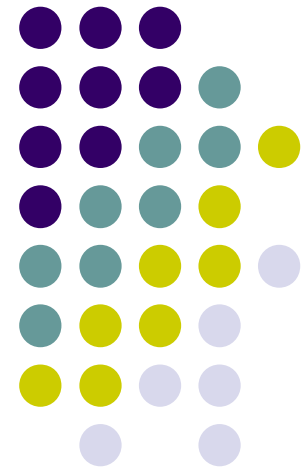
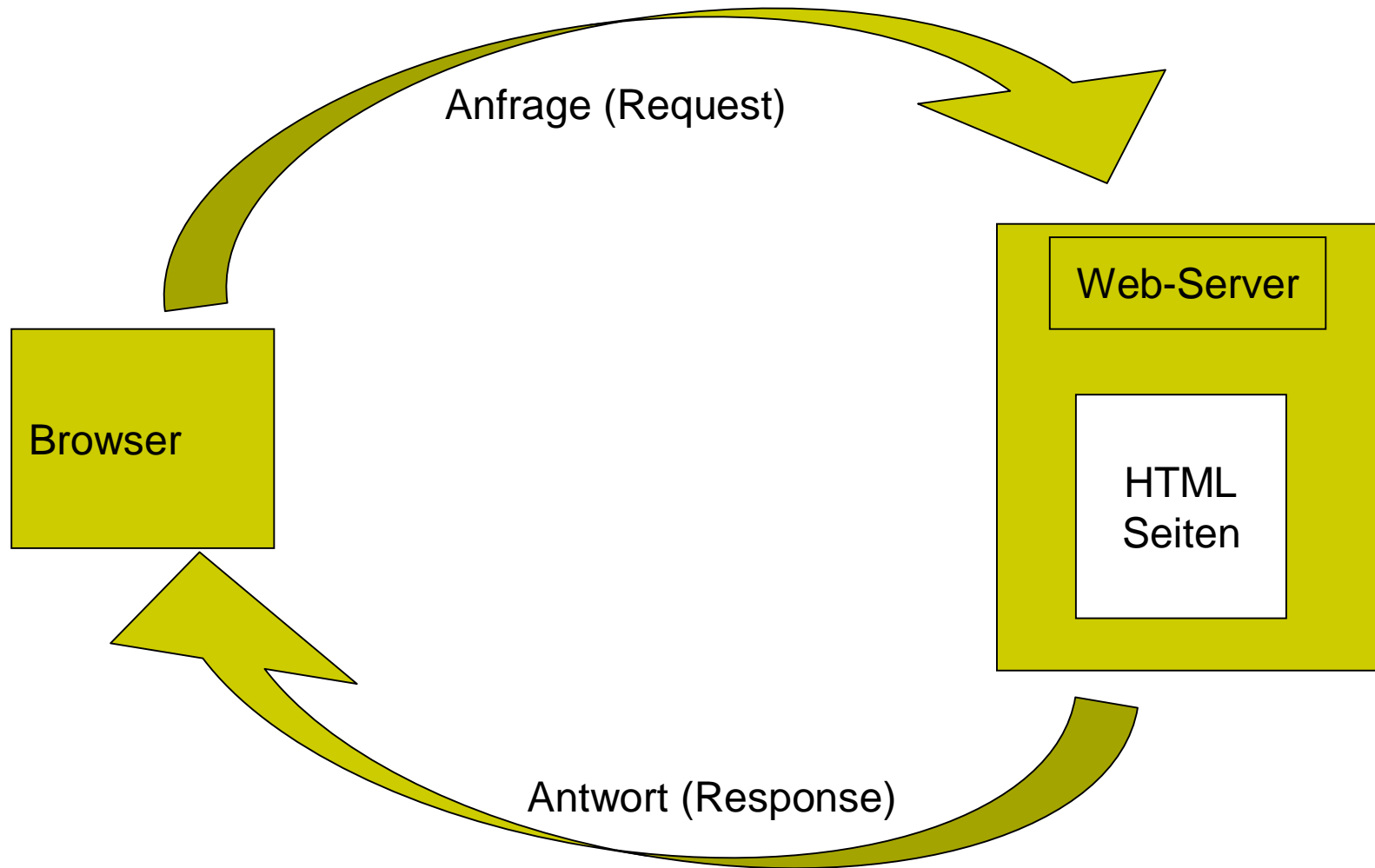


