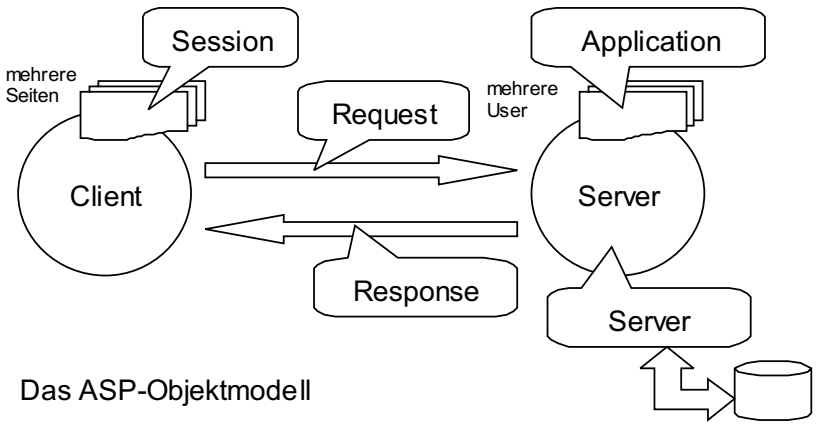
Webdatenbank

Franz Fiala



Das ASP-Objektmodell

Wenn wir im Internet tolle Sites bewundern, sehen wir in unserem Browser nur das Endprodukt eines komplexen Generierungsvorgangs, der auch durch einen Blick in den Quellkode nicht sichtbar wird, bestenfalls erahnbar, wenn nämlich der sichtbare Kode so ganz und gar nicht der sonst üblichen Ästhetik von Programmen entspricht und ein bisschen chaotisch daher kommt. Dieses für Programmierer eigentlich untypische Durcheinander kommt daher, dass der sichtbare Kode von einem anderen Programm (im allgemeinen einem ServerScript) erzeugt wird und dieses Programm seine eigene Logik hat. Vor allem erzeugt es für verschiedene Anfragen auch verschiedenen Kode.

Websites sind dann attraktiv, wenn sie dem Zielpublikum nützen und aktuell sind, wenn also alle Aussagen davon zeugen, dass sie vom Jetzt oder vom Morgen, nicht aber auf Unaktuelles verweisen.

Natürlich kann man Teams von Web-Editoren die Inhalte händisch bearbeiten und aktualisieren lassen, doch erschließen diese Editierarbeiten neue Fehlerkategorien, d.h. die Seiten sind - mit großem Aufwand - aktuell, aber oft unsystematisch dargestellt, weil eben von verschiedenen Personen zu verschieden Zeitpunkten hergestellt.

Diese Probleme werden vermieden, wenn die Inhalte der Webseiten in Datenbanken abgelegt werden und serverseitige Programme den HTML-Rahmenkode mit den Inhalten verbinden und als reines HTML zum Browser schicken; selbstverständlich für jede Browserversion den entsprechend zutreffenden Kode.

Die neuen Probleme, die man sich mit diesem Konzept einkauft, versucht der vorliegende Artikel zu beschreiben.

Wer kann diese Technik nutzen?

Den ersten Teile dieses Artikels, das Auslesen von Tabellen in Datenbanken, kann jeder benutzen, weil keine besonderen Schreibrechte notwendig sind. Da hier ASP, eine Microsoft-Technologie beschrieben wird, benötigt man dazu einen Windows-Server. (Die Server des CCC/PCC sind Windows-Server.)

Wenn, wie im zweiten Teil des Artikels gezeigt wird, die Tabelle auch über ein Webinterface editierbar sein soll, muss die Datenbank mit Schreibrechten versehen sein; entweder für den jeweils berechtigten Redakteur oder, wie es bei Gästebüchern der Fall ist, für den anonymen Internetuser.

ASP steuert den Server

Ein Web-Server hat die relativ einfache Aufgabe, Dateien mit der Endung **HTM** (oder **HTML**), **GIF**, **JPG**... zum Client zu schicken. Details dieses Transfers werden im HTTP-Protokoll abgehandelt. Der Benutzer bemerkt die Dienste dieses Protokolls nur am Rande; etwa über die Adresszeile oder über fallweise Fehlermeldungen. Man kann aber am Server genau einstellen, was mit anderen Dateiendungen (**shtm**, **php**, **asp**..) zu geschehen hat. Im allgemeinen durchsucht der Server diese Dateien nach Steuerinformation und handelt danach. Die Dateiendung **asp** (*Active Server Pages*) veranlasst den Server, in der Datei nach den Zeichen `<%..%>` zu suchen und den Inhalt in einer anfangs definierten Sprache (*VBScript*=default oder *JScript*) zu interpretieren.

