ein dynamisches Web erzeugt eine praktisch beliebig große Anzahl zusammenhängender Webseiten aus nur einer ASP-Datei. Diese Datei erhält über die Kommandozeile ein Argument übermittelt, das die betreffende Seite kennzeichnet. Beispiel:

http://dynamic.iam.at/webgen.asp?seite=seite1a.htm

Beachten Sie bei der Besprechung des folgenden Programmbeispiels die große Symmetrie zum Programm im Beispiel Web-Generator. Die meisten Unterschiede kommen von den Sprachunterschieden zwischen VBA und VBScript und von dem unterschiedlichen Ziel der Ausgabe. Beim VBA-Programm wurde eine Datei angelegt (`Print #1,...`), hier wird der String zum Client geschickt (`Response.Write...`, oder `%>...<%`).

Die verwendete Tabellen **WEB** und **STIL** wurden im Artikel Web-Generator besprochen und können dort nachgelesen werden. Die Tabelle **WEB** ist für die Anzahl der Seiten verantwortlich. In diesem Beispiel gibt es nur 3 Records, daher auch nur 3 Webseiten.

webgen.asp

```
<% Language = "VBScript"
```

Zunächst muss das ASP-Skript das Argument abfragen. Kommt kein oder ein falsches Argument, wie z.B. in **http://dynamic.iam.at/webgen.asp** gibt es eine Fehlermeldung.

```
Seite = Request.QueryString("Seite")
```

Die Datenbank wird geöffnet. Der Anmeldename `000` wurde im ODBC-System eingetragen, daher müssen wir bei diesem Skript nicht mehr wissen, wo die Datei eigentlich gespeichert ist.

```
set ConnObj = Server.CreateObject("ADODB.Connection")
  ConnObj.Mode = adModeRead
  ConnObj.Open "000"
```

Alle Abfragen erfolgen als SQL-Statement, auch wenn in diesem Beispiel eine Access-Datenbank abgefragt wird.

```
SQLAbfrage =
"SELECT WEB.*, STIL.* " &
"FROM STIL RIGHT JOIN WEB ON STIL.N STIL = WEB.N STIL " &
"WHERE (((WEB.N_DST)='"+Seite+"'));"
```

Wenn keine erfolgreiche Abfrage in die Datenbank möglich ist, gibt es eine Fehlermeldung:

```
Set DS = ConnObj.Execute(SQLAbfrage)
If DS.BOF Then
  Response.Write("keine Datei")
  Response.End
End If
```

Die Aufgabe des Skript ist die Erzeugung einer HTML-Datei, die an den Client zurückgeschickt wird, danach werden die Abfrage und die Datenbank wieder geschlossen.

```
GeneriereHTMLDatei DS
DS.Close
ConnObj.Close
```

In diesem Beispiel wird bei jeder Anfrage die Datenbank geöffnet. In einem realen Web wird man das aber aus Performancegründen nur einmal tun, wenn das Web gestartet wird. Dieser Kode kommt in die dafür reservierte Datei **global.asa**. Die dort definierten Variablen (hier wäre das `ConnObj`) steht dann jedem Abfragevorgang bereits zur Verfügung.

An der folgenden Prozedur `GeneriereHTMLDatei` kann man die automatische Generierung der feststehenden Tags (z.B. `HTML`, `TITLE` usw.) erkennten. Man benötigt dazu keine besonderen Anweisungen, denn Text außerhalb der ASP-Klammern `<% %>` wird an den Client zurückgeschickt. Wenn diese Handhabung unzweckmäßig sein sollte, kann man den Text an den Client auch mit `Response.Write` zurückschicken.

```
Public Sub GeneriereHTMLDatei(DS)
  Dim n
  Dim DateiQuelle
  'Wir müssen bedenken, dass die DateiQuelle auch leer sein kann
  If IsNull(DS("F SRC")) Then
    DateiQuelle = ""
  Else
    DateiQuelle = DS("F SRC")
  End If

%>
  <HTML>
  <TITLE><%= DS("H_TITEL")%></TITLE>
  <BODY
<%
```

Die Farbinformation für das **BODY**-Tag stammen aus der Datenbank.

```
  If Not IsNull(DS("R TEXT")) Then
      Response.Write(" TEXT=#" + DS("R_TEXT"))
  End If
  If Not IsNull(DS("R_BACK")) Then
      Response.Write(" BGCOLOR=#" + DS("R BACK"))
  End If
  If Not IsNull(DS("R LINK")) Then
      Response.Write(" LINK=#" + DS("R_LINK"))
  End If
  Response.Write(">")
```