# ASP

Active **S**erver **P**ages  
VBScript



# Statische Webseite

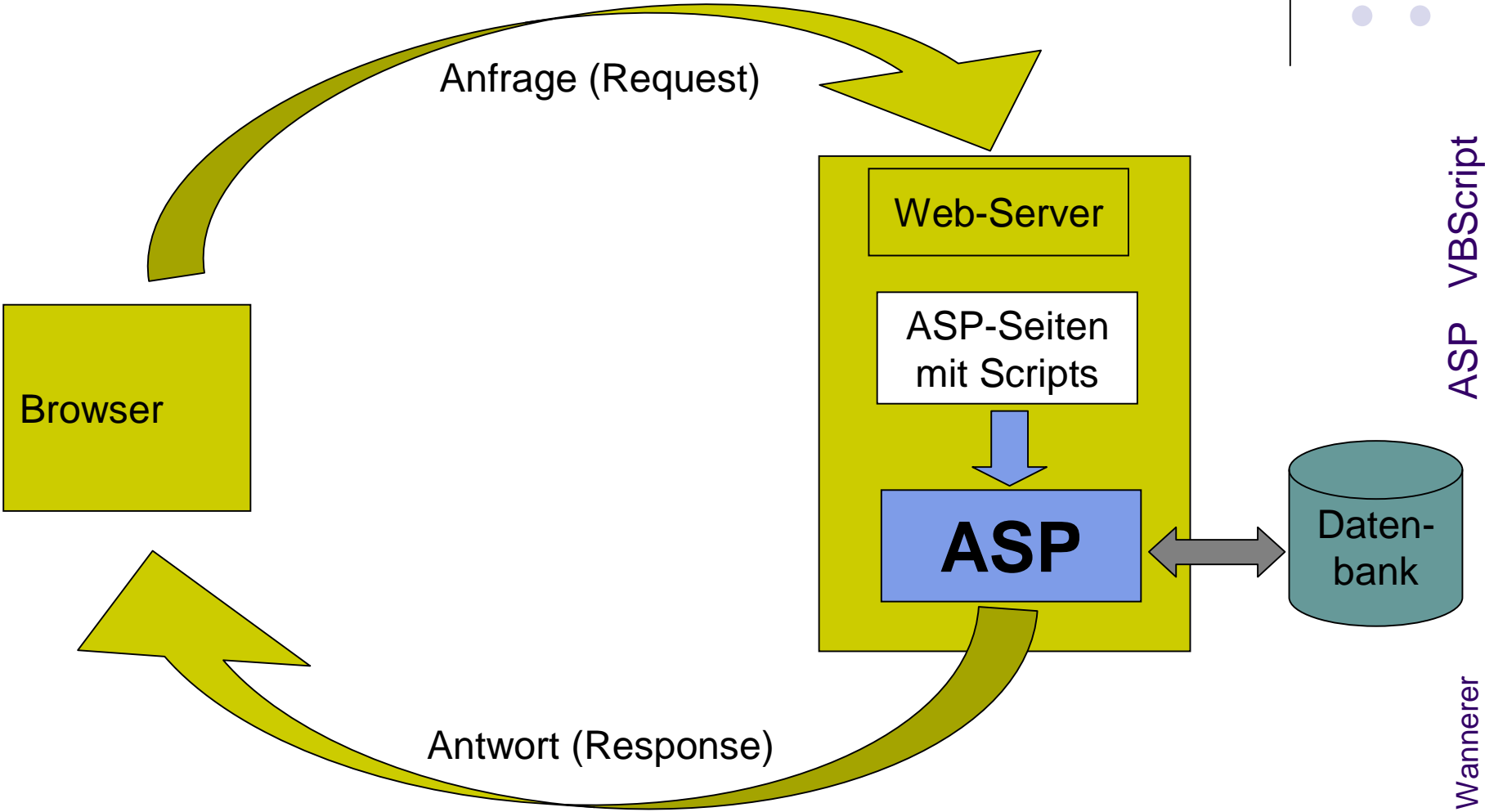


ASP VBScript

© Wannerer

Angeforderte Seite kommt (über HTTP), so wie auf dem Server gespeichert

# Dynamische Seite

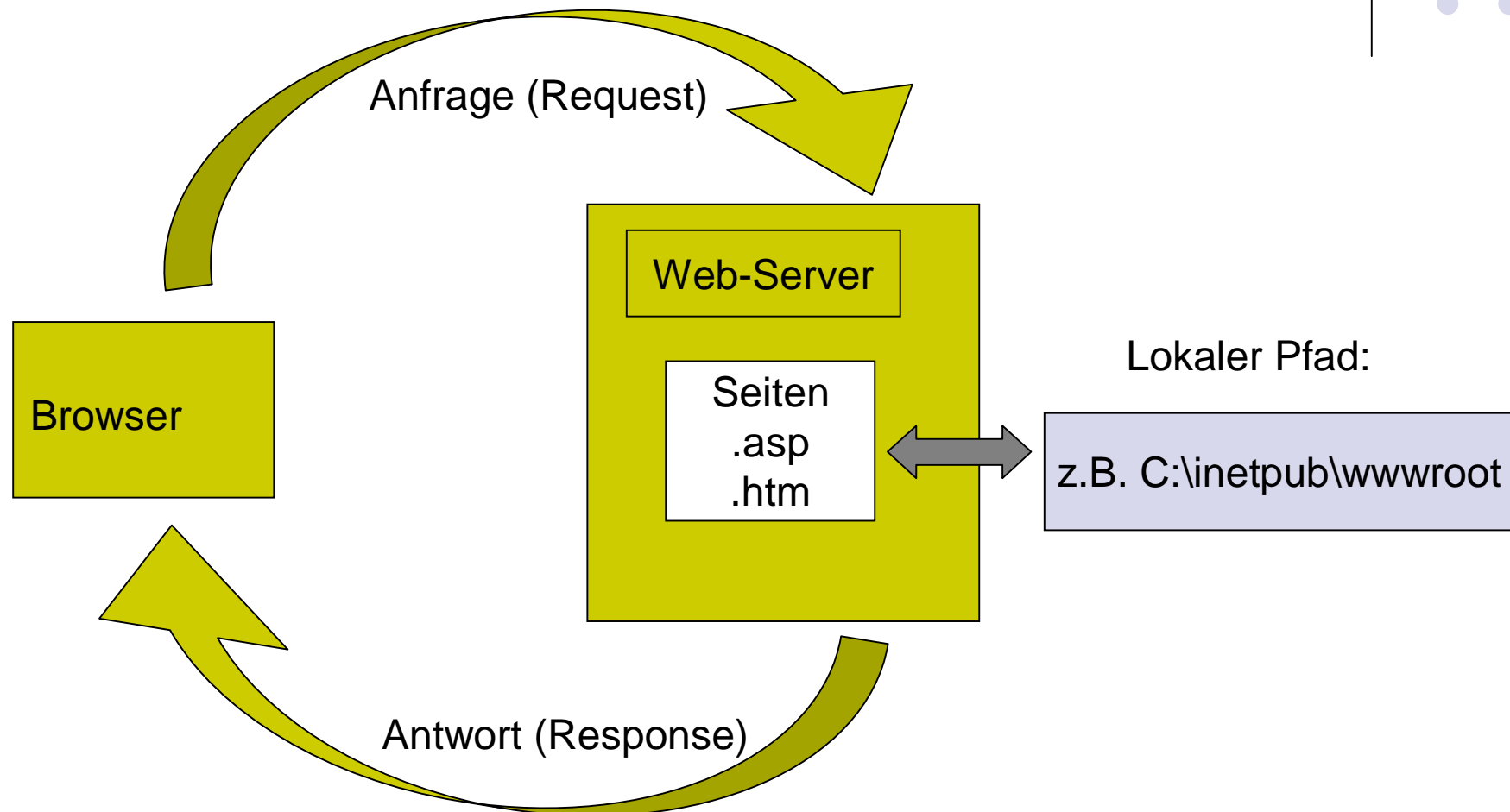


ASP verarbeitet Scripts und sendet HTML an den Browser

ASP VBScript

© Wannerer

# Lokaler Webserver (IIS, Babyweb...)



ASP VBScript

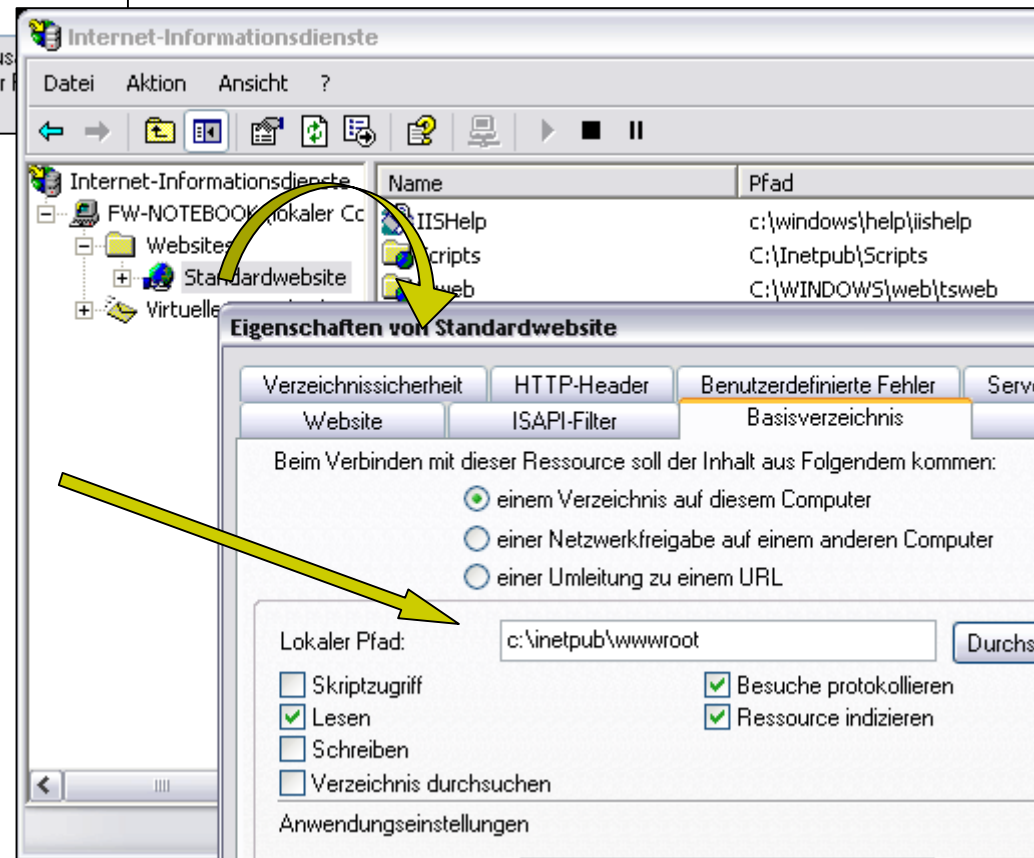
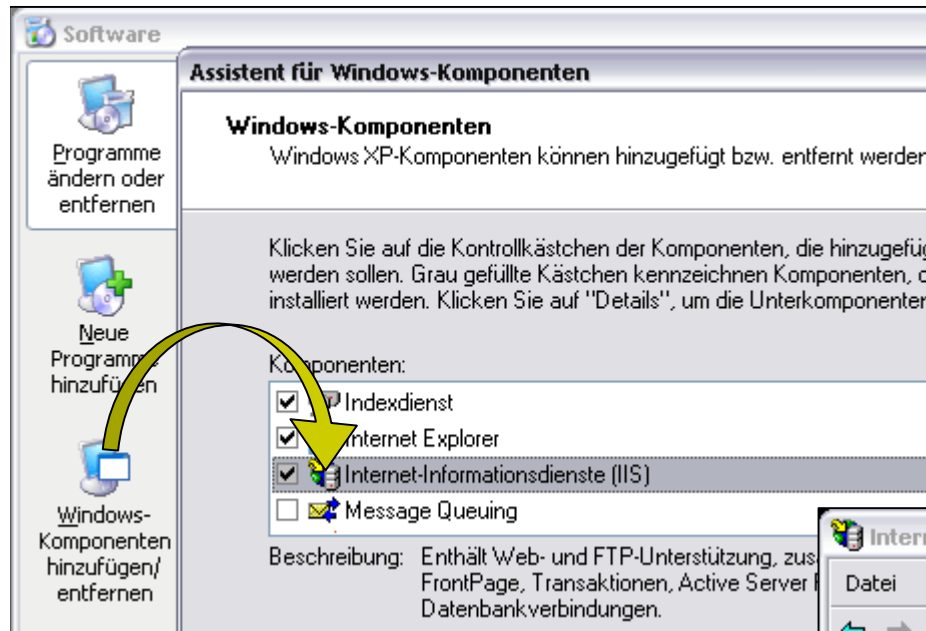
Aufruf des lokalen Web-Servers im Browser: <http://localhost/>

Alternativ zu localhost: [127.0.0.1](http://127.0.0.1/) oder [Computername](http://Computername)

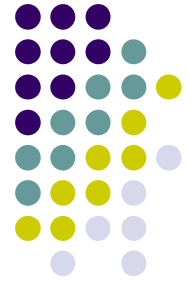
© Wannerer



# IIS Installation (2000 / XP Pro)



# ASP – Objekte (Response)



- | Erzeugt Ausgaben aus dem Script
- | Server integriert Ausgabe in das an den Benutzer gesendete HTML-Dokument
- | Häufigste Methode **Write**  
<% response.write(Variable) %> bzw.  
<% response.write("XXX") %>  
oder Kurzform(nur ein Ausdruck)  
<% = Variable %> bzw. <% = "XXX" %>

# Response-Objekt



- I HTML-Code unterdrücken  
`<% = Server.HTMLencode("<br />") %>`
- I Weiterleitung zu einer URL  
`<% Response.Redirect("URL") %>`
- I Ausgabe beenden ("schließen")  
`<% Response.End %>`

# ASP-Objekte (Request)



- | Request-Objekt
  - | Liest vom Browser übertragene Information
  - | Querystring (an URL angehängt): Paare von Bezeichnern und Werten („Kollektion“)  
<http://localhost/test.asp?name=Franz&ort=Wien>
  - | Gesamter Querystring ("Kollektion") übernehmen:  
`<% Request.QueryString %>`
  - | Einzelwert übernehmen:  
