Online PHP Einsteigerkurs

Inhaltsverzeichnis

1. Einfürung

PHP (Akronym für “PHP: Hypertext Preprocessor”) ist eine weit verbreitete und für den allgemeinen Gebrauch bestimmte Open Source Skriptsprache, die speziell für die Webprogrammierung geeignet ist, und z.B. in HTML eingebettet werden kann.

Was PHP von clientseitigen Sprachen wie JavaScript unterscheidet, ist dass der Code auf dem Server ausgeführt wird. Der Besucher sieht also den PHP-Quelltext nicht, sondern nur die jeweiligen Ausgaben (z.B. HTML, XML, PDF usw).

In diesem Kurs geht es nicht darum wie man es machen könnte, sondern wie man es machen sollte! Alternative Lösungswege und Schreibweisen werden absichtlich nicht behandelt!

Wir versuchen unsere Beispiele so kurz wie möglich zu halten. Weitere Informationen und Beispiele zu den einzelnen Kapiteln finden Sie jeweils auf www.php.net/manual/de/. Wir verwenden **PHP5**!

2. Installation

Da PHP eine serverbasierte Skriptsprache ist, benötigen Sie einen Webserver. **XAMPP** ist eine Distribution von Apache, MySQL, PHP und Perl, die es ermöglicht diese Programme auf sehr einfache Weise auf Ihrem PC zu installieren. Bitte klicken Sie auf das XAMPP-Logo und laden Sie sich die entsprechende Installationsdatei herunter.

Als Beispiel werden wir hier die Installation unter Windows näher beschreiben. Nachdem Sie die aktuellste **Installer-Version** von **XAMPP Windows** heruntergeladen haben, doppelklicken Sie die EXE-Datei und folgen Sie dem Installations-Wizard. Als Installationspfad wählen wir in unserem Beispiel das Verzeichnis **C:/Programme/**

Öffnen Sie nach der Installation die Datei **C:/Programme/xamp/apache/conf/httpd.conf** mit einem einfachen Text-Editor (z.B. Notepad). Aktivieren Sie das Modul **mod_rewrite** indem Sie auf

der Zeile

```
#LoadModule rewrite_module modules/mod_rewrite.so
```

das **#** am Anfang entfernen. Zu diesem Modul kommen wir später noch.

In der gleichen Datei passen wir als nächstes passen wir den **Document Root** an. Der Document Root ist das Basisverzeichnis für alle unsere Webprojekte, die wir in Zukunft haben werden. Wir setzen ihn in unserem Beispiel auf **C:/php** indem wir die folgenden Zeilen editieren.

```
#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "C:/php"

...

#
# Note that from this point forward you must specifically allow
# particular features to be enabled - so if something's not working as
# you might expect, make sure that you have specifically enabled it
# below.
#
#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "C:/php">
```

Speichern Sie und schliessen Sie diese Datei C:/Programme/xamp/apache/conf/httpd.conf.

Öffnen Sie auch die Datei **C:/Programme/xamp/apache/bin/php.ini** mit einem Text-Editor und ändern Sie die folgende Zeile von **On** auf **Off**:

```
; Magic quotes for incoming GET/POST/Cookie data.
magic_quotes_gpc = Off
```

In der php.ini befinden sich auch alle Extensions, die Sie ganz einfach freischalten können. Um z.B. die Extension cURL zu aktivieren, entfernen Sie einfach das Semikolon **;** am Anfang der folgenden Zeile:

```
;extension=php_curl.dll
```

Beachten Sie, dass alle diese Extensions nicht auf jedem Webserver aktiviert sein könnten!

Erstellen Sie eine Textdatei mit Ihrem PHP-Editor mit dem folgenden Inhalt:

```
<?php
phpinfo();
?>
```

Speichern Sie diese unter dem Pfad **C:/php/phpinfo.php**

Starten Sie das **XAMPP Control Panel** und **starten** Sie den **Apache**. Öffnen Sie anschliessend Ihren Browser und geben Sie folgende URI ein: **http://localhost/phpinfo.php** Nun wird Ihnen Ihre PHP Konfiguration als Ausgabe angezeigt:

System	Windows NT [redacted] 5.1 build 2600
Build Date	Nov 29 2005 01:14:17
Configure Command	cscript /nologo configure.js "--enable-snapshot-build"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Programme\xampp\apache\bin\php.ini
PHP API	20041225
PHP Extension	20050922
Zend Extension	220051025
Debug Build	no
Thread Safety	enabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	php, file, http, ftp, compress.zlib, https, ftps
Registered Stream Socket Transports	tcp, udp, ssl, sslv3, sslv2, tls
Registered Stream Filters	convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, zlib.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v2.1.0-dev, Copyright (c) 1998-2005 Zend Technologies

Powered By Zend Engine 2

Wenn dies der Fall ist, haben Sie die **Installation erfolgreich abgeschlossen**

3. Grundlagen Theorie

In diesem Abschnitt lernen Sie die grundlegendsten Bausteine von PHP kennen. Ohne diese kommt kein PHP-Skript aus. Man kann nicht lange genug betonen, dass diese das A und O der Programmierung mit PHP bilden. Deshalb sollten sie auch nachträglich immer wieder detailliert geübt werden! Dieser Teil muss praktisch auswändig gelernt und verstanden werden!

Bitte speichern Sie alle folgenden Übungsdateien in einem Ordner **C:/php/grundlagen/theorie/** damit Sie die Dateien unter **http://localhost/grundlagen/theorie/** testen können.

3.1. Das erste PHP-Skript

Starten und beenden Sie PHP immer mit den folgenden Tags: **<?php** und **?>** Als Beispiel betten wir nun eine einfache PHP-Ausgabe in ein bestehendes HTML-Dokument ein:

```
<html>
<head>
<title>Das erstes PHP-Skript</title>
</head>
<body>
<?php echo 'Hallo Welt!';
?>
</body>
</html>
```

Speichern Sie dieses Skript unter dem Namen **C:/php/grundlagen/theorie/hallo_welt.php** und rufen Sie das Skript mit Ihrem Browser unter der URI **http://localhost/grundlagen/theorie/hallo_welt.php** auf und betrachten Sie den Quelltext der Seite.

