

# Dynamischer Datenbankzugriff mit ASP

Von  
Thomas Ohlhauser, Tübingen

# Dynamischer Datenbankzugriff mit ASP

## 1. Grundlagen

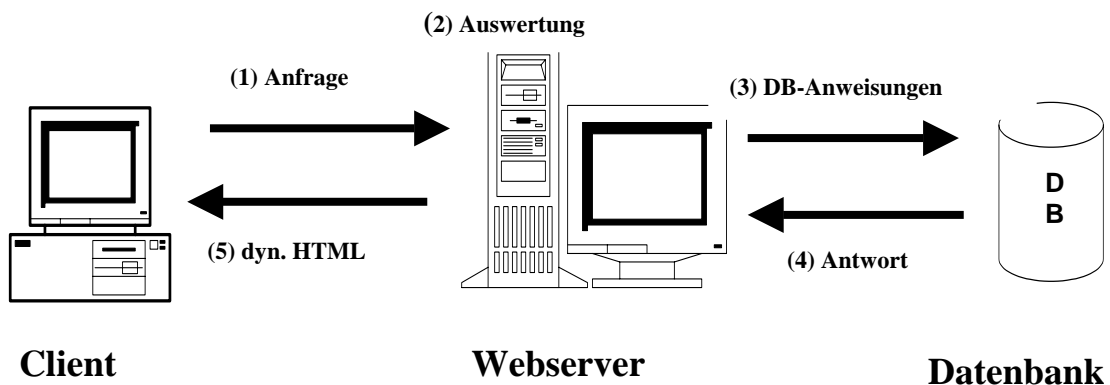
Die virtuelle Welt des Internets wird von zwei Lebensformen besiedelt: Clients und Server. Ein Rechner der sich im Internet anmeldet, ist der Client, der Webseiten in Form von HTML-Dokumenten von den Servern (Webservern) bezieht. Früher waren diese Webseiten vor allem statischer Natur, das heißt, die Inhalte einer Webseite blieben unverändert. Dies traf auf den Text als auch auf die Grafiken und die Formatierung zu. Wenn der Entwickler den Inhalt der Seite verändern wollte, musste er den Text manuell verändern und ihn dann zurück ins Internet stellen.

In den letzten Jahren haben die Webentwickler erkannt, dass statische Inhalte nicht ausreichen. Immer mehr Unternehmen haben damit begonnen, ihre Webseiten mit dynamischen Inhalten auszustatten. Dynamische Webseiten verändern ihren Inhalt im Laufe der Zeit automatisch.

Beispiele: Webshop mit Datenbankanbindung, Börsenkurse, Wetter.

Bereits bei Web-Anwendungen kleinen bis mittelgroßen Umfangs kann eine Anbindung an eine relationale Datenbank sinnvoll sein. Sie dient zum einen dazu, Eingaben des Benutzers oder Daten von beliebigen anderen Datenquellen laufend zu speichern und zum anderen liefert eine Datenbank den gesammelten Bestand an Informationen in dynamisch erzeugtes HTML.

Die dadurch herbeigeführte Strukturänderung der Web-Kommunikation zeigt sich bereits auf oberster Ebene: Aus dem ehemaligen 2-Schichten-System, bestehend aus dem Client und dem Webserver, ist eine sogenannte 3-Pfeiler-Architektur entstanden. Dritter Pfeiler dieser Struktur ist die Datenbank.



## 2. Erstellung dynamischer Webseiten mit ASP

Die grundlegenden Techniken zum Erstellen dynamischer Webseiten sind reich an Namen. Zu den prominentesten Vertretern zählt sicherlich **ASP (Active Server Pages)**. ASP erweitert den Webserver, so dass Webseiten durch den Zusatz von speziellen Tags programmierbar werden. Daten und Befehle werden in einer Datei gemischt. Der ASP-Code wird einfach als Skript in bestehende HTML-Seiten eingefügt und bei einer Anfrage seitens des Clients vom Webserver ausgeführt. Der Webserver erkennt an der Dateierweiterung .asp, dass die Datei neben HTML-Inhalten auch ASP-Skriptbefehle enthält.

```
Beispiel: beispiel1.asp

<html>
<head>
<title>Mein erstes ASP-Skript</title>
</head>
<body>
<h1>Hallo Welt</h1>
<br>
Auf meiner Homepage ist es
<%
=Now()
%>
<br>
<br>
<%
Response.Write "Ich glaube es wird Zeit eine Pause zu machen!"
%>
</body>
</html>
```

Die folgende Grafik zeigt die Wirkung des ASP-Skripts:



### **Erläuterung:**

Mit den Tags `<% .....%>` wird dem Server signalisiert, dass sich hier der ASP-Code befindet, der verarbeitet werden muss. Der Befehl `now()` bedeutet, dass jedes Mal, wenn man diese Seite aufruft, das Datum und die Uhrzeit vom Server erneut gelesen und angezeigt wird. Der Befehl `Response.Write` wird verwendet, um einen Wert auszugeben.