`<% Request.QueryString("Bezeichner") %>`  
`<% Request.QueryString("ort") %>`
  - | Querystring è Formulardaten mit GET-Methode übernehmen

# ASP-Objekte (Request; Formular)



- Übernahme von Formulardaten  
Formular: **GET** è **Request.QueryString**

Formular.htm

```
....  
<form method="get"  
action="test.asp">  
...  
<input name="ort"  
type="text">  
...  
</form>  
.....
```

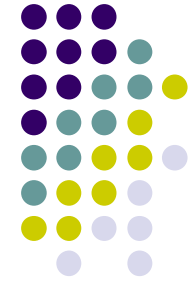


Test.asp

```
....  
<%  
o = request.querystring("ort")  
%>  
....
```

Formulardaten an URL angehängt

# ASP-Objekte (Request; Formular)



- Übernahme von Formulardaten  
Formular: **POST** → **Request.Form**

Formular.htm

```
<form method="post"  
action="test.asp">  
  
...  
  
<input name="ort"  
type="text">  
  
...  
  
</form>
```



Test.asp

```
....  
<%  
o = request.form("ort")  
%>  
  
....
```

# ASP-Objekte (Request; Formular)




- | übergebene Werteliste = "Kollektion"
- | **Request.Form("Bezeichner")** liefert Einzelwert
- | **Request.Form** liefert gesamte Kollektion
- | Kollektion auslesen mit **for ...each**-Schleife

```
<% for each feldname in request.Form %>  
    <p><% =feldname %> : <% =request.form(feldname) %></p>  
<% next %>
```

à Gleiche Anwendung bei der Kollektion **QueryString!**

# ASP-Objekte (Request; Servervariable)



- | Informationen aus dem HTTP-Header des Webservers
- | `Request.ServerVariables` (*ServerVariable*)
  - | `HTTP_REFERER` liefert URL der aufrufenden Seite
  - | `REMOTE_ADDR` liefert IP-Adresse des Browsers
  - | `QUERY_STRING` 
  - | `SCRIPT_NAME` – virtueller Pfad der aktuellen ASP-Seite
  - | `HTTP_USER_AGENT` – Browserdaten