Der Aufbau einer Seite erfolgt so, dass ein Textteil aus der Datenbank kommt (**H_SRC**) und ein anderer Teil aus einer eventuell vorhandenen externen Datei (**T_SRC**):

```
  'Datenbankfeld H SRC inkludieren
  Response.Write("<P><I><B>Hier wird der Text aus dem " & _
   Datenbankfeld "  H_SRC inkludiert (falls vorhanden):</B></I><BR>")
  If IsNull(DS("H SRC")) Then
    Response.Write("kein Text")
  Else
    Response.Write(DS("H_SRC"))
  End If

  'Datei H SRC inkludieren
  Response.Write("<P><I><B>Hier wird der Text aus " & _
     der Datei T SRC inkludiert (falls vorhanden):</B></I><BR>")
  Response.Write("</P>")

  If DateiQuelle = "" Then
    Response.Write("keine Datei")
  Else
    InkludiereDatei DateiQuelle
  End If
  Response.Write("</P>")
```

Die Links zu den anderen, ebenfalls dynamisch generierten Seiten ergeben sich aus weiteren Datenbankfeldern.

```
  'Navigation
  Navigation DS
  %>
  <P>Bearbeitet am <%=FormatDateTime(DS("D DATUM"))%> <BR>
  Aktualisiert am  <%= FormatDateTime(Now())%></P>
  </BODY>
  </HTML>
  <%
End Sub
```

Die Inklusion einer externen Datei (falls vorhanden).

```
Public Sub InkludiereDatei(DateiQuelle)
  Response.Write(DateiQuelle)
  Const ForReading = 1, ForWriting = 2, ForAppending = 3
  Set fs = CreateObject("Scripting.FileSystemObject")
  Set f = fs.GetFile("D:\~ueb\000\webgen\"+DateiQuelle)
  Set ts = f.OpenAsTextStream(ForReading, TristateUseDefault)
  Do While Not ts.AtEndOfStream
    s = ts.ReadLine
    Response.Write (s)
  Loop
  ts.Close
End Sub
```

Die Navigation besteht aus 4 Zeilen, die zu den benachbarten Dokumenten und Kapitel verweisen.

```
Public Sub Navigation(DS)
  Response.Write ("<CENTER><NOBR>")
  Hyperlink "Voriges Kapitel", DS("L KAPPREV")
  Hyperlink "Voriges Dokument", DS("L DOKPREV")
  Hyperlink "Nächstes Dokument", DS("L_DOKNEXT")
  Hyperlink "Nächstes Kapitel", DS("L KAPNEXT")
  Hyperlink "Webmaster", "mailto:my@name"
  Response.Write ("</NOBR></CENTER>")
End Sub
```

Da die Formatierung eines Link eine mühsame Sache zwischen spitzen Klammern ist, wurde sie einer eigenen Prozedur übertragen.

```
Public Sub Hyperlink(TXT, LINK)
  If LINK <> "" Then
    Response.Write ("<FONT SIZE=1><A HREF= " &
      http://dynamic.iam.at/generator.asp?Seite="+ LINK+ ">")
    If Not IsNull(TXT) Then
      Response.Write("[" + TXT + "]")
    End If
    Response.Write("</FONT></A><BR>")
  Else 'Es gibt keinen Link
    If Not IsNull(TXT) Then
      Response.Write("<FONT SIZE=1 " &
        COLOR=#555555>[" + TXT + "]</FONT><BR>")
    End If
  End If
End Sub
```

Es darf nicht vergessen werden, das der ASP-Abschnitt mit `%>` korrekt abzuschließen ist.

```
%>
```

Demonstration

Diese Beispiel können Sie unter

http://ueb.iam.at/000/webgen/webgen.asp?seite=seite1a.htm

ausprobieren, die Dateien zum Download finden Sie unter

http://ueb.iam.at/000/webgen.zip

Die Datenbank unter

http://ueb.iam.at/000/mydb.mdb

Dynamische Pull-Downmenüs mit ASP und PHP

In dynamischen Webs kann praktisch jedes Element programmgesteuert gestaltet werden. Zum Beispiel können die Inhalte des Informationsangebots in Form von Aufzählungen (Linklisten) präsentiert werden. Diese haben den Vorteil, dass man alle Inhalte gleichzeitig ("auf einen Blick") erfassen kann aber den Nachteil, dass sie je nach Umfang verschieden große Bildschirmbereiche umfassen. Ein Ausweg sind Pull-Down-Menüs, deren Größe man beschränken kann (z.B. auf ein Element (=Überschrift) oder auf die ersten, wichtigsten Einträge). Die Pull-Down-Menüs beanspruchen immer denselben Platz am Bildschirm.