3.2 Error Reporting / Debugging

Damit uns auch alle PHP-Fehlermeldungen angezeigt werden, setzen wir die folgenden Zeilen bei allen kommenden Codeabschnitten jeweils auf die **erste PHP-Zeile**.

```
<?php
error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);
?>
```

Beachten Sie, dass wir diese Zeilen, der Übersichtlichkeit zuliebe, in unseren Übungen weglassen, Sie sie aber **immer stehen lassen** sollten!

Um Fehler aufzuspühren, gehen Sie wie folgt vor:

1. Man bemerkt, dass ein Skript nicht das tut, was es soll.
2. Man versucht, die Stelle die daran Schuld sein kann, schonmal einzugrenzen. Falls dies nicht geht, wird das komplette Skript als fehlerhaft angesehen.
3. An den Anfang des Scriptes schreibt man die oben erwähnten zwei Zeilen.
4. Error-Variablen und -Funktionen wie mysql_error(), $_FILES['datei']['error'] usw. auswerten.
5. An markanten Stellen im Skript lässt man sich wichtige Variableninhalte ausgeben und ggf. auch in bedingten Anweisungen eine kurze Ausgabe machen, um zu überprüfen, welche Bedingung ausgeführt wurde (oder auch nicht).
6. Schritt 4 wird so lange wiederholt, bis Unstimmigkeiten im Skript auffallen.
7. Damit hat man das Problem (Unstimmigkeit) gefunden und kann, mit Hilfe der Links, versuchen diese zu beheben.
8. Geben Sie die Fehlermeldung auch bei Google ein!
9. Lässt sich das konkrete Problem trotzdem nicht beheben, kann man in Foren um Rat fragen.
10. Das Programm läuft und man kann die Debug-Ausgaben wieder entfernen.

3.3 Variablen (“skalare Typen”)

In Variablen können Werte verschiedenster Art gespeichert werden. In PHP werden alle Variablen beginnend mit einem Dollarzeichen $ definiert.

```
<?php
$variable = 'Hallo Welt!';
echo $variable;
?>
```

Dieses Beispiel erzielt die exakt gleiche Ausgabe wie das “erste PHP-Skript”.

Variablen können natürlich auch andere Typen als Werte enthalten, wie das folgende Beispiel zeigt:

```php
<?php
// Typ String / Zeichenkette
$var = 'Hier werden <strong>HTML Ausgaben</strong> bespeichert.';

// Typ Integer
$var = 100;
$var = -53;

// Typ Float / Fliesskommazahl
$var = 14.547;
$var = -0.0000156;

// Typ Boolean
$var = true;
$var = false;

// Spezieller Typ NULL (Undefiniert)
$var = NULL;
?>
```

Da Strings mit einfachen Anführungszeichen gestartet und beenden werden, kann es bei diesen Zeichen **im** String selbst zu Problemen führen. Dies kann mit einem Backslash verhindert werden ("scapen"):

```php
<?php
$var = 'Heute ist\'s kalt.';
echo $var; // Heute ist's kalt.
?>
```

Strings können zwar auch in doppelten Anführungszeichen stehen, ist aber zu vermeiden, da dort enthaltene Variablen mitgeparst werden.

Variablen können mit den Zeichenketten-Operatoren **.** und **.=** beliebig verkettet werden:

```php
<?php
$name = 'Peter';
$groesse = 185;
$ausgabe = 'Ich heisse ' . $name . ' und bin ' . $groesse . ' cm gross.<br />';
$ausgabe .= 'Und wie gross bist Du?<br />';
echo $ausgabe;
?>
```

Um mehrzeilige Strings zu speichern, gibt es zwei Möglichkeiten:

```php
<?php
$varianteA = 'Das ist die erste Zeile.
und das hier die zweite Zeile';

$varianteB = 'Das ist die erste Zeile' . "\n";
$varianteB .= 'und das hier die zweite Zeile';

// Newlines in <br /> umwandeln und ausgeben
echo nl2br($varianteA);
echo '<hr />';
echo nl2br($varianteB);
?>
```

Variablen können Ihre Werte auch ändern oder verlieren:

```php
<?php
$nameA = 'Peter';
$nameB = 'Rudolf';

// Wert einer anderen Variable uebernehmen
$nameC = $nameB;

// Variable leeren
$nameB = NULL;

// Variable loeschen
unset($nameB);
```

```
// Neuer Wert => Neuer Typ
$nameA = 500.06;
?>
```

Variablen-Namen werden im sogenannten "CamelCase" und geschrieben. Hier einige Beispiele:

```
<?php
$camelCaseLooksLikeThis = 'Very nice!';
$dasIstEinText = 'Willkommen';
$anzahlTiereImGarten = 5;
$grueneFarbe = '#00FF00';
?>
```

Beachten Sie, dass all diese Namen im professionellen Einsatz in **Englisch** geschrieben werden sollten. Wir verwenden hier nur Deutsch, um einem besseren Verständnis unserer Übungen beizutragen 😁

3.4. Typ-Umwandlung

Manchmal müssen Variablen in einen bestimmten Typ umgewandelt werden. Hier einige wichtigen Beispiele:

```
<?php
$string = '456';
$integer = (int) $string; // 456
$boolean = (bool) $integer; // true

$string = '-456.64';
$integer = (int) $string; // -456
$boolean = (bool) $integer; // true

$string = 'abc';
$integer = (int) $string; // 0
$boolean = (bool) $integer; // false

$string = '123abc';
$integer = (int) $string; // 123

$string = 'abc123';
$integer = (int) $string; // 0

$float = 2.6786;
$integer = (int) $float; // 2
?>
```

Informationen über eine Variable erhalten Sie folgendermassen:

```
<?php
var_dump($variable);
?>
```

3.5. String-Funktionen

Strings müssen sehr oft bearbeitet werden. Die Funktionsreferenz für Strings finden Sie unter www.php.net/manual/de/ref.strings.php. Hier ein paar gängige Beispiele:

```
<?php
// Original
$string = 'Mäuse regieren die Welt.';

// M&auml;use regieren die Welt.
$htmlString = htmlentities($string);
echo $htmlString . '<br />';

// Mäuse regieren die Welt.
echo html_entity_decode($htmlString) . '<br />';

// Mäuse regieren die Kanalisation.
echo str_replace('Welt', 'Kanalisation', $string) . '<br />';
```

```php
// MÄUSE REGIEREN DIE WELT.
echo strtoupper($string) . '<br />';

// mäuse regieren die welt.
echo strtolower($string) . '<br />';

// 24
echo strlen($string) . '<br />';
?>
```

Wie Sie diese Funktionen finden und anwenden erklären wir Ihnen im nächsten Kapitel.