Beispiel:

```
<% WoherKommIch = Request.ServerVariables("HTTP_REFERER") %>
```

# ADO



- | **ActiveX Data Objects**
- | Zugriff auf Datenbanken über Webseiten
- | Zugriff auf ODBC-Datenbanken direkt aus Skripts (VBScript)
- | Wichtige Objekte der ADODB-Klasse
  - | Connection (Verbindung)
  - | Recordset (Datensatzobjekt)

# Datenbankanbindung

## Connection-String (1)



- Connection-Objekt erstellen ("Instanziiieren")

```
<% Set Con = Server.CreateObject("ADODB.Connection") %>
```

- Verbindung öffnen (physischer Pfad der Datenbank bekannt) mit Verbindungszeichenfolge (eine Zeile!)

```
<% VerbString="Provider=Microsoft.Jet.OLEDB.4.0;  
Data Source=c:\inetpub\wwwroot\test\test.mdb;" %>  
<% Con.Open VerbString %>
```

**ACCESS 2007**

```
<% VerbString="Provider=Microsoft.ACE.OLEDB.12.0;  
Data Source=c:\inetpub\wwwroot\test\test.accdb;" %>
```

- Verbindungszeichenfolge
- Treiber (MS-Access) muss installiert sein
- Andere DB – anderer Provider (z.B. SQL)

# Datenbankanbindung

## Connection-String (2)



### I Connection Objekt erstellen

```
<% Set Con = Server.CreateObject("ADODB.Connection") %>
```

### I Physischer Pfad der DB NICHT bekannt: ***Server.MapPath***-Methode zeigt Pfad

```
<% Pfad = Server.MapPath("test.mdb") %>
```

```
<% VerbString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & Pfad & ";" %>
```

```
<% Con.Open VerbString %>
```

- Verbindungszeichenfolge
- Treiber (MS-Access) muss installiert sein

# Datenbankanbindung

## ODBC-Datenquelle (1)

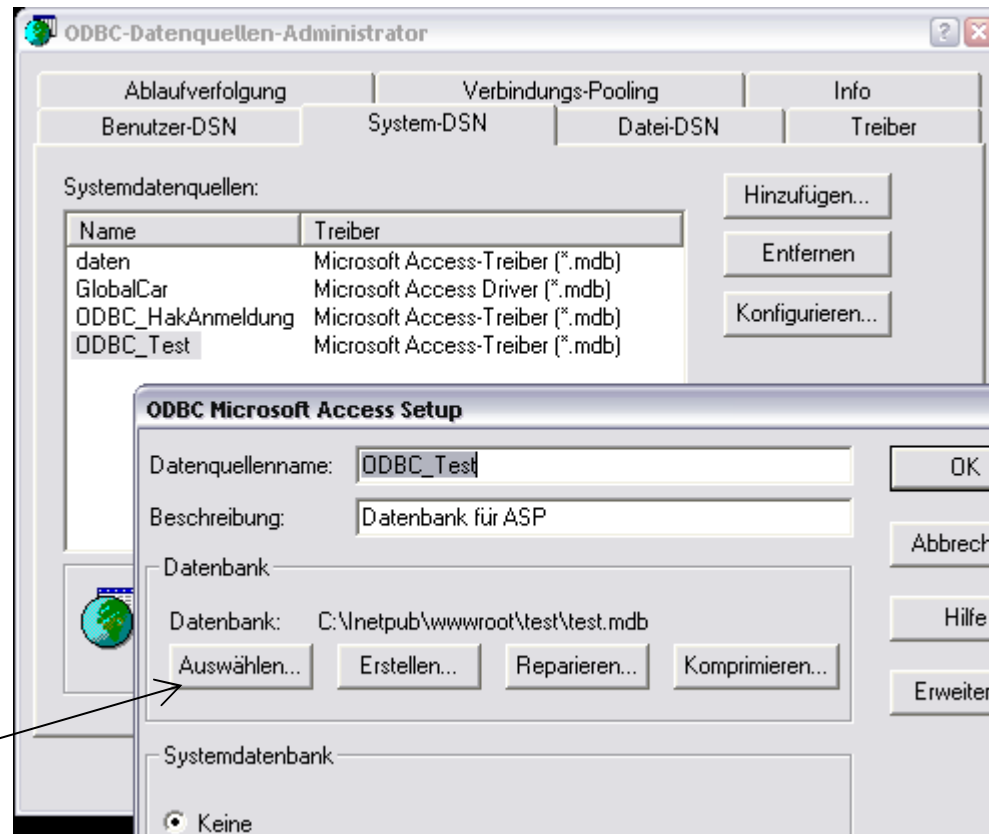


- ODBC erstellen: Systemsteuerung – Verwaltung – Datenquellen(ODBC) → System-DSN

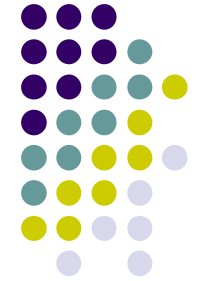
Vergeben von:

- **Name**
- **Pfad**

Datenbank bestimmen



# Datenbankanbindung ODBC-Datenquelle (2)



## I Connection Objekt

```
<% Set Con = Server.CreateObject("ADODB.Connection") %>
```

## I Verbindung öffnen (der Name der System-DSN ist hier "ODBC\_Test")

```
<% Con.Open "ODBC_Test" %>
```

# Datensatzobjekt - Recordset



Zur Ausgabe von Daten

## I Verbindung herstellen

```
<% Set Con = Server.CreateObject("ADODB.Connection") %>
```

```
<% Con.Open ("provider=microsoft.jet.oledb.4.0;datasource=c:\inetpub\wwwroot\test\test.mdb;") %>
```

## I Recordset instanziiieren

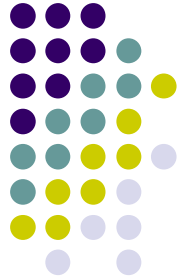
```
<% Set rs = Server.CreateObject("ADODB.Recordset") %>
```

## I SQL-Abfrage ausführen

```
<% sql="select * from daten" %>
```

```
<% rs.Open sql, con %>
```

# Datensatzobjekt - Recordset

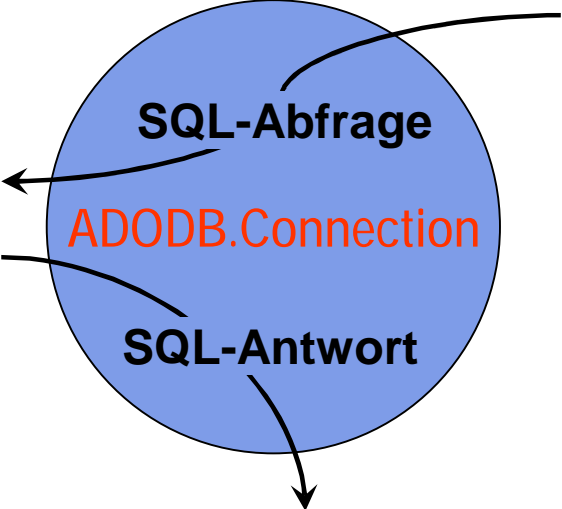


SQL-Tabelle  
(Access)

daten : Tabelle			
nr	zuname	groesse	geburtsdatum
1	Berger	155	12.12.1975
2	Adam	188	01.02.1988
3	Fugger	179	05.05.1955
4	Heinz	166	04.08.1993