Die Fragestellung ist zweistufig:

1. Wie programmiert man Pull-Down-Menüs, sodass die Auswahl einer bestimmten Zeile auch gleichzeitig zu dem gewünschten Inhalt verzweigt und
2. Wie verbindet man mit einer Datenbank damit genau dieser Kode entsteht (in ASP oder PHP)

Pull-Down-Menüs als Linkliste

Grundsätzlich wird ein Pull-Down-Menü im HTML-Kode mit einem `SELECT`-Tag, gefolgt von `OPTION`-Tags erzeugt:

```
<FORM NAME=Auswahl>
<SELECT SIZE=1 NAME=Auswahl1 onChange="Auswahl onChange();">
  <OPTION VALUE=0>Produktliste</OPTION>
  <OPTION VALUE=1>Auswahl 1</OPTION>
  <OPTION VALUE=2>Auswahl 2</OPTION>
  <OPTION VALUE=3>Auswahl 3</OPTION>
</SELECT>
</FORM>
```

In diesem Beispiel ist das Pull-Down-Menü einzeilig (`SIZE=1`). Das Formular Auswahl hat kein `METHOD`-Tag und auch kein `ACTION`-Tag, weil der Formular-Inhalt nicht zum Server gesendet wird. Eine Auswahl soll bewirken, dass eine neue Seite geladen wird. Da man die ursprüngliche Funktion des Pull-Down-Menüs verändert, muss man eine JavaScript-Funktion (`Auswahl_onChange()`) definieren, die bei Veränderungen der Auswahl aufgerufen wird. Würde man das `SELECT`-Feld ohne die `onChange`-Funktion bilden, sieht man zwar die Box am Bildschirm, wenn man aber etwas auswählt, passiert nichts. Daher gibt es den `onChange`-Event-Handler, der in der JavaScript-Funktion `Auswahl_onChange()` etwas ausführt. Die JavaScript-Funktion wird immer dann angesprungen, wenn der User eine neue Auswahl trifft. Dort wird dann ein neues Fenster mit dem gewünschten Ziel eröffnet. Der Kode dafür ist etwa so:

```
<SCRIPT>
funktion Auswahl onChange()
{
  if (document.Auswahl.Auswahl1.options[0].selected)
    ;
  if (document.Auswahl.Auswahl1.options[1].selected)
    open ("/seite1.htm");
  if (document.Auswahl.Auswahl1.options[2].selected)
    open ("/seite2.htm");
  if (document.Auswahl.Auswahl1.options[3].selected)
    open ("/seite3.htm");
  document.Auswahl.Auswahl1.selectedIndex=0;
}
</SCRIPT>
```

Dieser Kode prüft in aufeinanderfolgenden if-Entscheidungen, ob eine bestimmte `OPTION` im `SELECT`-Tag ausgewählt wurde. Die Optionen sind im Array `options[]` zu finden. Die Eigenschaft `selected` ist für die zutreffende Option `true` oder `false`. Die Schreibweise `document.Auswahl.Auswahl1.options[0].selected` zeigt die Objekthierarchie. `selected` ist eine Eigenschaft des `options`-Array, dieses eine Eigenschaft des `SELECT`-Tag `Auswahl1`, dieses ein Objekt des Formulars `Auswahl`, das wieder Bestandteil des Dokuments `document` des aktuellen Fensters ist.

Wenn also eine der Optionen gewählt wurde, wird eine von drei möglichen Folgeseiten **`seite1.htm`**, **`seite2.htm`** oder **`seite3.htm`** in einem neuen Fenster geöffnet. (Alternativ könnte man diese Dokumente in einem Teilfenster eines Frameset öffnen, was aber aus Gründen der Übersichtlichkeit hier nicht gezeigt wird.)

Schließlich soll sich das Pull-Down-Menü wieder in seiner ursprünglichen Form zeigen, daher wird mit `document.Auswahl.Auswahl1.selectedIndex=0;` wieder die erste Zeile eingeblendet. Diese erste Zeile bewirkt nichts, die zugehörige `if`-Entscheidung ist leer, weil in dieser Zeile der Titel steht.