3.6. Funktionsreferenz von php.net

Die Funktionsreferenz auf php.net ist sehr umfangreich und für Einsteiger nicht ganz einfach zu verstehen. Deshalb werden wir die Anwendung des Manuals mit einem kleinen Beispiel erklären: Nehmen wir an, wir möchten einen Namen rückwärts schreiben lassen, sprich aus "Zottelbart" wird "Trablettoz". Wie findet man nun so eine Funktion? Dazu gibt es verschiedene Ansätze. Eine Möglichkeit wäre folgende:

1. F: Was ist "Zottelbart" für ein Variablen-Typ?
   A: String
2. F: Wir brauchen eine String-Funktion. Wo findet man diese?
   A: In der Funktionsreferenz im Kapitel CLXI. String-Funktionen im Abschnitt Inhaltsverzeichnis.
3. F: So viele Funktionen, welche ist es nun?
   A: Überlegen Sie sich, welche anderen Wörter es für "rückwärts schreiben" noch geben könnte.
4. F: Mir fällt "umkehren" und "reverse" ein. Und jetzt?
   A: Suchen Sie nach diesen Wörtern auf der Seite. Spielen Sie auch mit Abkürzungen und Wortspielen wie "kehrt", "rück" oder "rev". Gefunden?
5. F: Ja, die Funktion heisst **strrev** oder?
   A: Korrekt! Klicken Sie nun auf den Link strrev. Sie sehen folgenes:

strrev Funktionsname

(PHP 4, PHP 5) In diesen PHP-Versionen ist die Funktion verfügbar

strrev — Kehrt einen String um Kurzbeschreibung

Beschreibung

1. 2. 3. 4.
string **strrev** (string $string)

1. Rückgabewert 3. Parameter Typ
2. Funktionsname 4. Parameter Variable

Gibt *string* in umgekehrter Reihenfolge zurück. Detaillierte Beschreibung

Nun wissen wir, dass die Funktion strrev **1 Parameter vom Typ String erwartet** und wiederum einen **String zurückliefert**. Das obige Schema ist die Grundlage der Funktionsreferenz und gilt für alle Funktionen.

```php
<?php
$string = 'Zottelbart';
echo strrev($string);
?>
```

Es gibt auch Funktionen mit **optionalen Parametern**. Bsp htmlentities:

string **htmlentities** (string $string [, int $quote_style [, string $charset]])

Die 2 Parameter, die in eckigen Klammern [] stehen, sind optional. Die Funktion kann auch ohne deren Übergabe verwendet werden. In diesem Beispiel ist es auch möglich, den Parameter *$quote_style* mit zu übergeben und *$charset* wegzulassen. Es ist aber **nicht** möglich, nur *$charset* zu übergeben und *$quote_style* wegzulassen!

Es liegt nun an Ihnen, diese Vorgehensweise zu verstehen und ausgiebig zu üben 😁

3.7. Mathematik

Mit PHP werden Sie ziemlich oft rechnen und zählen müssen. Hier einige Beispiele:

```php
<?php
$intA = 10;
$intB = 66;
$summe = $intA + $intB;
```

```
echo 'Die Summe von ' . $intA . ' und ' . $intB . ' ist ' . $summe . '<br />';

$int = 100;
$int -= 55; // 55 subtrahieren
$int += 10; // 10 addieren
$int++; // 1 addieren
$int--; // 1 subtrahieren
echo $int . '<br />';

$negativ = -5123;
$positiv = abs($negativ); // Absolutwert
echo $positiv . '<br />';

$int = (20/100*30+50*2)/2.13; // = 49.76525822...
$int = round($int, 2); // = 49.77
echo $int . '<br />';

$dividend = 27;
$divisor = 4;
$rest = $dividend%$divisor;
echo $rest . '<br />';
?>
```

Machen Sie nun selbst ein paar Beispiele und lernen Sie wie sich PHP bei Mathematischen Funktionen verhält.

3.8. Datum und Zeit

Sehr oft müssen bestimmte Zeitpunkte gespeichert werden oder Inhalte zu einer bestimmten Zeit ein- bzw. ausgeblendet werden. Dies erreicht man mit den Datums- und Zeit-Funktionen. Um diese Funktionen nutzen zu können, müssen wir als erstes unsere Standard-Zeitzone definieren:

```
<?php
date_default_timezone_set('Europe/Berlin');
?>
```

Hier die meist verwendetsten Funktionen:

```
<?php
$datum = date('d.m.Y'); // 01.12.2007
$uhrzeit = date('H:i:s'); // 16:54:45
$zeitstempfel = time(); // 1196467200 (Anzahl Sekunden seit 1. Januar 1970 00:00:00 GMT)
$datum = date('d.m.Y H:i:s', mktime(16, 54, 45, 12, 30, 2007)); // 30.12.2007 16:54:45

// Zeitmessung der Laufzeit
$start = microtime(true);
sleep(2);
$stop = microtime(true);
$laufzeit = $stop - $start;
echo $laufzeit;
?>
```

3.9. if / else / elseif

Eine der wichtigsten Kontrollstrukturen ist das *if*-Konstrukt. Mit diesem lassen sich bestimmte Code Abschnitte bedingt ausführen. Das wichtigste in Kürze:

```
<?php
if (2 > 1) {
    echo '2 ist groesser als 1<br />';
}

// Standardzeitzone
date_default_timezone_set('Europe/Berlin');

$wochentag = date('w');
// Wenn Samstag oder Sonntag
if (6 == $wochentag || 0 == $wochentag) {
    echo 'Juhu, wir haben Wochenende!<br />';
```

```
}
// Wenn Freitag
elseif (5 == $wochentag) {
    echo 'Bald ist Wochenende!<br />';
}
// Alle anderen Tage
else {
    echo 'Arbeiten macht auch Spass!<br />';
}

$stunde = date('G');
// Zwischen 22 Uhr und 6 Uhr
if (22 >= $stunde && 6 < $stunde) {
    echo 'Gute Nacht!<br />';
}
// Zwischen 13 Uhr und 22 Uhr
elseif (22 < $stunde && 13 >= $stunde) {
    echo 'Es ist Nachmittag!<br />';
}
// Zwischen 12 und 13 Uhr
elseif (12 == $stunde) {
    echo 'Guten Appetitt!<br />';
}
// Zwischen 6 und 9 Uhr
elseif (6 >= $stunde && 9 < $stunde) {
    echo 'Zeit zum Aufstehen!<br />';
}
// Zwischen 9 und 12 Uhr (alle anderen Stunden)
else {
    echo 'Es ist Vormittag!<br />';
}
?>
```

Warum if (10 == $var) und nicht if ($var == 10) ? Beides ist korrekt, jedoch passiert es schnell, dass man mal ein = vergisst. Dann steht da plötzlich **if ($var = 10)** was immer zutreffend wird, weil der Wert 10 der Variable $var immer zugewiesen werden kann. Wir erhalten also keinen Fehler. Im umgekehrten Fall **if (10 = $var)** jedoch schon!