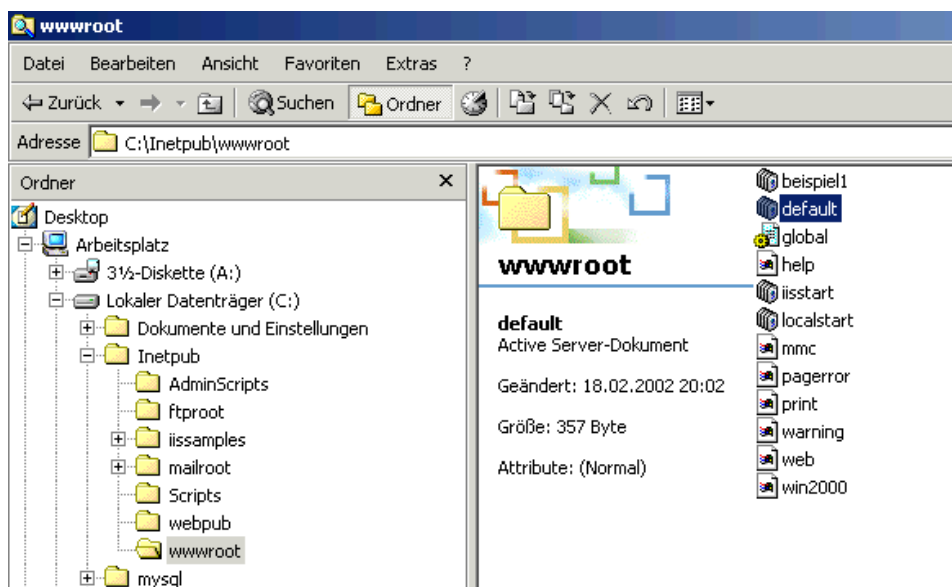
### 3. Voraussetzungen

Das Ansprechen dynamischer Webseiten mit ASP ist einfach und unkompliziert. Dazu ist es notwendig einen Webserver zu verwenden der ASP unterstützt.

Beispiele:

- PWS (Personal Web Server). Er ist im Lieferumfang von Windows 9x enthalten und kann bei der Installation aktiviert oder später nachinstalliert werden (*Systemsteuerung/Software*).
- Der IIS (Internet Information Server). Er ist im Lieferumfang von Windows NT/2000 enthalten und wird bei der Installation als Dienst eingerichtet. Zum Konfigurieren des **IIS** stehen in der *Systemsteuerung* unter *Verwaltung Tools* zur Verfügung:

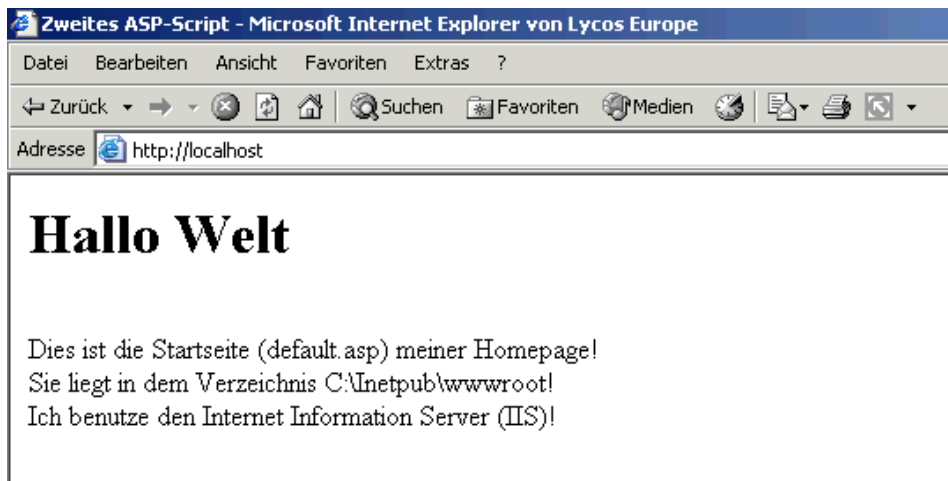
Nach dem Aktivieren/Starten des Webservers legt der IIS standardmäßig den Ordner C:\inetpub\wwwroot als Standardordner (Bild unten) für die Webdateien fest. Der Browser findet dann die Startseite (standardmäßig die Datei default.asp), mit http://<Computername>. Localhost ist die Bezeichnung (Computername) für den eigenen Rechner.