Über diesen Mechanismus hinaus kann man aber dieses Verhalten am Server ändern. Wenn man zum Beispiel ein bestehendes Web, das ausschließlich aus HTML-Dateien besteht, so verändern will, dass zum Beispiel am Ende einer jeden Datei das aktuelle Datum auszugeben wäre (was nur mit Mitteln einer serverseitigen Skriptsprache funktioniert), dann kann man Server das Verhalten bezüglich der HTML-Dateien verändern und anordnen, dass auch diese von dem ASP-Mechanismus betroffen sind.

Was ist ASP?

ASP ist der besagte Umschaltemechanismus und die Integration einiger Objekte in die jeweilige Sprache: `Server`, `Response`, `Request`, `Session`, `Application`. Diese Objekte müssen nicht extra geladen werden, sie sind immer geöffnet.

Server

Das Objekt `Server` hat die wichtige Aufgabe, zusätzliche externe Objekte von der Platte zu laden; außerdem kann es die URL-Kodierung und die HTML-Kodierung beliebiger Texte vornehmen.

Response

Das Objekt `Response` ist für alle Aktivitäten zuständig, die zum Client gerichtet sind, z.B. das Senden eines Textes zum Client mit der Methode `Response.Write` oder das Setzen eines Cookie mit `Response.Cookies`.

Request

Das Objekt `Request` behandelt alle Informationen, die von Client zum Server kommen: HTTP-Header (CGI-Schnittstelle), Formulardaten, Daten der Kommandozeile, Cookies

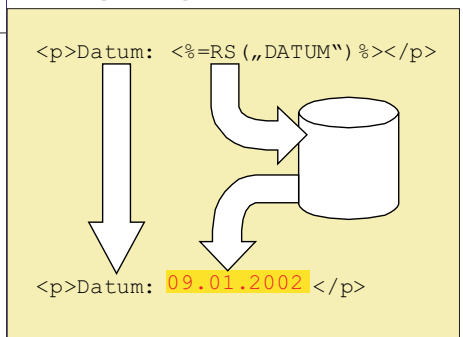
Session

Das Objekt Session ist ein Adressraum für Variable, die während des gesamten Zeitraums eines Besuchs einer Website durch einen User gelten. (Normalerweise leben Variablen aus einer asp-Seite nur bis zum Ende des Dokuments. Das `Session`-Objekt ermöglicht etwa das Anlegen von Warenkörben.

Application

Das Objekt `Application` ist ein Adressraum für Variable, die allen Usern einer Website gemeinsam zur Verfügung steht. Damit können beispielsweise zwei gleichzeitig eingeloggte User kommunizieren.

Grundprinzip



Server ersetzt die ASP-Formel durch das Datumsfeld aus der Datenbank.