3.10. switch / case

Das obige Beispiel mit den Wochentagen eignet sich auch hervorragend für ein *switch/case*-Konstrukt:

```
<?php
// Standardzeitzone
date_default_timezone_set('Europe/Berlin');

$wochentag = date('w');
switch ($wochentag) {
    // Wenn Samstag oder Sonntag
    case 6:
    case 0:
        echo 'Juhu, wir haben Wochenende!<br />';
        break;
    // Wenn Freitag
    case 5:
        echo 'Bald ist Wochenende!<br />';
        break;
    // Alle anderen Tage
    default:
        echo 'Arbeiten macht auch Spass!<br />';
        break;
}
?>
```

3.11. Trinitäts-Operator

Ist eine Kurzform des *if/else*-Konstrukts:

```
<?php
// Standardzeitzone
date_default_timezone_set('Europe/Berlin');

$wochentag = date('w');

// Variable = Wenn Bedingung Zutrifft -> true, sonst -> false
$wochenende = (0 == $wochentag || 6 == $wochentag) ? true : false;

// Wenn kein Wochenende ($wochenende ist Typ boolean)
if (!$wochenende) {
    echo 'Arbeiten macht auch Spass!<br />';
}
?>
```

Verwenden Sie den Trinitäts-Operator wirklich nur wenn es kurz und übersichtlich bleibt!

3.12. While-Schleife

Mit While-Schleifen können bestimmte Codeabschitte beliebig oft ausgeführt werden.

```
<?php
$i = 1;
while ($i <= 10) {
    echo '$i ist' . $i . '<br />';
    $i++;
}
?>
```

Die While-Schleife verwenden wir nur, wenn die Anzahl Durchläufe **nicht** bekannt ist!

3.13. For-Schleife

Auch mit For-Schleifen können bestimmte Codeabschitte beliebig oft ausgeführt werden. Sie wird immer dann bevorzugt, wenn die Anzahl Durchläufe im Voraus bekannt ist!

```
<?php
for ($i=1; $i<=10; $i++) {
    echo '$i ist' . $i . '<br />';
}
?>
```

3.14. Arrays

Arrays sind spezielle Variablen, welche mehr als 1 Wert beinhalten können.

```
<?php
// Schreibweise bei wenig Elementen
$integers = array(1, 2, 3, 4, 5);
$strings = array('eins', 'zwei', 'drei', 'vier', 'fünf');

// Schreibweise bei vielen Elementen
$integers[] = 1;
$integers[] = 2;
$integers[] = 3;
$integers[] = 4;
$integers[] = 5;
$strings[] = 'eins';
$strings[] = 'zwei';
$strings[] = 'drei';
$strings[] = 'vier';
$strings[] = 'fünf';
?>
```

Da Sie den Inhalt eines Arrays nicht mehr einfach mit echo ausgeben können, verwenden Sie dafür bitte die Funktion print_r.

```
<?php
```

```
echo '<pre>';
print_r($integer);
print_r($strings);
echo '</pre>';
?>
```

Wie Sie sehen, beginnt ein Array mit dem Element 0. Deshalb erreichen Sie das 3. Element über den Array-Schlüssel 2.

```
<?php
echo $integers[2] . '<br />'; // 3
echo $strings[2] . '<br />'; // drei
?>
```

Dies muss nicht sein. Sie können die Array-Schlüssel auch selbst definieren:

```
<?php
// Array-Schlüssel als Integer
$zahlen[1] = 'eins';
$zahlen[2] = 'zwei';
echo $zahlen[1];
echo $zahlen[2];

// Array-Schlüssel als String
$zahlen['drei'] = 'Three';
$zahlen['vier'] = 'Four';
echo $zahlen['drei'];
echo $zahlen['vier'];
?>
```

3.15. Multidimensionale Arrays

Arrays können auch mehrere Dimensionen haben:

```
<?php
$zahlen['negativ'][] = -1;
$zahlen['negativ'][] = -2;
$zahlen['negativ'][] = -1;

$zahlen['positiv'][] = 1;
$zahlen['positiv'][] = 2;
$zahlen['positiv'][] = 1;

$zahlen['das']['ist']['ein']['sechs']['dimensionaler']['array']
['a'] = 'Hallo Welt A';
$zahlen['das']['ist']['ein']['sechs']['dimensionaler']['array']
['b'] = 'Hallo Welt B';

echo '<pre>';
print_r($zahlen);
echo $zahlen['das']['ist']['ein']['sechs']['dimensionaler']['array']
['b']; // Hallo Welt B
echo $zahlen['negativ'][1]; // -2
echo $zahlen['positiv'][0]; // 1
echo '</pre>';
?>
```

3.16. Foreach-Schleife

Die Foreach-Schleife ermöglicht ein einfaches Durchlaufen von Arrays. Hier zwei Beispiele:

```
<?php
// Einfache Arrays
$elemente['erste'] = 'Wert 1';
$elemente['zweite'] = 'Wert 2';
$elemente['dritte'] = 'Wert 3';
$elemente['vierte'] = 'Wert 4';
$elemente['fünfte'] = 'Wert 5';
// Einfache Foreach-Schleife
```

```php
foreach ($elemente as $position => $wert) {
    echo 'Das ' . $position . ' Element hat den Wert ' . $wert . '<br />';
}

// Multidimensionale Arrays
$i = 0;
$personen[$i]['Vorname'] = 'Rudolf';
$personen[$i]['Name'] = 'Müller';
$i++;
$personen[$i]['Vorname'] = 'Martina';
$personen[$i]['Zweiter Vorname'] = 'Martina';
$personen[$i]['Name'] = 'Meier';
$i++;
$personen[$i]['Vorname'] = 'Hans';
$personen[$i]['Name'] = 'Becker';
$personen[$i]['Alter'] = 45;
unset($i);
// Verschachtelte Foreach-Schleife
foreach ($personen as $nr => $person) {
    echo '<strong>Person Nr. ' . $nr . '</strong><br />';
    foreach ($person as $eigenschaft => $wert) {
        echo $eigenschaft . ': ' . $wert . '<br />';
    }
    echo '<hr />';
}
?>
```