#### Beispiel: default.asp

```
<html>
<head>
<title>Zweites ASP-Script</title>
</head>
<body>
<h1>Hallo Welt</h1>
<br>
<%
Response.Write "Dies ist die Startseite (default.asp) meiner Homepage!" & "<br>"
Response.Write "Sie liegt in dem Verzeichnis C:\inetpub\wwwroot!" & "<br>"
Response.Write "Ich benutze den Internet Information Server (IIS)!" & "<br>"
%>
</body>
</html>
```

Die folgende Grafik zeigt die Wirkung des ASP-Skripts:



#### 4. Zugriff auf eine ACCESS-Datenbank über ADO

Eines der wichtigsten Anwendungsgebiete von ASP ist, Informationen aus einer Datenbank auszulesen und über den Webbrowser anzuzeigen. Um an die Objekte einer Datenbank zu gelangen, gibt es mehrere **Schnittstellen**.

Dabei stellt die Bibliothek Active Data Objects (ADO) den Mechanismus für die Verbindung mit einer Datenbank und die Bearbeitung der darin enthaltenen Daten bereit. Voraussetzung hierzu ist dass die Datenbankdatei in einem Ordner liegt, der vom Webserver erreicht werden kann.

Die wichtigsten ADO Objekte für Datenbankzugriffe sind:

- **Connection** (Datenbankverbindung) - Repräsentiert die Verbindung zur Datenquelle ,
- **Recordset** (Sammlung von Datensätzen) - Besteht aus den Datensätzen die aus einer Datenbankabfrage stammen und dem zugehörigen Cursor, der auf den aktuellen Datensatz zeigt,
- **Field** (Datenfeld) - Enthält Daten einer Datenbankspalte und Informationen über diese Daten wie Datentyp, Datenlänge und Ähnliches. Das Recordset Objekt stellt dabei die **Fields** Collection zur Verfügung, die alle Field Objekte eines Datensatzes beinhaltet.

Beispiel: dbzugriff1.asp

```
<html>
<head></head>
<body>
<%
'Pfad und Dateiname der Datenbank
strDB = "Data Source=" & _
Server.MapPath("haro.mdb")

'Parameter zum Öffner der Datenbank
strCon = "Provider=Microsoft.Jet.OLEDB.4.0;" & strDB

'Name der Tabelle oder Abfrage, die geöffnet werden soll
strTabelle= "Teile"

'Objekt für den Verbindungsaufbau mit der Datenbank
Set ConDB = Server.CreateObject("ADODB.Connection")
ConDB.Open strCon
Response.Write "Verbindung zur Datenbank geöffnet.<br>"

'Objekt für das Recordset-Objekt
Set RS = Server.CreateObject("ADODB.Recordset")
Rs.open strTabelle, ConDB
Response.Write "Leistungstabelle geöffnet.<br>"

'Befehle zur Bearbeitung der Daten

Response.Write "<HR>"
'Geht zum ersten Datensatz
Rs.MoveFirst

'Liest bis das Ende der Daten erreicht ist
Do While Not Rs.EOF
    Response.Write Rs("TeileNr") & " , "
    Response.Write Rs("Bez") & " , "
    Response.Write Rs("LagerNr") & " , "
    Response.Write Rs("VKPreis") & " "
    Response.Write Rs("Bestand") & " , "
    Response.Write "<br><hr>"
'Geht einen Datensatz weiter
```

```
Rs.MoveNext
Loop
```

```
'Verbindungsobjekt schließen
ConDB.close
Response.Write "Verbindung geschlossen.<br>"
%>
<hr>
</body>
</html>
```

### **Erläuterung:**

Den einzelnen Codezeilen sind Kommentare zur Erklärung vorangestellt. Einen Kommentar erkennt man an dem einfachen Hochkomma ('). Kommentare haben keinen Einfluss auf das Skript, sondern dienen dem Programmierer als Hilfestellung.