Das Grundprinzip der serverseitigen Datenbakprogrammierung zeigt die folgende Skizze am Beispiel eines Datumsfeldes. Alle feststehenden Kodeteile enthalten reine HTML-Anweisungen (`<P>Datum:`); variable Bestandteile, hier das Datum, müssen aus der Datenbank geholt werden. Dazu benötigt man einerseits einen Umschaltemechanismus, der dem Server sagt, dass an dieser Stelle Handlungsbedarf besteht (das ist hier die Zeichenfolge `<%=...%>`) und zusätzlich einen Ansprachemechanismus für eine bestimmte Spalte einer Tabelle, hier die Spalte `DATUM`. Wie in allen modernen Sprachen verpackt auch das hier verwendete Visual-Basic-Script diese Ansprache in ein Objekt, das sogenannte `Recordset`-Objekt, das sich durch den Objektnamen `RS` darstellt. Das Argument `"DATUM"` wählt genau den Inhalt der Spalte `Datum` aus.

```
<%@ Language=VBScript %>
<% Option Explicit %>
<%

'Umgebung erforschen?
Dim ScriptName
ScriptName = Request.ServerVariables("SCRIPT_NAME")
'Datenbank im selben Verzeichnis
Const DateinameDatenbank = "redaktion.mdb"
Dim Path
Path = Request.ServerVariables("PATH_TRANSLATED")
Path = Left(Path, InStrRev(Path,"\"))+DateinameDatenbank

'Datenbank
Const adModeRead = 1
Dim Redaktionsdatenbank
Set Redaktionsdatenbank = Server.CreateObject("ADODB.Connection")
Redaktionsdatenbank.Mode=adModeRead

'Datenbank öffnen mit ODBC
'Redaktionsdatenbank.Open "redaktion"

'Datenbank öffnen mit Treiber-Name und Pfad
Redaktionsdatenbank.Open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="+Path

Dim SQLquery
Dim RS

SQLquery = _
     "SELECT REDAKTION.* " & _
     "FROM REDAKTION " & _
     "ORDER BY REDAKTION.DATUM DESC;"
Set RS = Redaktionsdatenbank.Execute(SQLquery)

'Was sieht der Browser?
HTMLHeader "Redaktion", "Redaktion"

RedaktionOutput RS

HTMLFooter

Sub RedaktionOutput
     %><TABLE><%
     Do While Not RS.EOF
          BeitragOutput RS
          RS.MoveNext
     Loop
     %></TABLE><%
End Sub
'---------------------------------------------------------------------
Sub BeitragOutput RS

     %><TR><%

     %><TD><%=FormatDateTime(RS("DATUM"),2)%><BR><%

     If RS("LINK")<>"" Then
          %><A HREF="javascript:neu=window.open('<%=RS("LINK")%>');
window.location.reload()"><B><%=RS("BETREFF")%></A></B><BR><%
     Else
          %><B><%=RS("BETREFF")%></B><BR><%
     End If
     %><%=RS("BEITRAG")%></TD><%
     %></TR><%

End Sub
'---------------------------------------------------------------------
Sub HTMLHeader (Titel, Ueberschrift)
     %>
     <HTML>
     <HEAD>
     <TITLE><%=Titel%></TITLE>
     </HEAD>
     <BODY>
     <H1><%=Ueberschrift%></H1><%
End Sub
'---------------------------------------------------------------------
Sub HTMLFooter
%><P><%=now%></BODY>
</HTML>
<%
End Sub
%>
```

read.asp: Datenbankinhalt lesen

Datenbank lesen (lesen.asp)

Der Inhalt einer Datenbank ist die Grundlage für den Inhalt einer Webseite. Der Ablauf ist prinzipiell folgender:

- Datenbank initialisieren
- HTML-Einleitung
- Alle Datensätze lesen
  - Einen Datensatz ausgeben
- HTML-Schluss

Initialisierung

ASP benutzt zwar als Anfangswert die Sprache VBScript aber üblicherweise gibt man das auch explizit an. Mit `Option Explicit` wir die Definition von Variablennamen erzwungen.

Umgebung erforschen

In vielen Fällen ist es nützlich zu wissen, wie das aktuelle Skript heißt; man erfährt es über die Umgebungsvariable (`SCRIPT_NAME`). Wenn sich die Datenbank im selben Verzeichnis befindet, kann man aus dem Pfad des aktuellen Skripts (`PATH_TRANSLATED`) den Pfad der Datenbank bestimmen, man muss dazu nur den Dateienamen wegschneiden.

Datenbank öffnen

Die Datenbank wird im Lese-Modus geöffnet. Dazu benötigt man das Objekt `ADODB.Connection` nachladen. Je nachdem, ob die Datenbank am ODBC-System angemeldet ist, muss man die Datenbank mit dem ODBC-Namen oder mit Treibername/Pfad öffnen. Der Datenbankname `Redaktionsdatenbank` wurde gewählt, weil dieses und das nachfolgende Programm ein allgemeines Redaktionssystem für eine Webdatenbank beschreiben.

Abfrage öffnen

Die Datenbank wird geöffnet, wobei die Abfrage (`SELECT *`) alle Spalten zurück liefert. Geordnet wird absteigend (`DESC`) nach dem Datum.

Hauptprogramm

Erst jetzt wird der erste Kode an den Browser zurückgeschickt. Da das Schreiben des Kopfteils einer HTML-Datei ein immer wiederkehrender Vorgang ist, wird hier eine parametrierbare Funktion `HTMLheader` verwendet. Der einzige Parameter in diesem Beispiel ist die Titelzeile für den Browser. Entsprechend gibt es eine Funktion `HTMLfooter`.