3.17. Funktionen

Funktionen sind vorallem dazu da, wiederkehrende Codeabschnitte zu vereinen. Eine Funktion kann mehrere Parameter haben, jedoch nur einen Rückgabewert. Hier ein kleines Beispiel:

```php
<?php
/**
 * Berechnet den Mwst-Anteil eines Betrages und addiert diesen.
 *
 * @param mixed $betrag Betrag exkl. Mwst
 * @param float $mwst Mwst-Satz
 * @return float Betrag inkl. Mwst
 */
function addiereMwst($betrag, $mwst = 19.0)
{
    // Parameter umwandeln
    $betrag = (float) $betrag;
    $mwst = (float) $mwst;
    // Mwst-Betrag ausrechnen, runden und zum Betrag addieren
    $betrag += round($betrag/100*$mwst, 2);
    // Betrag zurückgeben
    return $betrag;
}

$test = 100;
echo addiereMwst($test); // 119
echo addiereMwst(55.85); // 66.46
echo addiereMwst(100, 7.6); // 107.6
?>
```

Wie Sie erkennen, werden Funktionen auch speziell dokumentiert. Der **mehrzeilige Kommentar** am Zeilenanfang hat ein spezielles Format. Wenn Sie Ihre Funktionen so dokumentieren, schaffen Sie eine gute Grundlage für den PHP-Documentor, den wir beim objekt-orientierten Programmieren häufig einsetzen werden. Mehr dazu in einem Fortgeschrittenen-Kurs.

Optionale Parameter kennen Sie ja schon aus dem Kapitel String-Funktionen. Der Parameter *$mwst* wird hier optional indem wir einfach einen Standardwert (19.0) zuweisen.

Als nächstes schauen wir uns den **Gültigkeitsbereich** von Variablen in Zusammenhang mit Funktionen an:

```php
<?php
$aussen = 'Ich bin ausserhalb von Funktionen.<br />';
```

```php
/**
 * Testfunktion
 *
 * @return void
 */
function test()
{
    echo $aussen; // Führt zu einem Fehler (undefined variable)
    $innen = 'Ich bin innerhalb der Funktion test().<br />';
}

echo $innen; // Führt zu einem Fehler (undefined variable)
?>
```

Sie werden feststellen, dass hier 2 Fehler entstehen. Variablen innerhalb einer Funktion sind nähmlich ausserhalb nicht zugänglich. Wenn eine Variable von ausserhalb in einer Funktion verwendet werden soll, ist diese als **Parameter** der Funktion zu übergeben:

```php
<?php
$aussen = 'Ich bin ausserhalb von Funktionen.<br />';

/**
 * Testfunktion
 *
 * @param string $var Bezeichnung
 * @return void
 */
function test($var)
{
    echo $var; // Funktioniert
    $innen = 'Ich bin innerhalb der Funktion test().<br />';
}

// Funktion ausführen
test($aussen);

echo $innen; // Führt immer noch zu einem Fehler (undefined variable)
?>
```

Es gibt als Alternative noch das Schlüsselwort *global*, das Variablen von ausserhalb in eine Funktion importieren kann. Dies ist aber wenn immer möglich zu **vermeiden**! Deshalb gehen wir hier mit Absicht nicht darauf ein 😁

Nun wissen wir, wie wir Variablen von ausserhalb in die Funktion einschleusen können. Aber wie bekommt man die Werte wieder aus der Funktion heraus? Dies geschieht über den **Rückgabewert** der nach *return* steht. Sobald eine Funktion einen Wert zurückgibt, wird diese abgebrochen.

```php
<?php
$aussen = 'Ich bin ausserhalb von Funktionen.<br />';

/**
 * Testfunktion
 *
 * @param string $var Bezeichnung
 * @return void
 */
function test($var)
{
    echo $var; // Funktioniert
    $innen = 'Ich bin innerhalb der Funktion test().<br />';
    // Rückgabewert
    return $innen;

    echo 'Test'; // wird nicht ausgeführt, da die Funktion mit return schon abgebrochen wurd
}

// Funktion ausführen
echo test($aussen); // Gibt nun den Inhalt von $innen aus.
?>
```

Schreiben Sie ein paar eigene Funktionen, damit Sie dieses Kapitel gründlich verstehen. Das Wissen über Funktionen bildet die wichtigste Grundlage für eine spätere objektorientierte Programmierung.

3.18. Konstanten

Konstanten sind Variablen, die weder überschrieben noch gelöscht werden können. Konstanten werden auschliesslich grossgeschrieben und dürfen Underlines im Namen enthalten. Hier einige Beispiele:

```
<?php
define('DAS_IST_EINE_SCHOENE_KONSTANTE', 'Schön!');
echo DAS_IST_EINE_SCHOENE_KONSTANTE;

define('KONSTANTE_A', 'Wert A');
echo KONSTANTE_A;

// Alternative für variable Namen
$buchstabe = 'A';
echo constant('KONSTANTE_' . $buchstabe);
?>
```

Übrigens sind Konstanten überall verfügbar – auch in Funktionen:

```
<?php
define('HELLO', 'Hallo Welt!');

/**
 * Testfunktion
 *
 * @return void
 */
function test()
{
    echo HELLO; // Funktioniert
}

// Funktion ausführen
test();
?>
```

3.19. include / require

Die Funktionen include, include_once, require und require_once bilden eine der wichtigsten Sprachelemente für PHP überhaupt. Diese binden Dateien in ein PHP-Skript ein und wertet diese aus. Erstellen Sie eine Datei mit dem Namen **C:/php/grundlagen/theorie/include/vars.php** mit folgendem Inhalt:

```
<?php
$farbe = 'rot';
$groesse = 'XL';
echo 'Hallo aus vars.php<br />';
?>
```

Erstellen Sie eine zweite Datei mit dem Namen **C:/php/grundlagen/theorie/include/include.php** mit folgendem Inhalt:

```
<?php
echo 'Start Test mit include()<br />';
echo $farbe . '<br />'; // funktioniert nicht, Fehler 'undefined variable'

// Datei vars.php einbinden
include('./vars.php');

echo $farbe . '<br />'; // gibt 'rot' aus
$farbe = 'blau'; // Hier überschreiben wir die Variable $farbe
echo $farbe . '<br />'; // gibt jetzt 'blau' aus

echo 'Mitte Test mit include()<br />';
```

```
// Datei vars.php nochmal einbinden
include('./vars.php');

echo $farbe . '<br />'; // gibt 'rot' aus, weil $farbe in vars.php wieder überschrieben wurde

echo 'Ende Test mit include()<br />';
?>
```

Rufen Sie das Skript unter **http://localhost/grundlagen/theorie/include/include.php** auf und sehen Sie was passiert.

Nun machen wir dasselbe mal mit einem include_once. Erstellen Sie eine dritte Datei mit dem Namen **C:/php/grundlagen/theorie/include/include_once.php** mit folgendem Inhalt:

```
<?php
echo 'Start Test mit include()<br />';
echo $farbe . '<br />'; // funktioniert nicht, Fehler 'undefined variable'

// Datei vars.php einbinden
include_once('./vars.php');

echo $farbe . '<br />'; // gibt 'rot' aus
$farbe = 'blau'; // Hier überschreiben wir die Variable $farbe
echo $farbe . '<br />'; // gibt jetzt 'blau' aus

echo 'Mitte Test mit include()<br />';

// Datei vars.php nochmal einbinden
include_once('./vars.php');

echo $farbe . '<br />'; // gibt 'blau' aus

echo 'Ende Test mit include()<br />';
?>
```

Sie werden feststellen, dass beim letzten *echo $farbe* die Farbe 'blau' ausgegeben wird. Das include_**once** sorgt also dafür, dass eine Datei nicht mehrfach eingebunden werden kann.

Probieren Sie die selben 2 Beispiele nun noch mit **require** bzw. **require_once** aus. Sie werden keinen Unterschied feststellen!

Der einzige Unterschied zwischen include und require ist folgender: Wenn die einzubindende Datei bei einem include nicht gefunden wird, wird zwar ein Fehler ausgegeben, das Skript läuft aber trotzdem weiter. Bei einem require hingegen wird auch dieses Skript abgebrochen. Probieren Sie es aus

---

4. Grundlagen Praxis

Nachdem wir nun die grundlegenden Bausteine von PHP kennen gelernt haben, sind Sie theoretisch schon in der Lage, kleinere bis mittlere Applikationen zu schreiben. In diesem Kapitel lernen Sie dies Schritt für Schritt umzusetzen.

Bitte speichern Sie alle folgenden Übungsdateien in einem Ordner **C:/php/grundlagen/praxis/** damit Sie die Dateien unter **http://localhost/grundlagen/praxis/** testen können.

4.1. Bildergalerie

Erstellen Sie einen Ordner **C:/php/grundlagen/praxis/bildergalerie/bilder/** und legen Sie dort ein paar Bilder im JPG-Format ab.

```
<?php
error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

// Alle Dateien mit der Endung .jpg aus dem Verzeichnis ./bilder/ auslesen
$bilder = glob('./bilder/*.jpg');

// Alle Bilder durchlaufen und anzeigen
foreach ($bilder as $bild) {
    echo '<img src="' . $bild . '" alt="' . basename($bild) . '" /><hr />';
}
?>
```

Speichern Sie das Skript unter **C:/php/grundlagen/praxis/bildergalerie/bilder/galerie.php** und testen Sie es. Wie Sie sehen, haben wir mit ein paar wenigen Zeilen eine einfache Bildergalerie erstellt.

4.2. Text-Dateien schreiben und lesen

PHP5 bringt zwei einfache Funktionen mit, um Dateien zu schreiben bzw. zu lesen: file_put_contents und file_get_contents

```
<?php
error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

// Zieldatei
$datei = './test.txt';

// Inhalt
$inhalt = 'Das ist ein Text.' . "\n"; // "\n" = neue Zeile

// Wir möchten nun diesen Text in eine Datei speichern.

/*
Wenn die Datei noch nicht existiert, wird sie erstellt und der Inhalt in die
Datei geschrieben. Wenn die Datei bereits existiert, wird der neue Inhalt
an den bereits vorhandenen angehängt (FILE_APPEND). Wenn wir den Inhalt
nicht anhängen möchten, sondern die Datei immer wieder neu angelegt werden
soll, lassen wir diesen 3. Parameter einfach weg.
*/
if (file_put_contents($datei, $inhalt, FILE_APPEND)) {
    echo 'Die Datei wurde erfolgreich erstellt und der Inhalt gespeichert.
<br />';
}

// Variablen, die wir nicht mehr brauchen, löschen
unset($inhalt);

// Nun möchten wir die Datei auslesen

// Fehler und Skriptabbruch wenn Datei nicht existiert
if (!is_file($datei)) {
    die('Die Datei existiert noch nicht!');
}

// Fehler und Skriptabbruch wenn Datei nicht lesbar
if (!is_readable($datei)) {
    die('Die Datei kann nicht gelesen werden!');
}

/*
Fehler und Skriptabbruch wenn Inhalt der Datei nicht in Variable $inhalt
gespeichert werden kann.
*/
if (!$inhalt = file_get_contents($datei)) {
    die('Die Datei konnte nicht ausgelesen werden!');
}

// Der Inhalt der Datei wurde nun erfolgreich in $inhalt gespeichert.

// Inhalt mit HTML-Zeilenumbrüchen ausgeben
echo 'Der aktuelle Inhalt der Datei ' . $datei . ' ist folgender:<hr />';
echo nl2br($inhalt);
?>
```

Führen Sie das Skript ein paar mal aus (Browser aktualisieren) und sehen Sie selbst was passiert. Öffnen Sie das erstellte Text-File auch mal mit einem Text-Editor (Notepad, Wordpad etc.).

Schauen Sie sich auch alle die anderen Dateisystem-Funktionen im Manual an.

4.3. Arrays serialisieren

Sie haben in vorherigen Kapitel gelernt, wie man normale String-Variablen in eine Datei speichern kann. Wie macht man das nun mit Arrays ohne jeden einzelnen Wert darin zu speichern? Ausserdem könnte man aus solchen Daten in der Datei nur mit grossem Aufwand wieder einen Array erstellen!

PHP stellt zwei Funktionen zur Verfügung, die verschiedene Variablen in Strings umwandeln und zurückwandeln können: serialize und unserialize

```
<?php
error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

// Kunden-Array initialisieren
$kunden = array();

// Kunde 1
$kunde['vorname'] = 'Rudolf';
$kunde['name'] = 'Müller';
$kunde['jahrgang'] = 1960;

// Kunde 1 dem Kunden-Array hinzufügen
$kunden[] = $kunde;

// Kunde 2 ($kunde wird überschrieben, die Variable wurde ja oben bereits verwendet)
$kunde['vorname'] = 'Karl';
$kunde['name'] = 'Keller';
$kunde['jahrgang'] = 1965;

// Kunde 2 dem Kundenarray hinzufügen
$kunden[] = $kunde;

// Variable $kunde löschen, wir brauchen sie nicht mehr
unset($kunde);

// Wir haben nun unser Array $kunden und schauen es uns an:
echo '<pre>';
print_r($kunden);
echo '</pre><hr />';

/*
Diesen Array $kunden möchten wir nun in eine Datei speichern um ihn später
wieder verwenden zu können.
*/

// Zieldatei
$datei = './test.txt';

/*
Wir serialisieren nun diesen $kunden Array und speichern das Ergebnis
(Typ String) in der Variable $inhalt.
*/
$inhalt = serialize($kunden);

// Fehler wenn Datei nicht geschrieben werden konnte.
if (!file_put_contents($datei, $inhalt)) {
    die('Die Datei konnte nicht gespeichert werden!');
} else {
    echo 'Die Datei wurde erfolgreich erstellt und der Inhalt gespeichert.
<br />';
}

// Variablen, die wir nicht mehr brauchen, löschen
unset($inhalt, $kunden);

// Nun möchten wir die Datei wieder auslesen
```

```
// Fehler und Skriptabbruch wenn Datei nicht existiert
if (!is_file($datei)) {
    die('Die Datei existiert noch nicht!');
}

// Fehler und Skriptabbruch wenn Datei nicht lesbar
if (!is_readable($datei)) {
    die('Die Datei kann nicht gelesen werden!');
}

/*
Fehler und Skriptabbruch wenn Inhalt der Datei nicht in Variable $inhalt
gespeichert werden kann.
*/
if (!$inhalt = file_get_contents($datei)) {
    die('Die Datei konnte nicht ausgelesen werden!');
}

// Der Inhalt der Datei wurde nun erfolgreich in $inhalt gespeichert.

// Inhalt vom Typ String wieder zurückserialisieren in Array
$kunden = unserialize($inhalt);

// $kunden Array ausgeben
echo '<pre>';
print_r($kunden);
echo '</pre>';
?>
```

Gehen Sie dieses Beispiel Schritt für Schritt durch und versuchen Sie es mit verschiedenen Arrays aus.

4.4. Interaktionen mit dem Benutzer

Bezüglich Sicherheit gibt es eine Grunddevise: **Alles, was man nicht selbst geschrieben hat, ist unsicher!**

Das heisst, dass alle externen Daten, die von PHP verarbeitet werden, überprüft werden müssen. Dies gilt vorallem für Benutzereingaben. Bei Benutzereingaben wird hauptsächlich zwischen POST und GET unterschieden, die eine unterschiedliche Übertragungsart von Benutzereingaben darstellen. Mehr dazu in den folgenden Abschnitten.

4.5. Der GET-Parameter

Sogenannte GET-Parameter sind String-Variablen, die über die **URI** übertragen werden. Wenn Sie z.B. folgende URI in Ihrer Adresszeile stehen haben:

```
http://www.foobar.de/index.php?page=about&action=view
```

Erhält die Datei **index.php** auf dem Webserver die 2 Variablen **$_GET['page']** und **$_GET['action']** mit den Inhalten **'about'** bzw. **'view'**.

Beachten Sie, dass je nach Browser die Länge der URI begrenzt ist und somit keine unbegrenzte Anzahl an Parametern und Werten übertragen werden können.

Legen Sie eine neue Datei **C:/php/grundlagen/praxis/get/index.php** mit folgendem PHP-Code an:

```
<html>
<head>
<title>Meine Website</title>
</head>
<body>
<?php error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

/*
In diesem Beispiel-Skript werden wir eine kleine Website
mit PHP steuern. Die verschiedenen Inhaltsabschnitte werden wir in
mehrere Funktionen aufteilen.
*/
```

```php
// Hier werden die Seiten der Website in einem Array definiert
$meineSeiten['home'] = 'Home';
$meineSeiten['firma'] = 'Über uns';
$meineSeiten['produkte'] = 'Produkte';
$meineSeiten['kontakt'] = 'Kontakt';

/**
 * Zeigt das Menu zu den Seiten an.
 *
 * @param array $seiten Seiten-Array
 * @return void
 */
function zeigeMenu($seiten)
{
    foreach ($seiten as $key => $name) {
        echo '<a href="index.php?seite=' . urlencode($key) . '">';
        echo $name;
        echo '</a>';
        echo ' | ';
    }
    echo '<hr />';
}

/**
 * Zeigt den Inhalt unter "Home"
 *
 * @return void
 */
function zeigeHome()
{
    echo '<h1>Willkommen</h1>';
    echo 'Wie findet Ihr meine erste PHP-Website?';
}

/**
 * Zeigt den Inhalt unter "Firma"
 *
 * @return void
 */
function zeigeFirma()
{
    echo '<h1>Firmenportrait</h1>';
    echo 'Das ist ein Text &uuml;ber die Firma.';
}

/**
 * Zeigt den Inhalt unter "Produkte"
 *
 * @return void
 */
function zeigeProdukte()
{
    echo '<h1>Unsere Produkte</h1>';
    echo 'Hier stellen wir unsere Produkte vor.';
}

/**
 * Zeigt den Inhalt unter "Service"
 *
 * @return void
 */
function zeigeKontakt()
{
    echo '<h1>Kontakt</h1>';
    echo 'Rufen Sie uns an!';
```

```
}

/*
Bis hierher haben wir also unsere 4 Funktionen definiert, die bestimmte
Inhalte anzeigen. Wenn wir das Script nun so aufrufen würden, würde aber noch
nichts passieren. Wir müssen also noch veranlassen, wann welche Funktionen ausgeführt
werden soll.
*/

// Standard Seite
$seite = 'home';

/*
Wenn der GET Parameter 'seite' definiert ist und dieser als Schlüssel
im Array $meineSeiten existiert...
*/
if (!empty($_GET['seite']) && isset($meineSeiten[$_GET['seite']])) {
    // ... überschreibe die Standard Seite mit dem Inhalt des GET-
Parameters 'seite'
    $seite = $_GET['seite'];
}

/*
Da das Menu auf Wir zeigen nun das Menu an, indem wir der Funktion zeigeMenu() den Array-
Parameter
$meineSeiten übergebe und ausführen.
*/
zeigeMenu($meineSeiten);

/*
Nun möchten wir den Inhalt anzeigen. Anhand der Variable $seite wissen wir ja, welcher
Inhalt angezeigt werden soll. Dies eignet sich wiederum hervorragend für ein switch/case
Konstrukt:
*/
switch ($seite) {
    case 'firma':
        zeigeFirma();
        break;
    case 'produkte':
        zeigeProdukte();
        break;
    case 'kontakt';
        zeigeKontakt();
        break;
    default:
        zeigeHome();
        break;
}
?>
</body>
</html>
```