Die Parameter zum Öffnen der Datenbank werden über die Variable strCon festgelegt, wobei der Parameter Provider den Treiber definiert und Data Source den Pfad und Dateinamen der Datenbank bestimmt. Der Treibername Microsoft.Jet.OLEDB.4.0 arbeitet mit ACCESS 97 und ACCESS 2000.

ASP ist vereinfacht formuliert eine Sammlung von fünf vordefinierten Objekten. Das Beispiel macht lediglich vom Objekt Server Gebrauch. Im vorliegenden Fall wird es dazu genutzt, eine Instanz von ADODB.Connection zu erzeugen. Hierbei handelt es sich um eine Schnittstelle, die den Zugriff auf Datenbanken wesentlich erleichtert. Um schließlich eine Verbindung zur Datenbank herzustellen, erfolgt ein Aufruf der Methode Open. Diese Verbindung zur Datenbank wird letztlich durch den Aufruf der Funktion Close am Ende des Scripts wieder geschlossen.

Nachdem die Verbindung zur Datenbank geöffnet ist, sollen alle Teile mit Teilenummer, Bezeichnung, Lagernummer, Verkaufspreis, Bestand in einer Tabelle angezeigt werden.

Im ersten Schritt wird die Verbindung zur anzuzeigenden Tabelle (Leistung) geöffnet. Anschließend kann man mit dem Recordset-Objekt auf die Tabellendaten und die Ergebnisse einer Access-Abfrage zugreifen. Den Inhalt eines Datenbank-Feldes erhält man mit dem Befehl Recordset(„Feldname“). Um den Satzzeiger in der Tabelle zu bewegen gibt es verschiedene Befehle (Methoden) und Eigenschaften. „MoveFirst“ bewegt den Satzzeiger auf den ersten Datensatz des Recordsets. „Do While Not Rs.EOF“ bedeutet, dass die Tabelle solange durchlaufen werden soll, bis sich der Satzzeiger hinter dem letzten Datensatz der Tabelle befindet. Mit „Loop“ wird das Recordset beendet. „ConDB.Close“ schließt die Verbindung zur Datenbank.

Ändert man die Datensätze in einer Tabelle, so ändert sich automatisch die Anzeige der Datensätze im Webbrowser.

Die folgende Grafik zeigt die Datensätze der Tabelle Teile:

http://localhost/industrie/db\_zugriff1.asp - Microsoft Internet Explorer

Datei Bearbeiten Ansicht Favoriten Extras ?

Zurück Suchen Favoriten Medien

Adresse http://localhost/industrie/db\_zugriff1.asp

Verbindung zur Datenbank geöffnet.  
Leistungstabelle geöffnet.

---

10000	, Tischstativ , F1 , 9,95 , 200
10001	, Stativ-Oberteil , T1 , 2,5 , 200
10002	, Stativ-Fuß , T1 , 0,5 , 450
10003	, Stativ-Gewinde , T1 , 0,05 , 700
10004	, Handlupe 90 mm , F1 , 9,59 , 600
10005	, Tisch 90 mm , T1 , 1,5 , 350

## 5. Abfragen mit SQL

In den bisherigen Beispielen wurden lediglich Daten aus einer bestehenden Tabelle vollständig ausgelesen und über den Webbrowser angezeigt. Um mit einer Datenbank zu kommunizieren und Befehle darin auszuführen, verwendet man allerdings eine weitere Sprache. **SQL (Structured Query Language)** ist eine Datenbanksprache für **relationale Datenbanken**. Sie ermöglicht dem Benutzer das Kommunizieren mit dem eigentlichen Datenbanksystem. SQL wird vom **American National Standard Institut (ANSI)** als Standardsprache für relationale Datenbanken erklärt.

SQL gliedert sich in Anweisungen für die Datendefinition (Tabellen erstellen, ändern und indizieren), Datenmanipulation (Datensätze einfügen, ändern und löschen) sowie die Datenabfrage.