Abfrageinhalt ausgeben

Die Ausgabe des Datenbankinhalts wird durch die Funktion `RedaktionOutput` übernommen.

`RedaktionOutput`: In diesem Beispiel wird der Funktion das Recordset `RS` als Parameter übergeben. Das hat den Vorteil, dass diese Funktion allgemein verwendbar ist und auch von anderer Stelle aufrufbar wäre. `RedaktionOutput` gibt alle Zeilen in eine Tabelle aus und sorgt für korrekten Anfang und korrektes Ende der Tabelle.

Einen Datensatz ausgeben

Der Hauptteil der Formatierung entfällt auf die Funktion `BeitragOutput`. Das Datum wird in jedem Fall ausgegeben aber die Betreffzeile wird in Abhängigkeit vom Inhalt des Link-Feldes mit oder ohne `A`-Tag ausgegeben. Das Link-Ziel wird in einem eigenen Fenster ausgegeben.

Datenbank editieren (redaktion.asp)

Bei seltenen Änderungen und bei kleinen Datenbanken kann es genügen, die komplette Datenbank vom Server zu holen, zu aktualisieren und dann wieder im Web zu speichern. Dazu eignen sich Access-Datenbanken sehr gut.

Wenn aber die Änderungsrate groß ist oder wenn mehrere Personen gleichzeitig Editieren sollen (Gästebuch) oder die Datenbank zu groß wird, oder sehr viele Zugriffe erfolgen, dann ist ein weiterer großer Programmierschritt erforderlich: Die Datenbank benötigt ein Programm, welches die grundlegenden Datenbankfunktionen **"Hinzufügen"**, **"Ändern"** und **"Löschen"** über geeignete Webinterfaces erschließt.

Zunächst ein wichtiger Ausgangspunkt: *"Alle datenbankgestützte Webprojekte sind ähnlich"*; zumindest, was diese grundlegenden Funktionen **"Hinzufügen"**, **"Ändern"** und **"Löschen"** betrifft. Es lohnt sich daher, die Organisation des Editierens gründlich zu erarbeiten und dann diese Vorgangsweise auch auf vergleichbare Anwendungsfälle zu übertragen.

Wie diese grundlegenden Funktionen ablaufen, zeigt das Bild unten. Es handelt sich nicht um ein einzelnes Programm, sondern um insgesamt 7 Programme, die gemeinsam die Datenbank editieren. Ein 8tes Programm ist dann die bereits beschriebene Funktion `read.asp`, die schon im vorigen Beispiel gezeigt wurde. Der Grundzustand wird durch den dicken Kreis links und rechts im Bild dargestellt. In diesen Grundzustand gelangt man durch die Adresszeile `http://.../redaktion.asp?who=redakteur`

Der Pfad **"Hinzufügen"** beginnt mit dem Zustand `Add`, in dem am Browser eine Eingabemaske dargestellt wird. Die einzelnen Feldnamen entsprechen den jeweiligen Namen in der Tabelle der Datenbank. Es gehört auch zum Zustand `Add`, dass alle Eingabefehler, zum Beispiel leere Felder oder unzulässige Eingaben, durch entsprechende JavaScript-Funktionen erkennt. Ist die Eingabe korrekt, gelangt man in den Zustand `AddeExec`, der alle eingegebenen Daten in die Datenbank zu schreiben hat. Es muss keine Ausgabe zum Client erfolgen, außer wenn Fehler auftreten oder vielleicht eine einfache Quittung *"Datensatz hinzugefügt"*.

Der Pfad **"Löschen"** beginnt mit einer Ausgabe aller Datensätze aber mit einem entsprechenden Editierelement, das eine Auswahl erlaubt. Es müssen nur jene Felder ausgegeben werden, die zu einer eindeutigen Identifikation des Datensatzes erforderlich sind. Jedes Auswahl-Element (hier ein Radio-Button) bekommt als Namen die `ID` des betreffenden Datensatzes. Eine eventuelle Rückfrage, ob der Datensatz tatsächlich gelöscht werden soll, kann man durch eine JavaScript-Funktion an dieser Stelle einfügen. Das durch den Button "Löschen" ausgelöste Programm `DelExec` löscht den Datensatz ohne Rückfrage, eine einfach Quittung *"Datensatz gelöscht"* ist möglich.