RS Objekt

```
sql=" SELECT * FROM daten
      WHERE groesse<170"
rs.Open sql, con
```



rs("nr")	rs("zuname")	rs("grosse")	rs("geburtsdatum")
1	Berger	155	12.12.1975
4	Heinz	166	04.08.1993

Response.Write(rs("zuname"))

RS Objekt

rs.MoveNext

rs.eof

ASP VBScript

© Wannener

# Code mit Recordset



```
<% 'Verbindung zur Datenbank herstellen Connection-Objekt instanziiieren
set con = server.CreateObject("ADODB.Connection")
verbstring = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\inetpub\wwwroot\test\db\test.mdb;"
con.open verbstring
'Recordset-Objekt instanziiieren
set rs_daten = server.CreateObject("ADODB.Recordset")
'SQL-Anweisung bauen
sql = "select * from personen"
rs_daten.Open sql, con 'Recordset öffnen(=Abfrage ausführen)
rs_daten.MoveFirst 'gehe zum ersten Datensatz
<% 'Schleife durchläuft das Recordset und gibt Datensätze in HTML-Absätzen aus
Do While Not rs_daten.eof %>
    <p><% = rs_daten("id") %>&nbsp; <% = rs_daten("Zuname") %> <p>
    <% rs_daten.MoveNext 'gehe zum nächsten Datensatz
Loop %>
<% con.close 'Verbindung schließen und Objektverweise zerstören
set con = nothing
set rs_daten = nothing %>
```

Ausgabe von Daten in dynamischen Absätzen

ASP VBScript

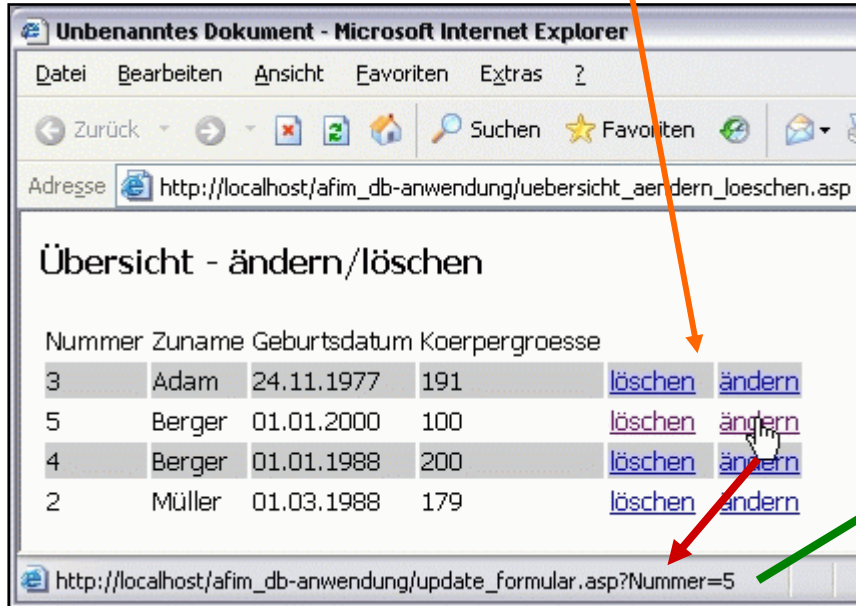
© Wannerer

# URL-Parameterübergabe (Querystring)

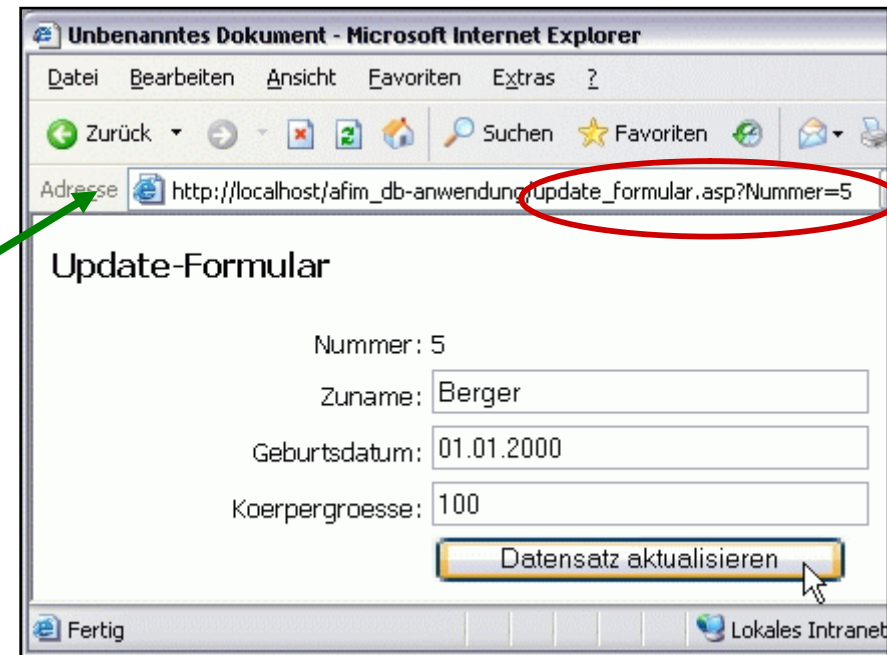


```
<A HREF="update_formular.asp?Nummer=<%= rs("Nummer") %>">ändern</A>
```

Hyperlink mit  
Parameterübergabe



Ausgewählter Datensatz im  
Formular; Inhalte (value)  
dynamisch erzeugt



```
<% sql="SELECT ..... WHERE Nummer =" & Request.QueryString(„Nummer“) & ";" %>
```

# SQL ohne Recordset



Bei Datenbankoperationen, die keine Ausgabe von Datensätzen erfordern  
(z.B. INSERT INTO, DELETE, UPDATE)

## I Verbindung herstellen

```
<% Set Con = Server.CreateObject("ADODB.Connection") %>
```

```
<% Con.Open ("provider=microsoft.jet.oledb.4.0;datasource=c:\inetpub\wwwroot\test\test.mdb;") %>
```

## I SQL-Abfrage ausführen(neuer Datensatz)

```
<% sql="INSERT INTO daten (Feld1, Feld2) VALUES ('Wert1', 'Wert2');" %>
```

```
<% con.execute sql %>
```

# Recordset – Methoden – Eigenschaften



## I Grundlegend

- I Open
- I Close
- I Delete
- I Update

## I Navigation

- I Move *anzahl*
- I MoveFirst
- I MoveNext
- I MovePrevious
- I MoveLast

## I Eigenschaften

- I BOF *-Beginn*
- I EOF *-Ende*

# DB-Operationen beenden

Am Ende des Scripts!

## I Verbindung schließen

```
<% con.close %>
```

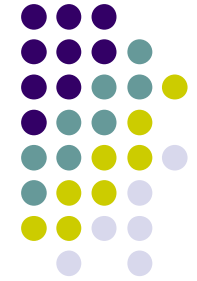
## I Objektverweise zerstören (Freigabe des Speicherplatzes am Server)

```
<% set con = nothing %>
```

```
<% set rs_daten = nothing %>
```

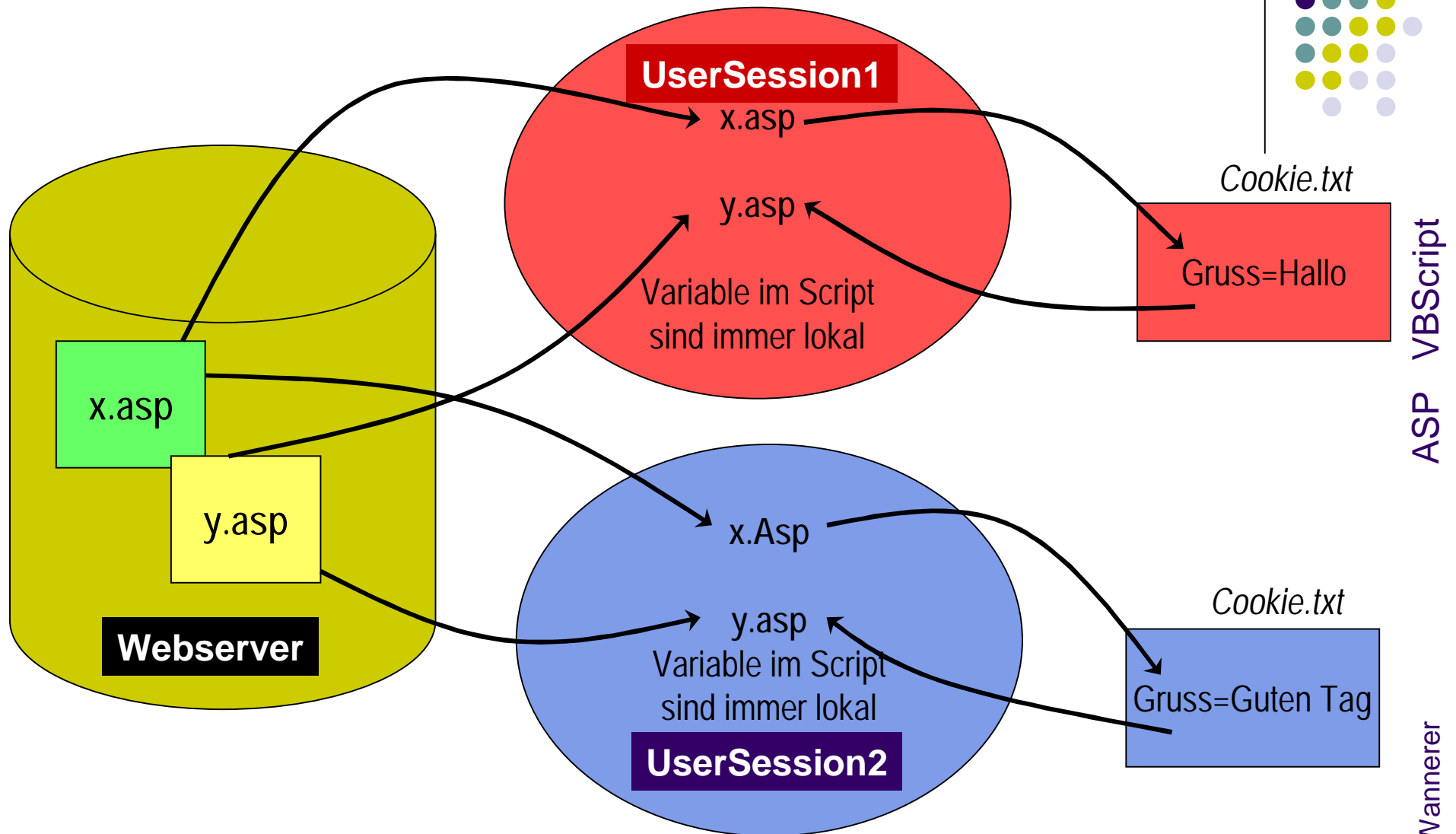


# Sessions (Sitzungen)



- | HTTP = verbindungsloses Protokoll = für jede Seite eine neue Verbindung
  - | Nutzer kann nicht zugeordnet werden
- | **Sessions** führen Status über mehrere Seiten hinweg mit (für "Warenkorb" etc.)
- | Sessions verwenden Cookies
  - | Cookies = Dateien, die der Browser anlegt und nur vom Webserver gelesen werden können
  - | Session-Variable

# Sessions - Arbeitsweise

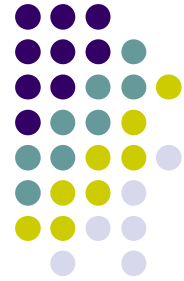


- Mehrfache Aufrufe des Scripts unabhängig voneinander
- Sessionvariable in lokalen Cookies gespeichert (VB-Variable nicht geeignet)

# Sessions - Anwendung

- | Session beginnt mit Besuch der Seite
- | Session-Ende
  - | Standard nach 20 Minuten oder andere Zeit
    - à `Session.Timeout = 30`
  - | Gesteuert: `Session.Abandon`
- | Sessionvariable behalten Wert
  - | Beispiele: 

```
Session("ID") = 12345678  
Session("Name") = "Franz"
```
- | Session-Nummer – vom Server vergeben zur Unterscheidung einzelner Sessions
  - | `Session.SessionID` (wird bei Sitzungsende gelöscht)
- | Session-Variable leeren
  - | `Session.Contents.Remove("Variable")`



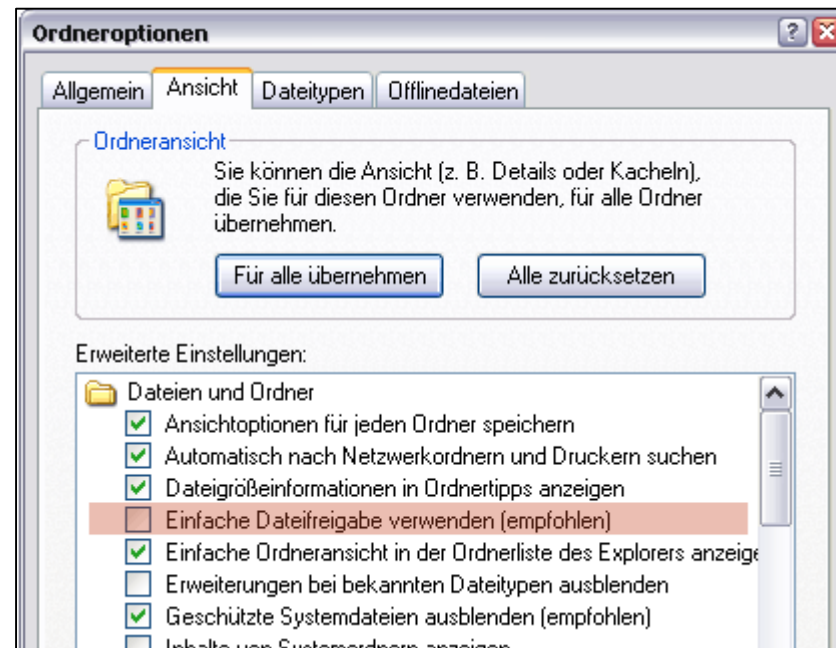
# Schreibrechte für die Datenbank (NTFS)