### Beispiel: dbzugriff2.asp

```
<html>
<head></head>
<body>
<%
'Pfad und Dateiname der Datenbank
strDB = "Data Source=" & _
Server.MapPath("haro.mdb")

'Parameter zum Öffnen der Datenbank
strCon = "Provider=Microsoft.Jet.OLEDB.4.0;" & strDB

'Name der Tabelle oder Abfrage, die geöffnet werden soll
strSQL = "Select TeileNr, Bez, LagerNr, VKPreis, Bestand From Teile Where LagerNr ='T1';"

'Objekt für den Verbindungsaufbau mit der Datenbank
Set ConDB = Server.CreateObject("ADODB.Connection")
ConDB.Open strCon
Response.Write "Verbindung zur Datenbank geöffnet.<br>"

'Objekt für das Recordset-Objekt
Set Rs = Server.CreateObject("ADODB.Recordset")
Rs.open strSQL, ConDB

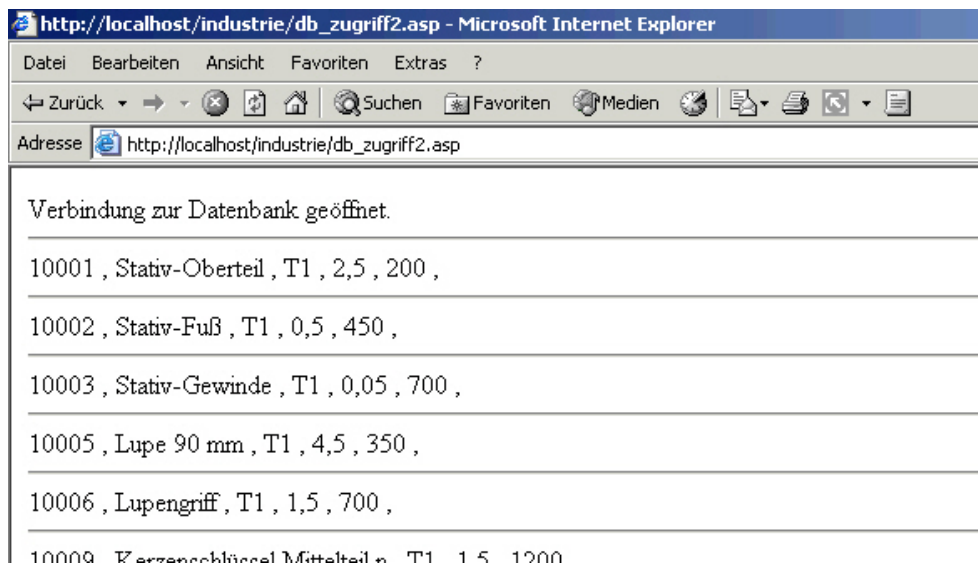
'Befehle zur Bearbeitung der Daten
Response.Write "<HR>"

'Geht zum ersten Datensatz
Rs.MoveFirst

'Liest bis das Ende der Daten erreicht ist
While Not Rs.EOF
Response.Write Rs("TeileNr") & " , "
Response.Write Rs("Bez") & " , "
Response.Write Rs("LagerNr") & " , "
Response.Write Rs("VKPreis") & " , "
Response.Write Rs("Bestand") & " "
Response.Write "<br><hr>"
Rs.MoveNext
Wend

'Verbindungsobjekt schließen
ConDB.close
Response.Write "Verbindung geschlossen.<br>"
%>
<hr>
</body>
</html>
```

Die folgende Grafik zeigt die SQL-Abfrage:



### **Erläuterung:**

Die Grundstruktur einer **SQL – Abfrage** sieht wie folgt aus:

<b>Select</b>	<b>Attribute</b>
<b>From</b>	<b>Tabellen</b>
<b>Where</b>	<b>Selektionskriterien;</b>

Dabei sind nur die Select- und die From- Klausel zwingend. Alle SQL-Anweisungen sind mit einem Semikolon (;) abzuschliessen.

Beispiel: Es werden alle Teile gesucht, welche im Lagerort "T1" eingelagert sind.

```
Select TeileNr, Bez, LagerNr, VKPreis, Bestand  
From Teile  
Where LagerNr='T1';
```

Im ASP-Skript werden die SQL-Befehle zur weiteren Verarbeitung in eine Variable gepackt.

```
strSQL = "Select TeileNr, Bez, LagerNr, VKPreis, Bestand From Teile Where LagerNr ='T1';"
```

Für den Zugriff auf die Datenbank benötigt man die Befehle, die in den vorhergehenden Beispielen erläutert wurden. Da man statt der gesamten Leistungstabelle nur bestimmte Daten sehen möchte, wird lediglich der Parameter Leistungstabelle durch die **SQL-Anweisung** ausgetauscht.