Das war der statische Kode, wie ihn der User in seinem Browser "sieht". In Wirklichkeit sind aber die Spungziele im Pull-Down-Menü sowohl in der Anzahl als auch in den Zielen variabel und abhängig von der Anzahl der Records in einer Datenbank. Das ASP/PHP-Skript muss jetzt diesen Text in Abhängigkeit von der Anzahl der Einträge dynamisch aus der Datenbank generieren.

Datenbank als Input

Eine Datenbank, die am ODBC-System unter dem Namen **`000`** angemeldet wurde, enthält eine Tabelle **`X_PRODUKTE`**, die für jedes Produkt eine eindeutige Nummer **`IDPRODUKT`** und einen Namen enthält.

Das Feld **`IDPRODUKT`** ist eine eindeutige Nummer und dient als Name für eine HTML-Datei mit der Produktbeschreibung auf die der Verweis erfolgt.

Auf der folgenden Seite ist derselbe Kode in ASP und in PHP dargestellt. Dabei wurde in PHP ebenso wie in ASP die Datenbank über ODBC angesprochen.

Referenzen zu ODBC in PHP

ODBC-Teil im PHP-Manual
http://www.php.net/manual/ref.odbc.php

Funktionensammlung ADODB8
http://php.weblogs.com/

Manual zu ADODB8
http://php.weblogs.com/ADODB_manual/

Alle Bibliotheksfunktionen von ADODB8
http://ueb.iam.at/adodb/

Archiv ADODB8
http://ueb.iam.at/adodb/adodb080.zip

Vorteile von ODBC

Der Vorteil der ODBC-Datenbank-Programmierung ist, dass der entstehende Kode von der verwendeten Datenbank praktisch unabhäng ist. In ASP erfolgt die Zuweisung des ODBC-Anmeldenamens in der Systemsteuerung. In PHP entscheidet eine Zeile über die Art der Datenbank. Ersetzt man

```
ADOLoadCode('access');              // load Access code
```

durch

```
ADOLoadCode('mysql');               // load MySQL code
```

hat man auch gleichzeitig die Übungsdatenbank durch eine SQL-Datenbank ersetzt.

Eine beachtenswerte Besonderheit diese Beispiels ist, dass nicht nur der reine HTML-Kode von den Scripts generiert wird, sondern auch der JavaScript-Kode, dessen Länge ebenfalls von der Datenbank-Struktur abhängig ist.

Auf der folgenden Seite wird die Symmetrie der beiden Kodevarianten **`produkte.asp`** und **`produkte.php`** gezeigt. Die Unterschiede können leicht durch zeilenweisen Vergleich studiert werden.

Das im generierten HTML-Kode gerufene Script **`produkt.asp`** bzw. **`produkt.php`** wurde hier nicht dargestellt; Sie finden es in der archivierten Webversion unter

http://ueb.iam.at/000/pulldown.zip

Weitere Beispiele

finden Sie unter http://ueb.iam.at/. Kurze Beispiele, die nur aus einer Datei bestehen sind in diesem Verzeichnis zu finden, Beispiele mit mehreren Dateien und eigenen Verzeichnissen. Damit auch Zugang zum Source-Kode besteht, wurden alle Beispiele auch in gezippter Form abgelegt.

Wenn Sie auf diesem Server experimentieren wollen, können Sie ein eigenes Verzeichnis bekommen. Schreiben Sie an pcnews@pcnews.at.

Pulldownmenü mit ASP

http://ueb.iam.at/000/pulldown/produkte.asp

```
<%@ Language=VBScript %>
<% Option Explicit %>
<!-- #include virtual="/adovbs.inc"-->
<%
Dim Datenbank 'Variable für das Datenbankobjekt
Dim Produkte 'Variable für die Abfrage
Dim SQLquery 'Variable für den Abfragestring

Set Datenbank = Server.CreateObject("ADODB.Connection")
  Datenbank.Mode = adModeReadWrite
  Datenbank.Open "000"
SQLquery = _
  "SELECT X_PRODUKT.*, X_BILD.BILD, X_AUTOR.AUTOR " &
  "FROM X_BILD INNER JOIN (X_AUTOR INNER JOIN X_PRODUKT ON
X_AUTOR.IDAUTOR = X_PRODUKT.IDAUTOR) ON X_BILD.IDBILD =
X_PRODUKT.IDBILD;"
Set Produkte = Server.CreateObject("ADODB.Recordset")
Produkte.CursorType = adOpenDynamic
Produkte.LockType = adLockOptimistic
Produkte.Open
SQLquery,Datenbank,adOpenDynamic,adLockOptimistic,adCmdText

If Produkte.BOF Then
  Response.Write "Es gibt kein Produkt"
  Response.End
End If
%>

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function Auswahl_onChange()
{
  if (document.Auswahl.Auswahl1.options[0].selected) { }
<%
Dim Zaehler
Zaehler=1
Do While Not Produkte.EOF
  %>
  if (document.Auswahl.Auswahl1.options[<%=Zaehler%>].selected) {
    open ('produkt.asp?IDPROD=<%=Produkte("IDPROD")%>');
    document.Auswahl.Auswahl1.selectedIndex=0;}
  <%
  Produkte.MoveNext
  Zaehler=Zaehler+1
Loop
%>
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME=Auswahl>
<SELECT SIZE=1 NAME=Auswahl1 onChange="Auswahl onChange();">
  <OPTION VALUE=0>Produktliste</OPTION>
<%
Produkte.MoveFirst 'Beginn wieder beim ersten Produkt


Zaehler=1
Do While Not Produkte.EOF
  %>
  <OPTION VALUE=<%=Zaehler%>><%=Produkte("NAME")%></OPTION>
  <%
  Produkte.MoveNext
  Zaehler=Zaehler+1
Loop
%>
</SELECT>
</FORM>
<%
Produkte.MoveLast 'Gehe zum letzten Produkt
%>
<SCRIPT>

open ('produkt.asp?IDPROD=<%=Produkte("IDPROD")%>');

</SCRIPT>
</BODY>
</HTML>
<%
Produkte.Close
Datenbank.Close
%>
```

Pulldownmenü mit PHP

http://ueb.iam.at/000/pulldown/produkte.php

```
<?

include('d:\~ueb\adodb\adodb.inc.php');  //ADODB-Kode
ADOLoadCode('access');             // load Access code



$Datenbank = &ADONewConnection();    // create a connection
$Datenbank->PConnect('000');

$SQLquery =
  'SELECT X_PRODUKT.*, X_BILD.BILD, X_AUTOR.AUTOR ' .
  'FROM X_BILD INNER JOIN (X_AUTOR INNER JOIN X_PRODUKT ON
X_AUTOR.IDAUTOR = X_PRODUKT.IDAUTOR) ON X_BILD.IDBILD =
X_PRODUKT.IDBILD;';
$Produkte = &$Datenbank->Execute($SQLquery);



if ($Produkte->EOF) {
  ?>
  <P>kein Produkt</P>
  <?
}
?>
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function Auswahl onChange()
{
  if (document.Auswahl.Auswahl1.options[0].selected) { }
<?

$Zaehler=1;
while (!$Produkte->EOF)
{
  echo "if (document.Auswahl.Auswahl1.options[".$Zaehler."].selected)
{ ";
  echo "open ('produkt.asp?IDPROD=".$Produkte->fields[0]."');";
  echo "document.Auswahl.Auswahl1.selectedIndex=0;}";
  $Produkte->MoveNext();
  $Zaehler=$Zaehler+1;
}
?>
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME=Auswahl>
<SELECT SIZE=1 NAME=Auswahl1 onChange="Auswahl_onChange();">
  <OPTION VALUE=0>Produktliste</OPTION>
<?
$Produkte->MoveFirst(); // Beginn wieder beim ersten Produkt
(funktioniert leider nicht, daher:
$Produkte = &$Datenbank->Execute($SQLquery);
$Zaehler=1;
while (!$Produkte->EOF)
{
  echo "<OPTION
VALUE=".$Zaehler.">".$Produkte->fields[1]."</OPTION>";
  $Produkte->MoveNext();
  $Zaehler=$Zaehler+1;
}
?>
</SELECT>
</FORM>
<?
$Produkte->MoveLast(); //Gehe zum letzten Produkt
?>
<SCRIPT>
<?
echo "open ('produkt.asp?IDPROD=".$Produkte->fields[0]."');";
?>
</SCRIPT>
</BODY>
</HTML>
<?
$Produkte->Close();
$Datenbank->Close();
?>
```