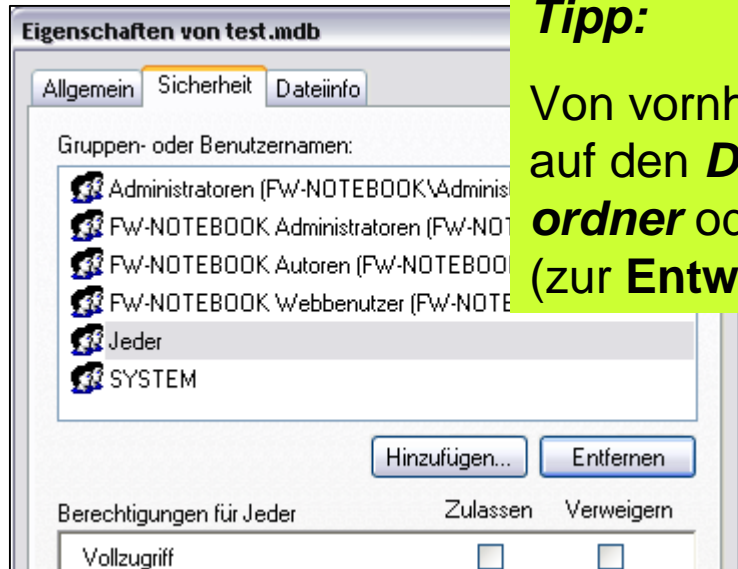
ASP-Fehlermeldung:

"... Muss eine aktualisierbare Abfrage enthalten"

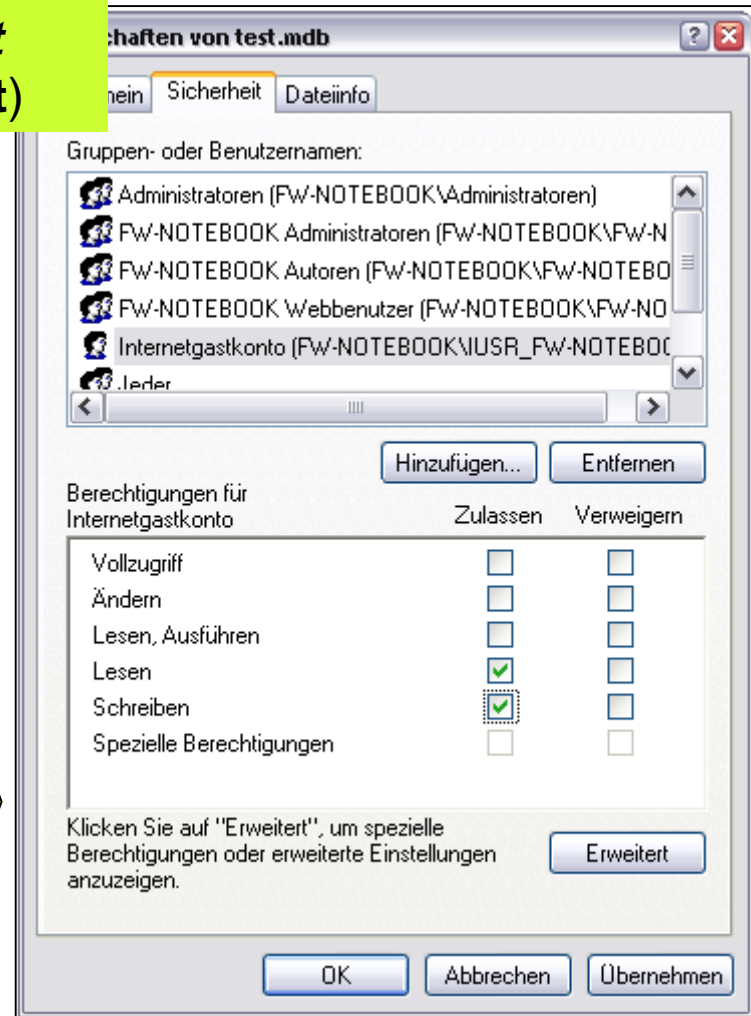
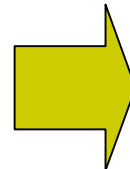
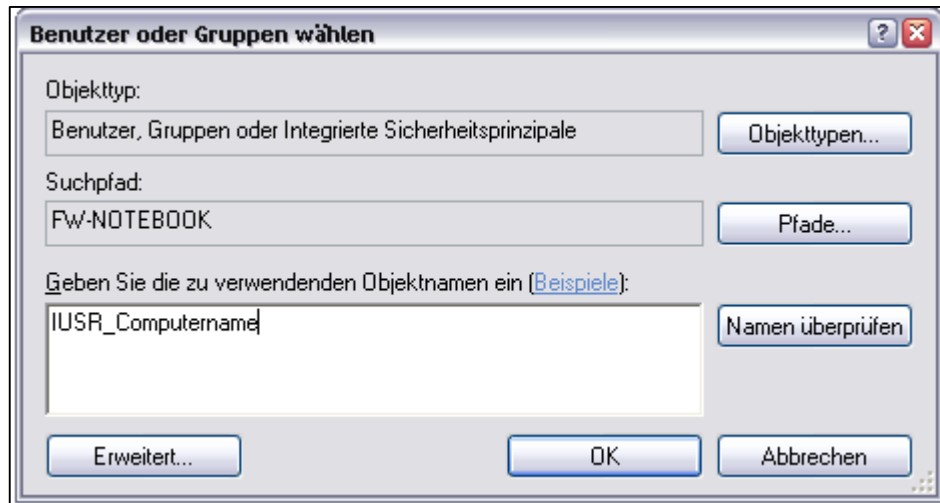
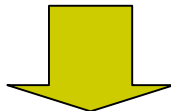
Einfache Dateifreigabe muss deaktiviert sein



# Schreibrechte für die Datenbank



**Tipp:**  
Von vornherein Vollzugriff auf den **Datenbank-ordner** oder **wwwroot** (zur Entwicklungszeit)

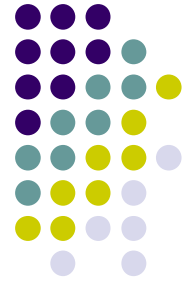


# VBScript – Datum/Zeit

- | Zeit und Datum
  - | NOW – Zeit und Datum aktuell
  - | DATE – nur Datum
  - | TIME – nur Zeit
- | Zerlegen
  - | MONTH(Date) – Zahlenwert des Monats
  - | DAY(Date) – Zahlenwert des Tages
  - | WEEKDAY(Date) – Sonntag = 1
  - | YEAR(Date) - Zahlenwert des Jahres
  - | HOUR(time)
  - | MINUTE(time)
  - | SECOND(time)



# FileSystemObject (ActiveX)



- | Erlaubt Zugriff auf das Dateisystem des Servers
- | Erstellen, Verarbeiten von Dateien und Verzeichnissen
- | Keine Datenbank notwendig
- | Anwendungsbeispiele:
  - | Logdatei
  - | Formulardaten ablegen - bei kleineren Anwendungen keine Datenbank notwendig (z.B. Gästebuch, Forum...)
  - | Datei-Upload

# FSO - Syntax

<%

```
Set fso = Server.CreateObject("Scripting.FileSystemObject")
```

```
Set MeineDatei = fso.CreateTextFile("c:\testdatei.txt", True)
```

```
    MeineDatei.WriteLine("Dies ist ein Test.")
```

```
MeineDatei.Close
```

```
Set fso = Nothing
```

```
Set MeineDatei = Nothing
```

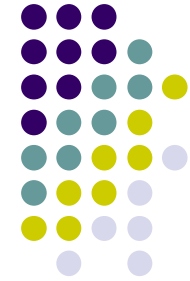
%>



Server.CreateObject	Methode zum Erzeugen des FSO
CreateTextFile	Methode zum <b>Erzeugen</b> eines <b>TextStream</b> -Objektes (Text Stream = leere Textdatei) <b>true</b> – bestehende Datei überschrieben
WriteLine	Schreibt eine Textzeile in das <b>TextStream</b> -Objekt
Close	Schließt die Datei

# TextStream-Objekt

## (Lesen, Schreiben, Anhängen)



*Objekt*.OpenTextFile(*Dateiname*, *Modus*, *Erstellen*)

### **Objekt**

Erforderlich. *Objekt* ist stets der Name eines `FileSystemObject`-Objekts.

### **Dateiname**

Erforderlich. *Zeichenfolgenausdruck*, der die zu öffnende Datei angibt.

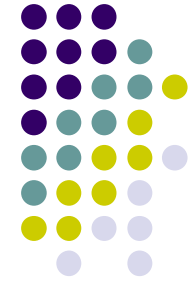
### **Modus**

Optional. Es gibt drei mögliche Konstanten: `ForReading`, `ForWriting` oder `ForAppending`. (siehe nächste Folie)

### **Erstellen**

Optional. Boolescher Wert, der angibt, ob eine neue Datei erstellt werden soll, wenn der angegebene *Dateiname* nicht existiert. `true`, wenn eine neue, `false`, wenn keine neue Datei erstellt werden soll. Falls ausgelassen, wird keine neue Datei erstellt.

# OpenTextFile-Methode



## Modus (Eingabe/Ausgabe)

Konstante	Wert	Beschreibung
ForReading	1	Öffnet eine Datei nur zum Lesen. Schreibzugriffe sind nicht möglich.
ForWriting	2	Öffnet eine Datei zum Schreiben.
ForAppending	8	Öffnet eine Datei und schreibt an das Dateiende.

ASP VBScript

*Beispiel:* Datei zum Anfügen von Text öffnen, Datei erstellen falls nicht vorhanden

```
<% Const ForReading = 1, ForWriting = 2, ForAppending = 8
Dim fso, f
Set fso = Server.CreateObject("Scripting.FileSystemObject")
Set f = fso.OpenTextFile("c:\test.txt", ForAppending, True)
f.Write "Hallo Welt!"
f.Close %>
```

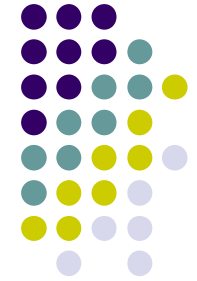
© Wannerer

# TextStream-Objekt - Methoden



Methoden/Eigenschaft	Beschreibung
Close	Schließt die Datei
Read(n)	Gib <i>n</i> Zeichen zurück
ReadAll	Gibt gesamte Datei aus!
ReadLine	Gibt ganze Zeile aus
Write(Text)	Schreibt <i>Text</i> ohne Zeilenumbruch
WriteLine(Text)	Schreibt Zeile ( <i>Text</i> mit Zeilenumbruch)
WriteBlankLines(n)	Schreibt <i>n</i> Zeilenumbrüche
AtEndOfLine	<i>True</i> am Ende einer Zeile (nur lesen)
AtEndOfStream	<i>True</i> am Ende der Datei (nur lesen)

# VBScript - Funktionen



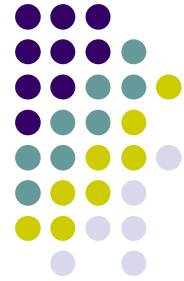
- | INT() – ganzzahliger Wert (nächstkleinere Zahl)
- | FIX() – ganzzahliger Wert (ohne Rundungsregeln)
- | ROUND(Wert,DezStellen) – rundet
- | SGN() – liefert Vorzeichen
- | ISNUMERIC() – prüft, ob numerisch; gibt TRUE oder FALSE zurück
- | RND() – liefert Zufallszahl zw. 0 und 1

# VBS-Zeichenketten-Funktionen



- | LEFT(String,Länge) – linker Teil
- | RIGHT(String,Länge) – rechter Teil
- | MID(String,Beginn, Länge) – mittlerer Teil
- | LEN(String) – liefert Länge
- | Leerzeichen entfernen:
  - | LTRIM() – links entfernen
  - | RTRIM() – rechts entfernen
  - | TRIM() – beidseitig entfernen
- | INSTR(String, Suchstring) – gibt Position von Suchstring zurück
- | UCASE() – Umwandlung in Großbuchstaben
- | LCASE() – Umwandlung in Kleinbuchstaben

# VBS - Bedingte Ausführung



## I IF ... THEN ... ELSE

- I ELSEIF und ELSE sind optional

```
if Zahl = 10 Then
    ...Befehle
elseif Zahl=9 Then
    .... Befehle
else
    .... Befehle
end if
```

## I SELECT CASE

```
Select case land
case 1
    response.write("Österreich")
case 2
    response.write("Italien")
case 3
    response.write("Tschechien")
case else
    response.write("unbekannt")
end select
```

# VBS - Schleifen

- I **DO ... LOOP** UNTIL bzw. WHILE als Eingangs- oder Ausgangsbedingung  
EXIT DO setzt nach LOOP fort

```
DO WHILE not rs.eof  
    ...  
    ...  
LOOP
```

- I **FOR ... NEXT** STEP für Schrittweite  
EXIT FOR setzt nach NEXT fort

```
FOR i=1 TO 10  
    ...  
    ...  
NEXT
```

- I **FOR EACH ... NEXT**  
Bearbeiten einer Kollektion  
EXIT FOR

```
FOR EACH var IN request.form  
    ...  
    ...  
NEXT
```