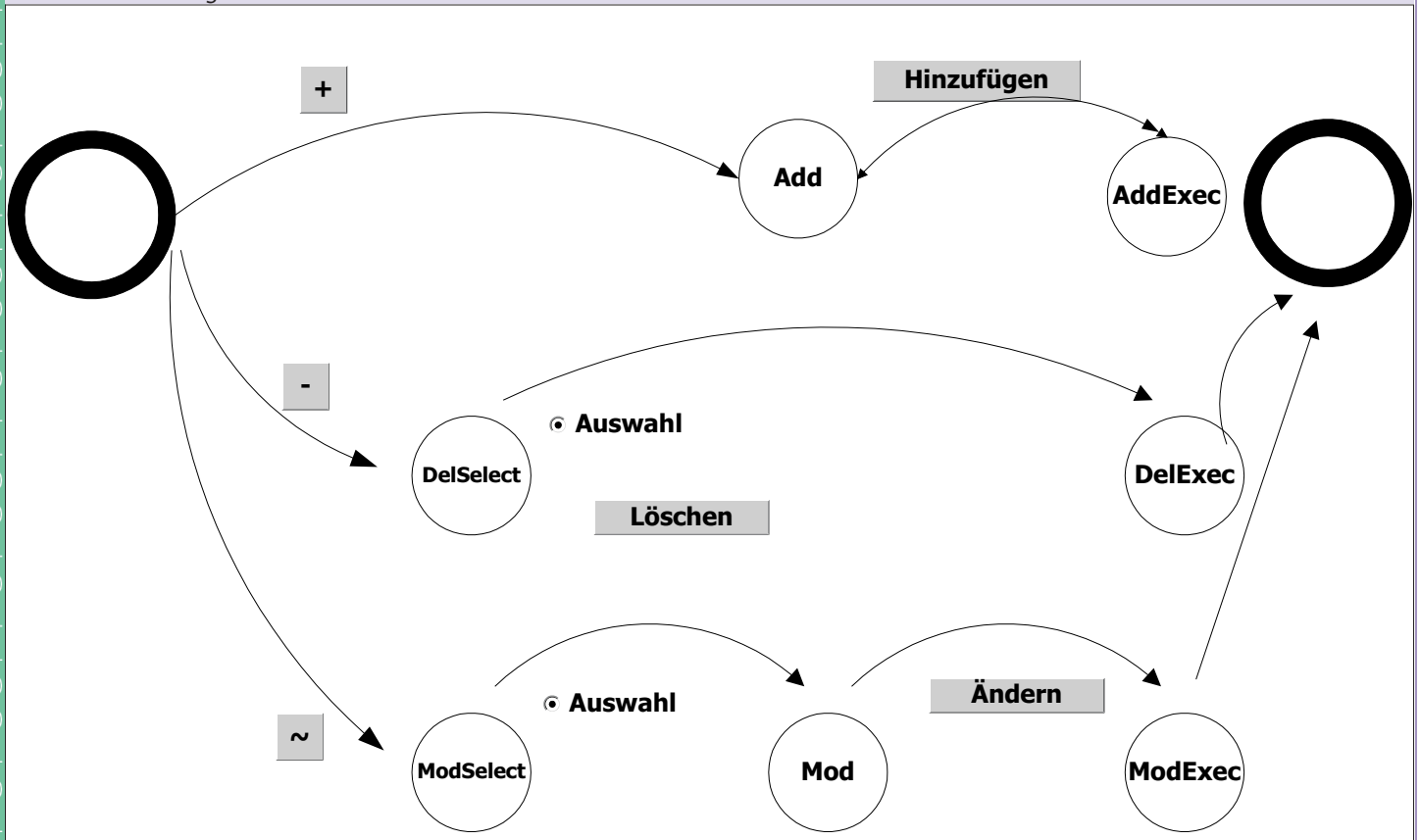
Der Pfad **"Ändern"** beginnt mit einer Ausgabe aller Datensätze aber mit einem entsprechenden Editierelement, das eine Auswahl erlaubt. Es müssen nur jene Felder ausgegeben werden, die zu einer eindeutigen Identifikation des Datensatzes erforderlich sind. Jedes Auswahl-Element (hier ein Radio-Button) bekommt als Namen die `ID` des betreffenden Datensatzes. Durch die Auswahl des betreffenden Datensatzes gelangt man in den Zustand `Mod`, in dem am Browser eine Eingabemaske dargestellt wird, die mit den jeweiligen Feldinhalten der Datenbanktabelle gefüllt sind. Es gehört auch zum Zustand `Mod`, dass alle Eingabefehler, zum Beispiel leere Felder oder unzulässige Eingaben, durch entsprechende JavaScript-Funktionen erkennt. Ist die Eingabe korrekt, gelangt man in den Zustand `ModExec`, der alle eingegebenen Daten in die Datenbank zu schreiben hat. Es muss keine Ausgabe zum Client erfolgen, außer wenn Fehler auftreten oder vielleicht eine einfache Quittung *"Datensatz geändert"*.

Die letzten drei Absätze enthalten absichtliche Symmetrien in der Formulierung, die darauf hindeuten, dass die im Bild jeweils übereinander liegenden Zustände ähnlichen, oft sogar identischen Kode enthalten. Wenn es daher jetzt darum geht, ein geeignetes Programmkonzept zu entwerfen, ist es zweckmäßig, diese Symmetrie auszunützen, damit zum Beispiel eine Programmmaske zur Dateneingabe in den Zuständen `Add` und `Mod` nicht etwa zwei Mal kodiert wird.

Es gibt zwei Entwurfsmöglichkeiten:

- Jeder Zustand wird in einer Datei kodiert und der Ablauf wird durch den korrekten Ablauf der jeweils nächsten Datei sichergestellt (Vorteil: schnell, übersichtlich, Nachteil: redundant)
- es gibt nur eine Datei, die im Stil einer *State Machine* abläuft und durch eine Zustandsvariable gesteuert wird (Vorteil: kompakt, Nachteil: langsamer)

Zustandsdiagramm für das Editieren einer Webdatenbank mit den Zuständen Add, AddExec, DelSelect, DelExec, ModSelect, Mod und ModExec. Diese Zustände können in gleichnamigen ASP-Dateien programmiert werden oder in einer einzigen Datei, die durch eine Zustandsvariable gesteuert wird.



Hier implementiert wurde die Methode der *State Machine*. Als Zustandsvariable wird `State` verwendet. Dieselbe Datei **`redaktion.asp`** ruft sich immer wieder selbst auf und übergibt die Zustandsvariable `State` in einem versteckten Formularfeld.

Das Datenbankzugriffsprogramm **`redaktion.asp`** hat zwei Aufrufmodi:

- ohne Parameter
- mit Parameter `who=redakteur`

Ohne Parameter zeigt das Programm den Datenbankinhalt nur an. Mit dem Parameter `who=Redakteur` zeigt das Programm zusätzlich drei Editierbuttons ⊞ ~ ⊟, für die drei Funktionen hinzufügen (`add*`), ändern (`mod*`), löschen (`del*`). Wenn eine dieser Tasten gedrückt wird, gelangt das Programm aus dem Grundzustand in den nächsten im Zustandsdiagramm.

Die nebenstehende Tabelle zeigt noch einmal die Symmetrien des Programms.

Variable State	""	"Add"	"AddExec"	"DelSelect"	"DelExec"	"ModSelect"	"Mod"	"ModExec"
Eingabemaske		●					●	
Auswahl				●		●		
keine Ausgabe			●		●			●
Parameter			Felder		ID		ID	Felder, ID
SQLquery	SELECT		INSERT	SELECT	DELETE	SELECT	SELECT	UPDATE

redaktion.asp

```
<%@ Language=VBScript %>
<% Option Explicit %>
<%

'Wer und wo bin ich?
Dim ScriptName
ScriptName = Request.ServerVariables("SCRIPT_NAME")

'Datenbank im selben Verzeichnis
Const DateinameDatenbank = "redaktion.mdb"
Dim Path
Path = Request.ServerVariables("PATH_TRANSLATED")
Path = Left(Path, InStrRev(Path,"\"))+DateinameDatenbank

'QueryString --ANFANG--
Dim Redakteur
If UCase(Request.QueryString("Who"))="REDAKTEUR" Then
     Redakteur=True
Else
     Redakteur=False
End If

Dim State : State=Request.QueryString("State")

Dim Datum
Dim Betreff
Dim Beitrag
Dim Link
Dim ID

Select Case State

Case "AddExec", "ModExec"

     Datum = Request.QueryString("DATUM")
     Betreff = Request.QueryString("BETREFF")
     Beitrag = Request.QueryString("BEITRAG")
     Link = Request.QueryString("LINK")

End Select

Select Case State
Case "Mod", "ModExec", "DelExec"
     ID = Request.QueryString("ID")
     If ID="" Then
          Response.Write "Fehler bei Parameter&uuml;bergabe"
          Response.End
     End If
End Select

'QueryString --ENDE--

'Datenbank öffnen
Const adModeReadWrite = 3
Dim Redaktionsdatenbank
Set Redaktionsdatenbank = Server.CreateObject("ADODB.Connection")
Redaktionsdatenbank.Mode=adModeReadWrite

'Datenbank öffnen mit ODBC
'Redaktionsdatenbank.Open "redaktion"

'Datenbank öffnen mit Treiber-Name unhd Pfad
Redaktionsdatenbank.Open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="+Path

Dim SQLquery
Dim RS

'Was brauchen wir von der Datenbank
Select Case State

Case "", "ModSelect", "DelSelect"

     SQLquery = _
          "SELECT REDAKTION.* " & _
          "FROM REDAKTION " & _
          "ORDER BY REDAKTION.DATUM DESC;"
     Set RS = Redaktionsdatenbank.Execute(SQLquery)

Case "Mod"

     SQLquery = _
          "SELECT * " & _
          "FROM Redaktion " & _
          "WHERE Redaktion.ID='"+ID+"';"
     Set RS=Redaktionsdatenbank.Execute(SQLquery)

     If RS.EOF Then
          Response.Write "Keine Daten"
          Response.End
     Else
          Datum = RS("DATUM")
          Betreff = RS("BETREFF")
          Beitrag = RS("BEITRAG")
          Link = RS("LINK")
     End If

Case "AddExec":

     SQLquery = _
          "INSERT INTO Redaktion ( DATUM, BETREFF, BEITRAG, LINK) " &
_
          "SELECT " & _
               "'"+Datum+"', " & _
               "'"+Betreff+"', " & _
               "'"+Beitrag+"', " & _
               "'"+Link+"';"
     Application.Lock
     Response.Write SQLquery
     Redaktionsdatenbank.Execute(SQLquery)
     Application.UnLock

Case "ModExec"

     SQLquery = _
          "UPDATE Redaktion SET " & _
               "DATUM = '"+Datum +"', " & _
               "BETREFF = '"+Betreff+"', " & _
               "BEITRAG = '"+Beitrag +"', " & _
               "LINK = '"+Link+"' " & _
          "WHERE Redaktion.ID='"+ID+"';"
     Application.Lock
     Redaktionsdatenbank.Execute(SQLquery)
     Application.UnLock

Case "DelExec"

     SQLquery = _
          "DELETE " & _
          "FROM Redaktion " & _
          "WHERE Redaktion.ID='"+ID+"';"
     Redaktionsdatenbank.Execute(SQLquery)

End Select

'Wie geht es weiter?
Select Case State

Case "", "Mod", "ModSelect", "DelSelect"
```

```
Case "AddExec", "ModExec", "DelExec"

     Response.Redirect ScriptName+"?who=redakteur"

End Select


'Was sieht der Browser?
HTMLHeader Redakteur, "Redaktion", "Redaktion"
     %><FORM Name=RedaktionEdit Method=GET><%
     %><INPUT Type=HIDDEN Name=State><%
     Select Case State
     Case "":
          RedaktionOutput False
     Case "Add"
          EingabeMaske "AddExec", _
               ID, FormatDateTime(now,0), _
               Betreff, Link, Beitrag, "hinzufügen"
     Case "Mod"
          EingabeMaske
            "ModExec", ID, Datum, Betreff, Link, Beitrag, "ändern"
     Case "DelSelect"
          EditierButton "DelExec", "löschen"
          RedaktionOutput True
     Case "ModSelect":
          EditierButton "Mod", "ändern"
          RedaktionOutput True
     End Select
     %></FORM><%
HTMLFooter

'---------------------------------------------------------------------
Sub RedaktionOutputEditButtons