Was schon anhand dieses kleinen Beispielen ersichtlich ist: Um Daten aus einer Datenbank in einem ASP Skript verwenden zu können, ist es durchaus hilfreich, wenn man ein gesundes Wissen über die Grundlagen der Datenbankprogrammierung und der Sprache SQL hat.

## 6. Erstellen von HTML – Formularen zur Datenbank-Abfrage

Für die Administration einer Datenbank ist es sinnvoll auf einer HTML-Seite Formularmasken zu erstellen, um folgende Aufgaben durchführen zu können:

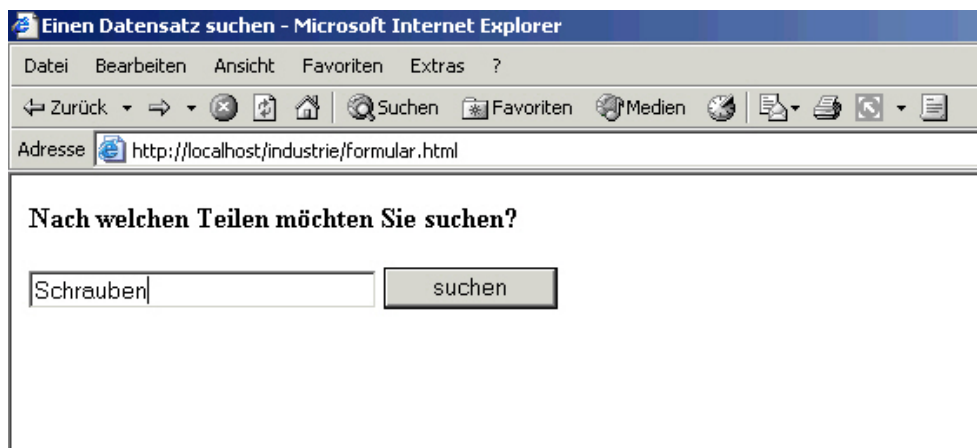
- Suche nach bestimmten Datensätzen
- Hinzufügen von Datensätzen
- Anzeigen der gesamten Datenbank
- Löschen von Datensätzen.

### Formular „Einen Datensatz suchen“

Das Formular besteht aus einem Eingabefeld (db\_feld1) und einem Submit-Button, der die Suchanfrage an die Seite db\_ergebnis.asp schickt.

```
<HTML>
<HEAD>
<TITLE>Einen Datensatz suchen</TITLE>
</HEAD>
<BODY>
<B>Nach welchen Teilen möchten Sie suchen?</B>
<FORM METHOD=POST ACTION="db_ergebnis.asp">
<INPUT TYPE="text" NAME="db_feld1" style="width:200px" size="20">
<INPUT TYPE="submit" value="suchen" style="width:100px">
</FORM>
</BODY>
</HTML>
```

Die folgende Grafik zeigt das HTML-Formular:



## Ergebnis einer Datenbanksuche ausgeben – db\_ergebnis.asp

```
<html>
<head></head>
<body>
<%
'Speichern der im Request-Objekt übergebenen Formulare Daten
Dim feld1
feld1 = Request.Form("db_feld1")

'Pfad und Dateiname der Datenbank
strDB = "Data Source=" & _
Server.MapPath("haro.mdb")

'Parameter zum Öffnen der Datenbank
strCon = "Provider=Microsoft.Jet.OLEDB.4.0;" & strDB

'Name der Abfrage, die geöffnet werden soll
strSQL = "Select TeileNr, Bez, LagerNr, VKPreis, Bestand From Teile Where Bez LIKE '%" & feld1 & "%'";

'Objekt für den Verbindungsaufbau mit der Datenbank
Set ConDB = Server.CreateObject("ADODB.Connection")
ConDB.Open strCon
Response.Write "Verbindung zur Datenbank geöffnet.<br>"

'Objekt für das Recordset-Objekt
Set Rs = Server.CreateObject("ADODB.Recordset")
Rs.open strSQL, ConDB

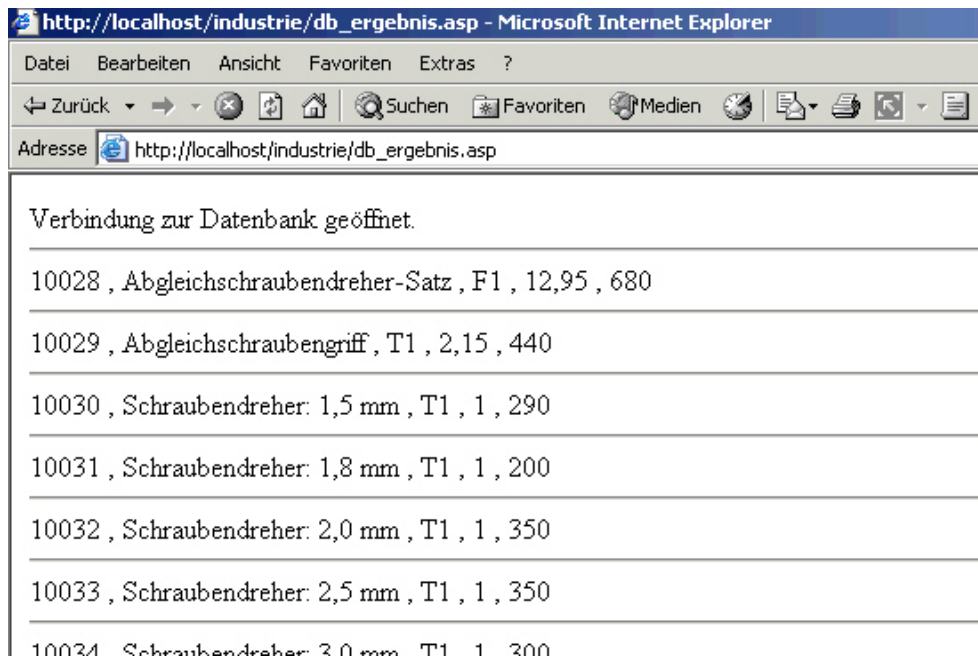
If Rs.EOF = False Then
Do While Not Rs.EOF
Response.Write Rs("TeileNr") & " , "
Response.Write Rs("Bez") & " , "
Response.Write Rs("LagerNr") & " , "
Response.Write Rs("VKPreis") & " , "
Response.Write Rs("Bestand") & " "
Response.Write "<br><hr>"
Rs.MoveNext
Loop

Else
Response.Write("<b>Leider keinen Eintrag gefunden!</b><br>")
End If

'Verbindungsobjekt schließen
ConDB.close

Response.Write "Verbindung geschlossen.<br>"
%>
<hr>
</body>
</html>
```

Die folgende Grafik zeigt das Ergebnis der Abfrage:



### **Erläuterung:**

Um die Programmierung übersichtlicher zu gestalten werden die im Request-Objekt übergebenen Formulardaten zunächst in eine Variable gespeichert.

```
Dim feld1
feld1 = Request.Form("db_feld1")
```

Danach wird die Verbindung zur Datenbank hergestellt und das Recordset-Objekt geöffnet. Mit der SQL-Abfrage Like '%Kriterium%' wird nach einem Teilstring innerhalb eines Feldes gesucht. Im Beispiel bezieht sich die Suche auf die Bezeichnung.

```
strSQL = "Select TeileNr, Bez, LagerNr, VKPreis, Bestand From Teile Where Bez LIKE '%" & feld1 & "%';"
```

Nur wenn das Recordset Datensätze als Ergebnis der Abfrage enthält, wird die Datenbankoperation auch ausgeführt.

```
If Rs.EOF = False Then
Do While Not Rs.EOF
```

Hat der User einen Begriff gesucht, der nicht in der Datenbank enthalten ist, wird eine Fehlermeldung ausgegeben.

```
Else
Response.Write("<b>Leider keinen Eintrag gefunden!</b><br>")
End If
```

Zum Schluss wird das Recordset und die Datenbankverbindung wieder geschlossen.

## Gesamte Tabelle durchsuchen

Bezieht sich die Suche nicht nur auf die Bezeichnung, so ist der SQL-Befehl folgendermaßen zu ergänzen:

Beispiel: db\_ergebnis1.asp

```
strSQL = "Select TeileNr, Bez, LagerNr, VKPreis, Bestand  
From Teile  
Where TeileNr & Bez & LagerNr & VKPreis & Bestand LIKE '%" & feld1 & "%';"
```

Die folgende Grafik zeigt das Ergebnis der Abfrage:

