

Inhaltsverzeichnis

1	Dreamweaver - Erstellen einer neuen Site:	1
2	Dreamweaver - Erstellen einer PHP-Seite:	2
3	Ausgaben aus dem Script:	2
4	Variablen:	2
5	Rechenoperatoren.....	3
6	Logische Operatoren	3
7	Zahlen formatieren	3
8	Kommentare:	3
9	Bearbeiten von Zeichenketten:	3
10	Umwandlung von HTML-Sonderzeichen	4
11	Zeichenkettenoperatoren	4
12	Vergleichsoperatoren	4
13	Fallunterscheidung mit If:	5
14	Schleifen:	5
15	Arrays:	6
16	Datum und Zeit	7
17	Senden von Formulardaten:	8
18	Daten per GET (Query-String) weitergeben:	8
19	Formularfeld Liste mit Mehrfachauswahl	8
20	Erstellen von Session-Variablen:.....	9
21	Dateiupload:.....	9
22	Datenbankzugriff Access (ODBC):.....	10
23	Einbinden von Dateien (include, require).....	10
24	Datenbankzugriff (MySQL):	11
25	Zusammenfassung wichtiger Funktionen beim Datenbankzugriff (MySQL)	11
26	MySQL-Datenbankzugriff (mittels Dreamweaver PHP-Assistent):	12
27	MySQL-Abfragen/Datensatzgruppen (mittels PHP-Assistent):.....	13
28	Header:.....	13
29	Serververhalten – Datensatz einfügen:	14
30	Serververhalten – Datensatz ändern:	14
31	Serververhalten – Datensatz löschen:	15
32	Serververhalten – Bereich wiederholen:.....	15
33	Serververhalten – Benutzerauthentifizierung:	15

PHP

(Abkürzung für „**PHP: Hypertext Preprocessor**“, ursprünglich „**Personal Home Page Tools**“) ist eine Scriptsprache mit einer an C bzw. C++ angelehnten Syntax, die hauptsächlich zur Erstellung von dynamischen Webseiten oder Webanwendungen verwendet wird.

1 Dreamweaver - Erstellen einer neuen Site:

Der lokale Stammordner einer Site muss sich immer im Ordner ...\\xampp\\htdocs\\ befinden. Beim Testserver ist das Servermodell „PHP MySQL“ auszuwählen, Zugriff: Lokal/Netzwerk. Der Testserver-Ordner entspricht dem lokalen Stammordner, beim URL-Präfix muss ebenfalls der Site-Ordner angegeben werden (hier aber ohne Pfad),

2 Dreamweaver - Erstellen einer PHP-Seite:

Um ein PHP-Script in eine Seite einbauen zu können, muss es sich um eine dynamische Seite handeln. Daher: Datei – Neu – Dynamische Seite – PHP.

3 Ausgaben aus dem Script:

Jedes PHP-Script muss innerhalb des PHP-Tags stehen:

```
<?php      ?>
```

Alle Zeilen (außer if/else und Schleifen) werden mit einem Strichpunkt abgeschlossen. Ausgaben erfolgen mittels des Befehls echo, wobei die auszugebenden Zeichen von Anführungszeichen ("doppel t" oder 'ei nfach') eingeschlossen sein müssen.

Auch HTML-Tags, wie z.B.
, können in den echo-Befehl eingebunden werden.

z.B.:

```
<?php echo "Ausgabe 01 <br />"; ?>
```

4 Variablen:

In PHP beginnt jeder Variablenname mit einem Dollarzeichen, auch die Groß- und Kleinschreibung ist relevant. Einer Variable wird mit = ein Wert zugewiesen, wobei auf der linken Seite die Variable und rechts der gewünschte Wert stehen muss.

z.B.:

```
<?php $Groesse = 185; ?>
```

Der bestehende Wert einer Variablen kann auch überschrieben werden:

z.B.:

```
<?php $Groesse = $Groesse + 10; //$Groesse dann 195 ?>
```

Identisch:

```
<?php $Groesse += 10; ?>
```

Möchte man Variable ausgeben, verwendet man wieder den Befehl echo. Soll die Variable in Kombination mit einem String ausgegeben werden, kann man sie entweder direkt in den String einfügen wenn dieser unter doppelten Anführungszeichen steht (bei assoziativen Arrays wie auch \$_GET[], \$_POST[] usw. müssen zur Begrenzung noch zusätzlich geschweifte Klammern verwendet werden). Bei Verwendung von einfachen Anführungszeichen verknüpft den String und die Variable mit einem Punkt.

z.B.:

```
<?php echo $Groesse; ?>
```

```
<?php echo "Die Größe beträgt $Groesse cm."; ?>
```

```
<?php echo 'Die Größe beträgt ' . $Groesse . ' cm.' ; ?>
```

Assoziative Arrays:

```
<?php echo "Die Anzahl ist: {$next['anzahl']} " ; ?>
```

Um zu prüfen, welcher Datentyp und Inhalt einer Variablen zugewiesen wurde, gibt es den Befehl var_dump. Mit echo wird der Datentyp im Browser ausgegeben.

z.B.:

```
<?php echo var_dump($Groesse); ?>
```

Um zu prüfen, ob eine Variable existiert kann die Funktion isset() verwendet werden. Die Funktion liefert true oder false

z.B.

```
<?php
if (isset($wert)) {
```

```
    echo "vorhanden";
}
```

```
?>
```

5 Rechenoperatoren

Operator	Rechnung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
%	Modulo-Division (Restrechnung)

6 Logische Operatoren

Operator	Wirkung
&&	Logisches Und
and	Logisches Und
	Oder
or	Oder
xor	Entweder oder
!	nicht

7 Zahlen formatieren

`number_format(Zahl, Stellen, Dezimalzeichen, Tausendertrennung)`

8 Kommentare:

Für Anmerkungen innerhalb des PHP-Scripts können Kommentare eingefügt werden. Steht der Kommentar in nur einer Zeile, wird er mit zwei Schrägstrichen eröffnet, ist er allerdings mehrere Zeilen lang, muss er mit `/*` bzw. `*/` eingegrenzt werden.

z.B.:

```
<?php // einzeiliger Kommentar ?>
<?php /* mehrzeiliger
Kommentar */ ?>
```

9 Bearbeiten von Zeichenketten:

Um ein bestimmtes Zeichen in einer String-Variable zu suchen, gibt es die Funktion `strpos()`. In der Klammer steht zunächst die Variable, die durchsucht werden soll, und danach unter Anführungszeichen das gesuchte Zeichen. Bei dieser Funktion wird auch die Groß- und Kleinschreibung berücksichtigt. Verwendet man die Funktion `stripos()`, ist die Groß- und Kleinschreibung irrelevant.

`strpos()` liefert die Position des ersten übereinstimmenden Zeichens von links, `strrpos()` von rechts. `strtolower()` gibt die ganze Zeichenkette in Kleinbuchstaben aus, `strtoupper()` in Großbuchstaben. Die Funktion `trim()` schneidet vor und nach dem String überflüssige Leerzeichen weg, `rtrim()` nur rechts, `ltrim()` nur links.

Mit der Funktion `substr()` können Teile einer Zeichenkette herausgeschnitten werden. In der Funktion steht zunächst der Name der Variable, danach die Position, an der der auszuschneidende Text beginnen soll, und schließlich die Länge des auszuschneidenden Textes. Steht vor der Positions- bzw. Längenangabe ein Minus, wird diese von rechts nach links berechnet.

```
z.B.:
<?php
$text = "Zeichenkette"; Ergebnis:
echo strpos($text, "e"); 1
echo strrpos($text, "e"); 0
echo stripos($text, "T"); 9
echo strtolower($text); zeichenkette
echo strtoupper($text); ZEICHENKETTE
echo trim($text);
echo substr($text, 7, 5); kette
echo substr($text, -6, -4); chen
?>
```

10 Umwandlung von HTML-Sonderzeichen

Manche Zeichen haben in HTML eine besondere Bedeutung. Z. B. erzeugt `
` einen Zeilenumbruch. Um an Stelle des Zeilenumbruchs den HTML-Code auszugeben, muss die Funktion `html special chars()` verwendet werden.

```
z.B.
Code                               Ergebnis auf der HTML-Seite
echo "<br />";                       Zeilenumbruch
echo html special chars("<br />")   <br />
```

11 Zeichenkettenoperatoren

Operator	Wirkung
.	Vereinigungs-Operator
.=	Vereinigungs-Zuweisungs-Operator

```
z.B.:
<?php
$a = "Hallo ";
$b = $a . "Welt!"; // $b enthält jetzt den Text "Hallo Welt!"

$a = "Hallo ";
$a .= "Welt!";    // $a enthält jetzt den Text "Hallo Welt!"
?>
```

12 Vergleichsoperatoren

Folgende **Vergleichsoperatoren** können angewandt werden:

==	gleich	>	größer
!=	ungleich	<=	kleiner gleich
<	kleiner	>=	größer gleich
===	identisch (auch Typ)	<>	ungleich

13 Fallunterscheidung mit If:

Mit der Funktion `if()` kann eine Variable auf ein bestimmtes Kriterium geprüft werden. In der Klammer steht die Bedingung, auf die geprüft werden soll, im `if`-Block die Anweisungen, die geschehen sollen, wenn das Kriterium zutrifft, und im `else`-Block jene, die ausgeführt werden, wenn es nicht zutrifft. Der `if`- bzw. `else`-Block muss immer von geschwungenen Klammern eingeschlossen sein.

Vergleichsoperatoren können angewandt werden:

z.B.:

```
<?php
$zahl = 100;
if ($zahl < 60) {
    echo "Die Zahl ist kleiner als 60. <br>";
}
else {
    echo "Die Zahl ist größer als 60. <br>";
}
?>
```

Mit `else if()` können natürlich auch neue Bedingungen eingeführt werden.

14 Schleifen:

Bei einer Schleife werden Anweisungen immer wieder ausgeführt, solange die Bedingung wahr ist. Man unterscheidet drei verschiedene Arten von Schleifen in PHP:

For-Schleife:

In der Schleife müssen drei Werte eingegeben werden: Zuerst die Initialisierung der Variablen, die geprüft werden soll, anschließend die Bedingung, auf die geprüft wird, und schließlich ein Ausdruck, der angibt, wie die zu prüfende Variable bei jedem Schleifendurchlauf verändert werden soll. Die Schleife läuft so lange, bis die Prüfung der Bedingung falsch ergibt. Die einzelnen Ausdrücke der `FOR`-Anweisung müssen durch Strichpunkte getrennt werden. Die Anweisungen, die mittels der Schleife wiederholt ausgeführt werden sollen, müssen innerhalb einer geschwungenen Klammer stehen.

Der Befehl `break` bewirkt, dass die Ausführung der Schleife sofort beendet wird, egal, ob die Bedingung noch weiterhin wahr wäre oder nicht. `Break` darf nicht direkt in einer Schleife stehen, da sie sonst nie ausgeführt würde, sondern nur in einem `if`-Befehl.

Mit der Anweisung `continue` wird der aktuelle Durchlauf der Schleife übersprungen und mit dem nächsthöherem Zähler fortgefahren. Ebenso wie `break` darf auch `continue` nur in einem `if`-Befehl und nicht direkt auftreten.

z.B.:

```
<?php
for ($i = 0; $i < 10; $i ++) {
    if ($i == 4) {
        continue;
    }
    if ($i == 8) {
        break;
    }
    else {
        echo $i . " <br>";
    }
}
?>
```

while-Schleife:

Bei einer While-Schleife werden Anweisungen ausgeführt, solange die Bedingung wahr ist. Damit die Schleife nicht endlos läuft, sollte man im Anweisungsblock eine Veränderung der zu prüfenden Variablen einfügen.

z.B.:

```
<?php
$i = 10;
while ($i > 0) {
    echo $i . "<br>";
    $i --;
}
?>
```

do-while-Schleife:

Die Do-While-Schleife funktioniert ähnlich wie die While-Schleife, nur wird hier die Bedingung erst nach dem ersten Durchlauf der Schleife geprüft. Das heißt, auch wenn die Bedingung von vornherein falsch war, wird die Schleife trotzdem **einmal ausgeführt**. Hingegen wird bei der While-Schleife zuerst geprüft und dann erst ausgeführt, sodass bei nicht zutreffender Bedingung die Schleife nie ausgeführt wird.

z.B.:

```
<?php
$i = 0;
do {
    echo $i . "<br>";
    $i ++;
} while ($i <= 5);
?>
```

foreach – Schleife

Die foreach-Schleife durchläuft Arrays (siehe nächster Abschnitt)

```
<?php

foreach ($arrayname as $ wert) {
    echo $wert;
}
?>
Zugriff auf den Index
<?php

foreach ($arrayname as $index => $ wert) {
    echo $index . " - " . $wert;
}
?>
```

15 Arrays:

Mithilfe von Arrays können der gleichen Variable mehrere Werte zugewiesen werden. Damit dennoch eine eindeutige Identifizierung möglich ist, wird an die Variable hinten in eckigen Klammern ein Index angehängt. Grundsätzlich gibt es zwei Arten von Arrays:

Arrays mit numerischem Index:

Der numerische Index beginnt immer bei null. Es können in derselben Variablen auch Werte mit verschiedenen Datentypen stehen. Mit der Funktion `array()` können einer Variablen verschiedene

Werte zugeteilt werden, der Index wird automatisch zugewiesen. Möchte man aber einen bestimmten Index mit einem Wert belegen, kann diesem auch direkt ein Wert zugewiesen werden. Eine weitere Möglichkeit, Werte zuzuweisen, ist der Operator =>. Hierbei steht in der Funktion array() zunächst der Index, danach der Operator und zum Schluss der Wert, der zugeordnet werden soll.

z.B.:

```
<?php
$x1 = array("Text", 500, true);           automatische Zuweisung
$x1[3] = 260;                            direkte Zuweisung 4. Element
$x2 = array(0 =>"Text", 1=>500, 2=>>true);  Zuweisung mit Operator =>
?>
```

Assoziative Arrays:

Bei assoziativen Arrays steht anstelle des Index ein String, der eine bestimmte Bedeutung hat. Die Zuweisung erfolgt mit dem Operator =>.

z.B.:

```
<?php
$farben = array("rot" => "#FF0000", "gelb" => "#FFFF00", "blau" => "#0000FF");
?>
```

Funktionen für Arrays:

is_array() prüft, ob eine Variable ein Array ist (gibt true oder false zurück)

in_array() prüft, ob ein Wert in einem Array enthalten ist (gibt true oder false zurück).

count() liefert die Anzahl der Elemente

sort() sortiert ein Array

Die Schleife foreach dient zur Iteration durch Arrays

16 Datum und Zeit

Timestamp = Sekunden

Daran denken: Ein Timestamp ist eine Zeitangabe in Sekunden seit dem 1.1.1970. Wenn man also zwei Zeitangaben als Timestamp hat, kann man diese voneinander subtrahieren und hat die Differenz in Sekunden - die dann noch in die gewünschte Einheit umrechnen und das war's :)

Timestamp erzeugen

Wenn Ihnen nur ein Datum vorliegt, etwa "14.12.2005, 11:42" und die Differenz bis heute berechnen möchten, nutzen Sie mktime() um aus dem Datum einen Timestamp zu erzeugen:

```
<?php
$jetzt = time();
$damals = mktime(11, 42, 00, 12, 14, 2005);
?>
```

Rechnen

Das Rechnen ist nun eine Kleinigkeit. Sie errechnen die Sekunden und daraus dann die jeweils gewünschte Einheit:

```
<?php
$jetzt = time();
$damals = mktime(11, 42, 00, 12, 14, 2005);
$differenz = $jetzt - $damals; // Differenz in Sekunden
$stage = $differenz / 86400; // Differenz in Tagen
```

```
$wochen = $stage / 7 ;           // Di fferenz i n Wochen
?>
```

Kleiner Tipp: Die unschönen Komma-Zahlen können mittels `floor()` oder `round()` passend gerundet werden.

17 Senden von Formulardaten:

Variablen, die über ein Formular gesendet wurden, können aus dem superglobalen Array `$_POST['Feldname']` und `$_GET['Feldname']` abgerufen werden. In den eckigen Klammern muss unter Anführungszeichen der Name des Feldes, in dem sich die Daten befinden, stehen. Weiters kann mit `isset()` geprüft werden, ob überhaupt Daten übertragen wurden.

```
z.B.:
<?php
if (isset($_POST["Eingabe"]))
{
    $meldung="Folgende Daten wurden gesendet: " . $_POST["Eingabe"];
}
else
{
    $meldung = "Es wurden keine Daten gesendet.";
}
?>
```

Da bei Formularen praktisch immer die Schaltfläche mit Namen 'Submit' vorhanden ist, kann mit `$_POST['Submit']` und `$_GET['Submit']` relativ sicher geprüft werden, ob Formulardaten versandt wurden.

18 Daten per GET (Query-String) weitergeben:

Sollen Daten per Query-String mit einem Link mitgeschickt werden, muß man im `<a>`-Tag nach der Zieladresse die gewünschte Information mit einem `?` anhängen.

```
z.B.
<a href="Zielseite.php?ID=<?php echo $Variable; ?>">Link mit Get </a>
```

Auf der nächsten Seite kann die gesendete Information aus dem Array `$_GET[]` abgerufen werden.

```
z.B.
<?php
$gesendeterWert = $_GET["ID"];
?>
```

19 Formularfeld Liste mit Mehrfachauswahl

Soll bei einer Liste Mehrfachauswahl möglich sein (`multiple="multiple"`) muss der Name des Feldes ein Array (leere eckige Klammern anfügen!) sein, damit man die gewählten Daten mit PHP verarbeiten kann.

```
z.B.:
<select name ="topten[]" size="10" multiple="multiple">
```

Im script können dann z.B. die Funktionen `in_array()` und `is_array()` angewendet werden.

20 Erstellen von Session-Variablen:

Neue Sitzung starten oder bestehende wiederaufnehmen (am Beginn eines scripts)

```
<?php
if(!isset($_SESSION))
{
    session_start();
}
?>
```

Session-Variablen werden mit dem Befehl `$_SESSION["NameVariable"]` erstellt. Dazu muss die Session gestartet sein. Dies funktioniert mit `session_start()`; Vorteil von Session-Variablen: Sie werden beim Aufrufen der nächsten Seite nicht gelöscht, sondern bleiben gespeichert, solange der Browser geöffnet ist oder die Variable per PHP-Befehl gelöscht wird (Syntax: `unset($_SESSION["Name"]);`)

z.B.:


```
<?php
if(!isset($_SESSION["Session"]))
{
    session_start();
}
$_SESSION["Benutzername"]=$Benutzername;
?>
```

Die Funktion `session_id()` liefert einen eindeutigen String zur Identifikation der Sitzung.

z.B.:

```
<?php
$_SESSION['Sitzungsnummer'] = session_id();
?>
```

21 Dateiupload:

Um Dateien hochladen zu können, muss im Formular ein Dateifeld (hier mit Namen 'file' eingefügt werden (Registerkarte Formular, Symbol ). Der Original-Dateiname wird mit `$_FILES["file"]["name"]` übergeben.

Weitere Parameter:

<code>\$_FILES["file"]["tmp_name"]</code>	temporärer Name der hochgeladenen Datei
<code>\$_FILES["file"]["size"]</code>	Dateigröße
<code>\$_FILES["file"]["type"]</code>	MIME-Typ

Hochgeladenen Dateien landen in einem Ordner, der in der Datei `php.ini` festgelegt ist.

Der Zielordner muss in einer Variable angegeben sein. Das Verschieben der temporären, hochgeladenen Datei erfolgt mit folgender Syntax:

```
move_uploaded_file(Temporärer Dateiname, Zielordner + Dateiname);
```

z.B.:

```
<?php
$ordner="../bilder/";
move_uploaded_file($_FILES["file"]["tmp_name"], $ordner . $_FILES["file"]["name"]);
?>
```

Die Funktion `move_uploaded_file()` liefert `true`, wenn die Datei erfolgreich verschoben wurde, sonst `false`.

22 Datenbankzugriff Access (ODBC):

Für eine ODBC-Verbindung muss zunächst in der Systemsteuerung eine ODBC-Datenquelle angelegt werden (Start – Systemsteuerung – Verwaltung – Datenquellen (ODBC) – Registerkarte System-DSN – Hinzufügen – Microsoft Access-Treiber - Namen eingeben und Datenbank auswählen – OK).

Im PHP-Script wird dann eine Verbindung zu dieser ODBC-Datenbank hergestellt. Hierbei weist man einer beliebigen Variable (meist `$odbc_id`) die Funktion `odbc_connect()` zu. In der Funktion steht zunächst der Name der ODBC-Verbindung, danach – falls vorhanden – der User und das Passwort für die Datenbank.

```
z.B.:
<?php
$odbc_id = odbc_connect("odbc_kunden", "", "");
?>
```

Anschließend verfasst man einen SQL-Befehl, mit dem auf die Datenbank zugegriffen wird. Diesen weist man ebenfalls einer Variable zu.

```
z.B.:
<?php
$sql = "SELECT * FROM Kunden";
?>
```

Um den SQL-Befehl ausführen zu können, gibt es die Funktion `odbc_exec()`. In der Klammer steht die Variable, in der die ODBC-Verbindung gespeichert ist, und die Variable mit dem SQL-Befehl.

```
z.B.:
<?php
$res = odbc_exec($odbc_id, $sql);
?>
```

Mit der Funktion `odbc_fetch_array()` wird eine Variable(Array) mit dem jeweils nächsten Datensatz der Tabelle gefüllt. Die einzelnen Felder werden als Array gespeichert. Um alle Datensätze auszugeben, wird dieser Befehl oft in einer Schleife verwendet.

```
z.B.:
<?php
while($next = odbc_fetch_array($res))
{
    echo "<p>". $next["Vorname"]. ", ". $next["Zuname"]. "</p>";
}
?>
```

Wenn die Verbindung nicht mehr benötigt wird, kann man mit folgenden Befehlen den Speicher wieder freigeben: `odbc_free_result()` für die SQL-Variable und `odbc_close()` für die Datenbankverbindung.

```
z.B.:
<?php
odbc_free_result($res);
odbc_close($odbc_id);
?>
```

23 Einbinden von Dateien (include, require)

Die Anweisung `include(Datei)` bindet die angegebene Datei ein und wertet sie aus. Dadurch ist eine Modularisierung möglich.

Oft verwendete scripts (z.B. functions etc.) werden in eigene Dateien ausgelagert und bei Bedarf eingebunden.

z.B.

```
<?php
include("connection.php");
?>
```

Die Anweisung `include_once()` verhindert die mehrfache Einbindung in einem Dokument. Die Anweisungen `require()` und `require_once()` unterscheiden sich von `include()` beim Auftreten von Fehlern

24 Datenbankzugriff (MySQL):

Um auf eine MySQL-Datenbank zugreifen zu können, muss diese vorhanden sein (erstellen in phpMyAdmin oder mit SQL-Anweisungen)

Syntax für die Verbindung mit dem MySQL-Server:

```
mysql_connect(„hostname“, „user“, „password“);
```

Die Datenbank selbst wird mit dem Befehl

```
mysql_select_db(„Datenbank“, „Verbindung“);
```

ausgewählt.

z.B.:

```
<?php
$Verbindung = mysql_connect(„localhost“, „root“, „“);
$Datenbank = mysql_select_db(„Schueler“, $Verbindung);
?>
```

Das Ausführen der SQL-Befehle erfolgt mit der Anweisung `mysql_query("SQL-Befehl");` Der ausgelesene Datensatz kann mithilfe des Befehls `mysql_fetch_array(Ergebnis d. Abfrage)` als Array geliefert werden.

z.B.:

```
<?php
$sql = "SELECT * FROM Schueler;";
$ergebnis = mysql_query($sql);
while($next = mysql_fetch_array($ergebnis))
{
    echo "<p>". $next["Vorname"]. ", ". $next["Zuname"]. "</p>";
}
?>
```

Der Speicher wird mittels `mysql_free_result($Ergebnis);` wieder freigegeben. Die Verbindung kann mit `mysql_close($verbindung);` geschlossen werden.

```
<?php
mysql_free_result($rs_Schueler);
mysql_close($dw_Schueler);
?>
```

25 Zusammenfassung wichtiger Funktionen beim Datenbankzugriff (MySQL)

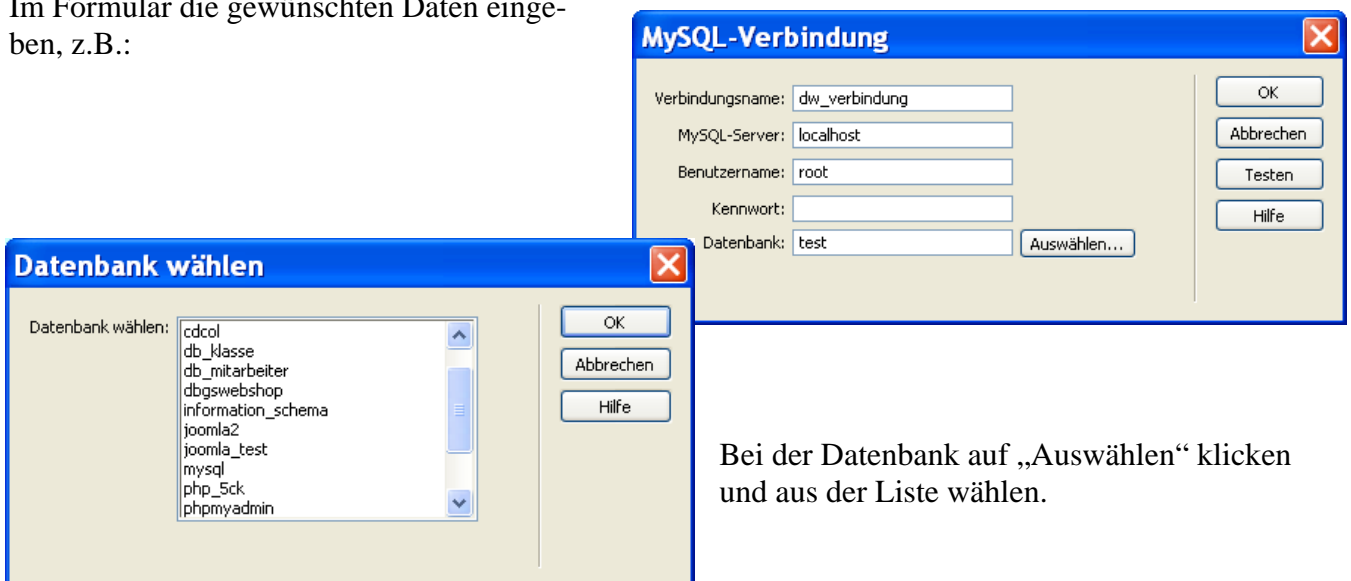
MySQL Beispieldaten

Beispiele für Datenbank: db_Personal, Tabelle: tbl_Personen (Felder: ID, Zuname, Vorname, Geburtsdatum, Groesse)

Aufgabe	PHP- Funktion	Beispielcode
Verbindung zur Datenbank herstellen	mysql_connect(Host, User, Passwort)	<code>\$conn=mysql_connect('localhost','root','');</code>
Datenbank auswählen	mysql_select_db(db_name, Kennung)	<code>mysql_select_db("db_Personal", \$conn);</code>
SQL-String		<code>\$sql = ' SELECT * FROM tbl_Personen ';</code>
SQL-Statement ausführen (bei SELECT-Anweisungen in einer Variablen [z. B. \$res] ablegen)	mysql_query(SQL-String, Link-Identifizier)	<code>mysql_query(\$sql,\$conn);</code> <code>\$res = mysql_query(\$sql,\$conn);</code>
Abfrageergebnis auswerten die Funktion liefert einen Datensatz in Form eines Arrays (z.B. \$next). Die Funktion liefert false, wenn keine Zeilen mehr vorhanden sind	mysql_fetch_array()	<code>\$next=mysql_fetch_array(\$res);</code>
Ausgabe aus dem Array (assoziativ)		<code>echo \$next["Zuname"];</code>
Ausgabe mehrere Zeilen mit einer Schleife. Die Schleife wird durchlaufen, solange das Array (z.B. \$next) Werte enthält		<code>while(\$next=mysql_fetch_array(\$res))</code> <code>{</code> <code> echo \$next["ID"]." " . \$next["Zuname"] . " " . "
;</code> <code>}</code>
Anzahl der Datensätze in der Abfrage	mysql_num_rows()	<code>\$anzahl = mysql_num_rows(\$res);</code>
Letzte ID bei INSERT-Anweisungen (auto_increment-Feld)	mysql_insert_id()	<code>\$letzte_id = mysql_insert_id()</code>

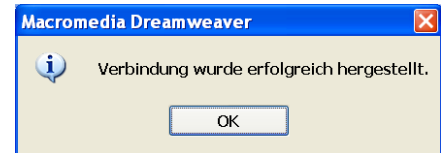
26 MySQL-Datenbankzugriff (mittels Dreamweaver PHP-Assistent):

Fenster Anwendungen – Registerkarte Datenbanken – auf „+“ klicken - MySQL-Verbindung wählen. Im Formular die gewünschten Daten eingeben, z.B.:



Bei der Datenbank auf „Auswählen“ klicken und aus der Liste wählen.

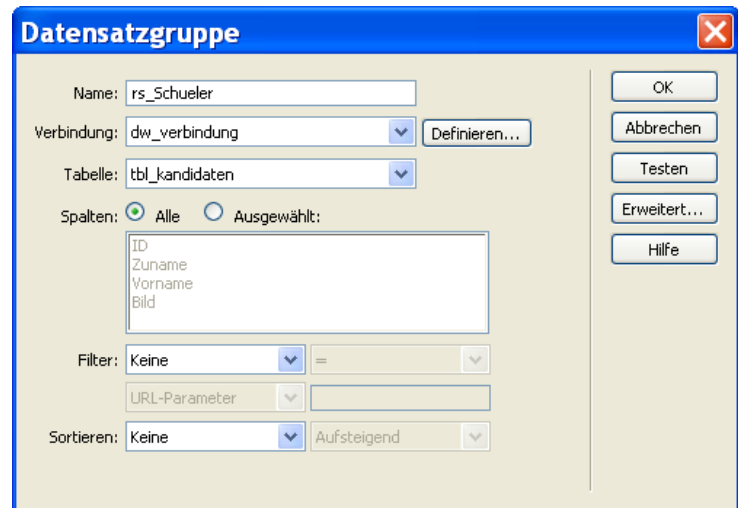
Mit der Schaltfläche „Testen“ kann probiert werden, ob die Verbindung hergestellt werden kann. Im Idealfall erscheint dann folgende Meldung:



27 MySQL-Abfragen/Datensatzgruppen (mittels PHP-Assistent):

Fenster Anwendungen – Registerkarte Bindungen – auf „+“ klicken – „Datensatzgruppe (Abfrage)“ wählen.

Namen vergeben, Verbindung wählen (mit der Schaltfläche „Definieren“ kann auch eine neue Verbindung erstellt werden), Tabelle auswählen, ggf. Filter oder Sortierung einstellen.



Filter:

URL-Parameter = \$_GET[]

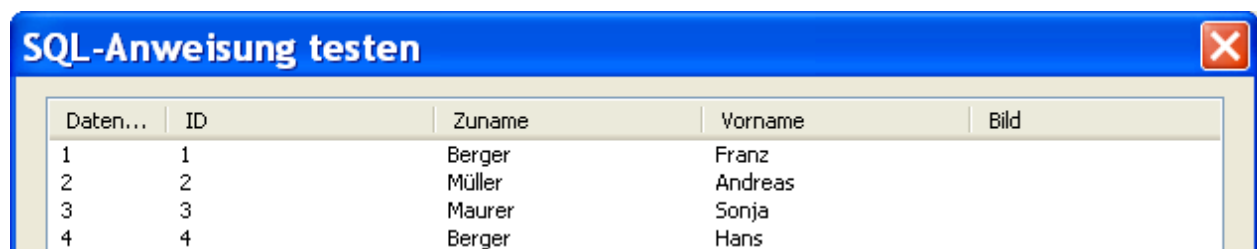
Formularvariable = \$_POST[]

Sitzungsvariable = \$_SESSION[]

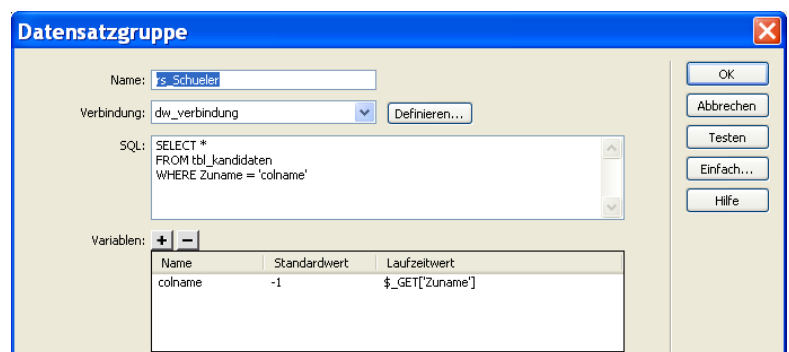
Servervariable = \$_SERVER[]

Eingegebener Wert = \$Variable

Mit der Schaltfläche Testen kann man sich das Ergebnis der Abfrage anzeigen lassen.



In der erweiterten Ansicht können komplexere SQL-Befehle manuell erstellt werden.



28 Header:

Header dienen zum Aufrufen einer anderen Seite, z.B. nach Ausführen eines SQL-Befehles. Syntax: header(„location: aufzurufendeSeite.php“);

z.B.:

```
<?php
```

```
header("location: index.php");
```

```
?>
```

Achtung:

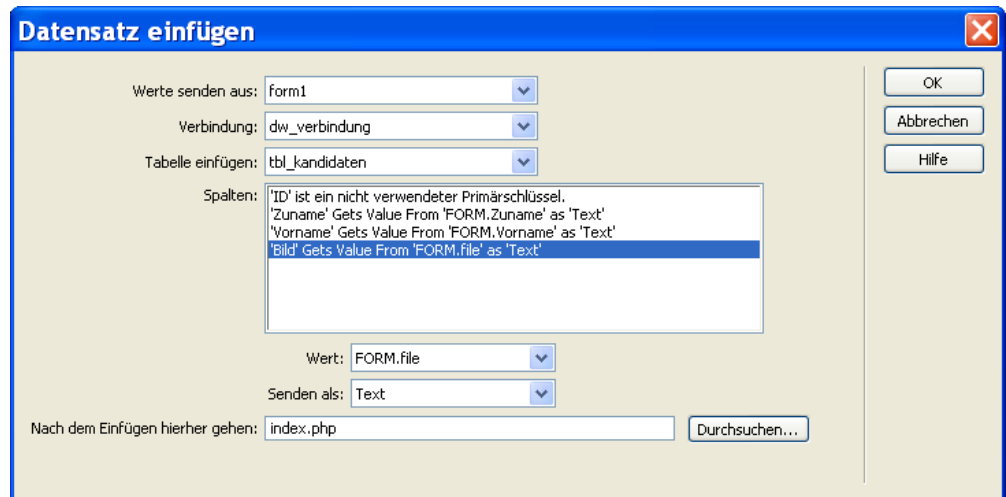
Im Script darf vor der Funktion header() KEINE Ausgabe (z.B. mit echo oder HTML-Tags) erfolgen! Ein oft unbeabsichtigtes Leerzeichen kann die Funktion schon behindern.

29 Serververhalten – Datensatz einfügen:

Formular erstellen mit Feldern, die in der Tabelle gespeichert werden sollen
 Form-Tag markieren, Fenster Anwendungen – Registerkarte Serververhalten – auf „+“ klicken – Datensatz einfügen.

Verbindung + Tabelle wählen, wenn den Spalten nicht automatisch die richtigen Formularfelder zugeordnet werden, Zeile markieren und unten im Drop-Down Feld „Wert“ das richtige Feld auswählen.

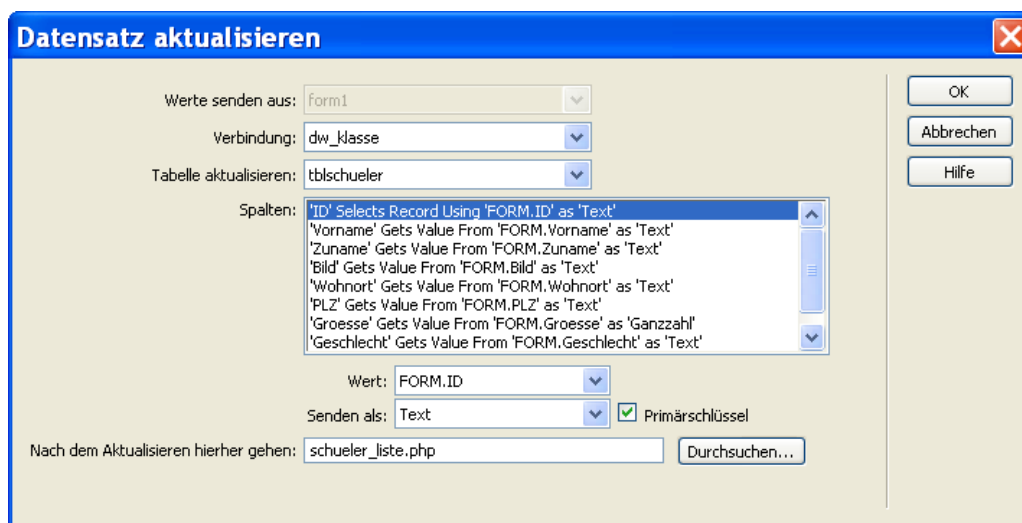
Auf Durchsuchen klicken und jene Seite auswählen, die nach dem Einfügen des Datensatzes angezeigt werden soll, OK.



30 Serververhalten – Datensatz ändern:

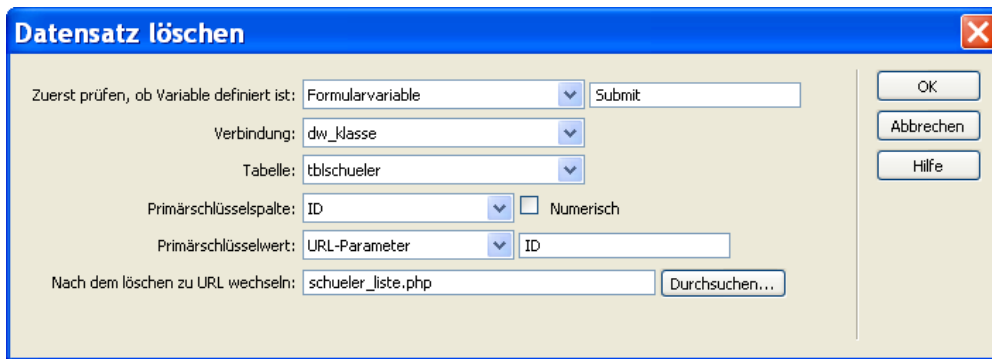
Abfrage/Recordset erstellen, in der der zu ändernde Datensatz aufgerufen wird
 Formular erstellen, bei den Textfeldern im Value-Attribut das Ergebnis der Datensatzgruppe ausgeben, z.B. `<input type="text" name="Vorname" value="<?php echo $row_rs_aendern[' Vorname']; ?>" size="32">`

Formular-Tag markieren, Serververhalten „Datensatz aktualisieren“ aufrufen
 Verbindung+Tabelle wählen, Werte kontrollieren, ggf. manuell aus Drop-Down-Feld zuweisen, Seite, die nach Änderung aufgerufen werden soll, auswählen, OK.



31 Serververhalten – Datensatz löschen:

Abfrage/Recordset erstellen, in der der zu ändernde Datensatz aufgerufen wird
 Formular erstellen, bei den Textfeldern im Value-Attribut das Ergebnis der Datensatzgruppe ausgeben, z.B. `<input type="text" name="Vorname" value="<?php echo $row_rs_aendern['Vorname']; ?>" size="32">`
 Formular-Tag markieren, Serververhalten „Datensatz löschen“ aufrufen
 Bei Variable „Formularvariable“ und „Submit“ eingeben → Datensatz wird erst nach Klick auf die Schaltfläche gelöscht und nicht schon beim Aufrufen der Seite
 Verbindung, Tabelle, Primärschlüssel und aufzurufende Seite wählen – OK.



Tipp:

Der Assistent "Aktualisierungsformular..." Erstellt ein Formular und das zugehörige Serververhalten in einem Zug.

32 Serververhalten – Bereich wiederholen:

Wenn eine Abfrage mehrere Ergebnisse liefert, kann mithilfe des Serververhaltens „Bereich wiederholen“ die Zeile (z.B. Tabellenzeile) wiederholt (mit jeweils dem nächsten Datensatz) angezeigt werden.



Zunächst muss eine Datensatzgruppe erstellt werden, dann Absatz/Tabellenzeile markieren, Serververhalten – Bereich wiederholen.

Datensatzgruppe wählen, einstellen, ob alle Datensätze gleichzeitig angezeigt werden sollen oder immer nur ein paar (z.B. nur 10 Datensätze auf einmal).

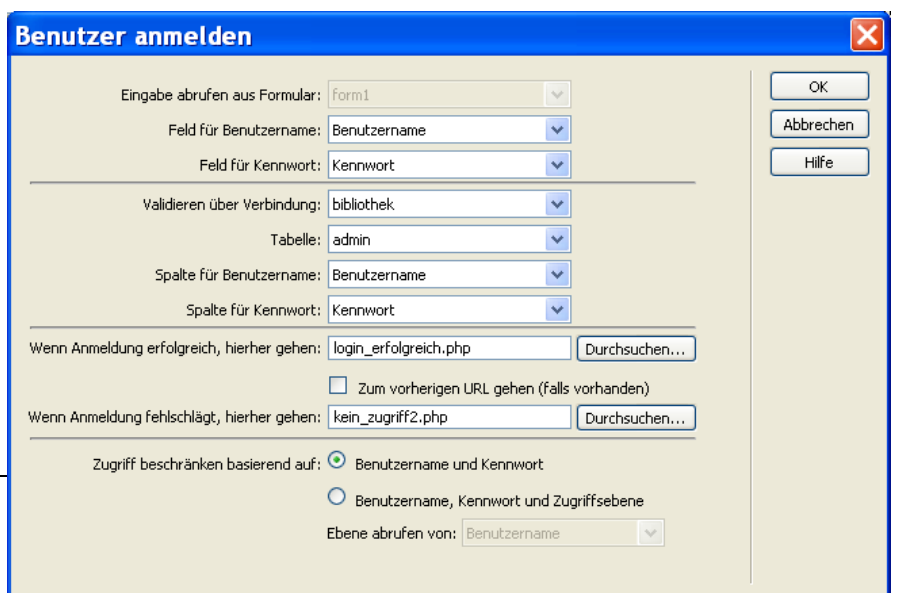
Hat man die Option gewählt, nur einen Teil der Datensätze anzeigen zu lassen, muss man eine Navigationsleiste einfügen, um auch die folgenden Seiten aufrufen zu können.

Registerkarte Anwendungen - anklicken, Datensatzgruppe und Text (z.B. „nächste Seite“) oder Bilder (z.B.) wählen.

33 Serververhalten – Benutzerauthentifizierung:

1. Benutzer anmelden:

Dient zum Identifizieren eines Benutzers und zum Erzeugen einer Session-Variable, in der der Benutzername des angemeldeten Users gespeichert wird. Zunächst muß ein Formular mit



Textfeldern für den Benutzernamen und das Passwort erstellt werden. Diese müssen in einer Tabelle gespeichert sein. Danach Formular-Tag markieren - Serververhalten Benutzerauthentifizierung – Benutzer anmelden. Felder, Verbindung, Tabelle und Spalten auswählen, Seiten angeben, die bei erfolgreicher/nicht erfolgreicher Anmeldung aufgerufen werden sollen, Optionsfeld „Benutzername und Kennwort“ oder „Benutzername, Kennwort und Zugriffsebene“ (nur wenn Ebene in der Tabelle gespeichert ist) wählen, OK.

2. Benutzer abmelden:

Zum Abmelden eines Benutzers zuerst einen Link erstellen, den man anklicken muß, um das Logout-Ereignis aufzurufen, oder eine Seite, bei deren Aufrufen der Benutzer abgemeldet wird, dann Serververhalten Benutzerauthentifizierung – Benutzer abmelden. Gewünschte Option und Seite, die nach dem Logout angezeigt werden soll, auswählen.

3. Zugriff auf Seiten beschränken:

Dieses Serververhalten dient dazu, dass Seiten eines internen Bereiches, für den eine Anmeldung erforderlich ist, nicht direkt aufgerufen werden können, wenn der Benutzer nicht angemeldet ist. Serververhalten Benutzerauthentifizierung – Zugriff auf Seite beschränken – Benutzername und Kennwort (evtl. Zugriffsebene) anklicken, Seite wählen, die bei verweigertem Zugriff angezeigt werden soll.