     %>
     <INPUT
          Type=BUTTON
          Value="+"
          onClick="window.location.href='<%=ScriptName%>?State=Add'">
     <INPUT
          Type=BUTTON
          Value="~"
          onClick=
            "window.location.href='<%=ScriptName%>?State=ModSelect'">
     <INPUT
          Type=BUTTON
          Value="-"
          onClick=
            "window.location.href='<%=ScriptName%>?State=DelSelect'">
     <%
End Sub
'---------------------------------------------------------------------
Sub RedaktionOutput (Buttons)
     %><TABLE><%
     Do While Not RS.EOF
          BeitragOutput Buttons
          RS.MoveNext
     Loop
     %></TABLE><%
End Sub
'---------------------------------------------------------------------
Sub BeitragOutput (Buttons)
     %><TR><%
     If Buttons Then
          %><TD VALIGN=TOP>
               <INPUT Type=RADIO Name=ID Value=<%=RS("ID")%>>
          </TD><%
     End If
     %><TD><%=FormatDateTime(RS("DATUM"),2)%><BR><%
     If RS("LINK")<>"" Then
          %><A HREF=
            "javascript:neu=window.open('<%=RS("LINK")%>');
             window.location.reload()">
          <B><%=RS("BETREFF")%></A></B><BR><%
     Else
          %><B><%=RS("BETREFF")%></B><BR><%
     End If
     %><%=RS("BEITRAG")%></TD><%
     %></TR><%
End Sub
'---------------------------------------------------------------------
Sub EingabeMaske (NextState, ID, Datum, Betreff, Link, Beitrag,
Beschriftung)

     %><SCRIPT Language="JavaScript">
     <!--
     function Redaktion_onAdd(state) {
          if (document.RedaktionEdit.BETREFF.value=="")
               alert ("Betreff eingeben");
          else {
               if (document.RedaktionEdit.LINK.value=="http://")
                    document.RedaktionEdit.LINK.value="";
               if (document.RedaktionEdit.BEITRAG.value=="")
                    alert ("Beitragstext eingeben");
               else {
                    document.RedaktionEdit.State.value=<%=NextState%>;
                    document.RedaktionEdit.action=document.URL;
                    document.RedaktionEdit.submit();
               }
          }
     }
     //-->
     </SCRIPT>
     <INPUT Type=Hidden NAME=ID Value="<%=ID%>">
     <TABLE>
     <TR><TD>Datum</TD>
       <TD><INPUT
          Type=Text
          NAME=DATUM
          Value="<%=CStr(Datum)%>"></TD></TR>
     <TR><TD>Betreff</TD>
       <TD><INPUT
          Type=Text
          Name=BETREFF
          Value="<%=Betreff%>"
          size=40></TD></TR>
     <TR><TD>Link (optional)</TD>
          <TD><INPUT
            Type=Text
            Name=LINK
            Value="<%=Link%>"
            size=40></TD></TR>
     <TR><TD>Beitrag</TD>
          <TD><TEXTAREA
            Name=BEITRAG
            cols=40
            rows=5><%=Beitrag%></TEXTAREA></TD></TR>
     <TR><TD></TD><TD>
       <INPUT Type=BUTTON
          Value="<%=Beschriftung%>"
          onClick="Redaktion_onAdd('<%=state%>')">
       <INPUT Type=BUTTON
          Value=Abbrechen
          onClick=
            "window.location.href='<%=ScriptName%>?who=redakteur'">
     </TD></TR>
     </TABLE>
     <%
End Sub
'---------------------------------------------------------------------
Sub EditierButton (NextState, Beschriftung)
     %><P>
     <INPUT Type=BUTTON
          Value="<%=Beschriftung%>"
          onClick="Select_onClick('<%=State%>')"></P>

     <SCRIPT>
     function Select_onClick(state) {
          document.RedaktionEdit.action="<%=ScriptName%>"
          document.RedaktionEdit.State.value=<%=NextState%>
          document.RedaktionEdit.submit()
     }
     </SCRIPT>
     <%
End Sub
'---------------------------------------------------------------------
Sub HTMLHeader (Redakteur, Titel, Ueberschrift)
     %><HTML><HEAD>
     <TITLE><%=Titel%></TITLE>
     </HEAD><BODY>
     <H1><%=Ueberschrift%><%
     'Editierknöpfe sind nur für Redakteure sichtbar
     If Redakteur Then
          RedaktionOutputEditButtons
     End If
     %></H1><%
End Sub
'---------------------------------------------------------------------
Sub HTMLFooter
     %><P><%=now%></BODY></HTML><%
End Sub
%>
```