Lesen Sie Zeile für Zeile genau durch und versuchen Sie alle Funktionen mit Hilfe des PHP-Manuals zu verstehen. Testen Sie das PHP-Skript unter **http://localhost/grundlagen/praxis/get/index.php** und klicken Sie auf die Links. Achten Sie besonders auf die Adresszeile Ihres Browsers. Versuchen Sie auch mal die Website zu "hacken" indem Sie z.B. einen falschen Parameter-Wert anhängen: **http://localhost/grundlagen/praxis/get/index.php?seite=hack** Dies funktioniert dank der Prüfung der folgenden Zeile nicht mehr.

```
<?php
if (!empty($_GET['seite']) && isset($meineSeiten[$_GET['seite']])) {
?>
```

Fahren Sie bitte erst weiter, wenn Sie dieses Skript von A bis Z verstanden haben 😁

Als **Zusatzaufgabe** können Sie die jeweiligen Inhalte auch in separate Dateien auslagern und via **include** bzw. **require** einbinden!

4.6. Apachemodul mod_rewrite

GET-Parameter haben einen Nachteil: Sie sind nicht gerade suchmaschinenfreundlich und auch nicht schön anzusehen. Das Apachemodul mod_rewrite bietet die Möglichkeit, die URL zu **manipulieren**.

Die RewriteEngine des Apache-Webservers könnte also z.B. URLs wie

```
http://www.website.de/kontakt.htm
http://www.website.de/firma.htm
```

"für uns" wie folgt umschreiben:

```
http://www.website.de/index.php?seite=kontakt
http://www.website.de/index.php?seite=firma
```

Dies geschieht mit einem Parser für Reguläre Ausdrücke, der die angeforderte URL manipuliert. Kopieren Sie eine Text-Datei mit dem Namen **.htaccess** mit dem untenstehendne Inhalt in den Ordner **C:/php/grundlagen/praxis/modrewrite/**

```
RewriteEngine on
RewriteBase /modrewrite
RewriteRule ^([a-z]+).htm$ index.php?seite=$1
```

Danach erstellen Sie die Datei **C:/php/grundlagen/praxis/modrewrite/index.php** mit dem folgenden Inhalt:

```php
<?php
echo '<pre>';
print_r($_GET);
echo '</pre>';
?>
```

Rufen Sie bitte folgende URLs mal auf:

```
http://localhost/modrewrite/
http://localhost/modrewrite/kontakt.htm
http://localhost/modrewrite/firma.htm
http://localhost/modrewrite/blablablabla.htm
http://localhost/modrewrite/test1234.htm
```

Wie Sie sehen funktioniert dies nur, solange der reguläre Ausdruck erfüllt wird. Da [a-z] keine Ziffern enthält, funktioniert hier das letzte Beispiel nicht. Mit mod_rewrite können Sie also Ihre URL selbst gestalten. Beachten Sie, dass dieses Modul **nicht auf allen Webservern** aktiviert ist!

Weitere Infos finden Sie unter www.modrewrite.de.

4.7. Der POST-Parameter

Sogenannte POST-Parameter sind String-Variablen, die normalerweise via **Formulare** weitergereicht werden. Anders als bei GET, haben diese Parameter keine Begrenzung in ihrer Anzahl und Länge. Erstellen Sie einfaches HTML Formular mit unter **C:/php/grundlagen/praxis/post/index.php** mit folgendem Code:

```php
<html>
<head>
<title>Formular</title>
</head>
<body>
<form id="formular" name="formular" method="post" action="index.php">
    <label for="testfeld">Testfeld</label>
    <input name="testfeld" type="text" size="10" maxlength="10" />
    <input type="submit" name="submit" value="Abschicken" />
</form>
<?php error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

if ('POST' == $_SERVER['REQUEST_METHOD']) {
    echo '<pre>';
    print_r($_POST);
    echo '</pre>';
}
?>
</body>
</html>
```

Füllen Sie das Textfeld mal aus und klicken Sie auf "Abschicken". Ihnen wird nun der superglobale Array $_POST ausgegeben, der die übergebenen Werte enthält. Sie werden also in Zukunft über

den Namen des Formularfeldes (hier "testfeld") an den eingegebenen Wert kommen, indem Sie die Variable folgendermassen auswerten:

```
<?php
// Variable initialisieren
$testfeld = '';

// Wenn Formularfeld gesendet und ausgefüllt wurde
if (!empty($_POST['testfeld'])) {
    // Inhalt in Variable $testfeld speichern
    $testfeld = $_POST['testfeld'];
}

// Variable ausgeben
echo $testfeld;
?>
```

Zum besseren Verständnis der Formularverarbeitung zeigen wir Ihnen hier ein etwas grösseres Formular, dass Sie Zeile für Zeile durchlesen und verstehen sollten:

```
<html>
<head>
<title>Formular</title>
</head>
<body>
<?php error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

// Wenn Formular abgeschickt
if ('POST' == $_SERVER['REQUEST_METHOD']) {

    // Werte initialisieren (Standardwerte)
    $name = '';
    $kommentar = '';
    $geschlecht = 'm';
    $altersbereich['von'] = 0;
    $altersbereich['bis'] = 0;
    $haustiere = false;

    // Textfeld auswerten
    if (!empty($_POST['name'])) {
        $name = $_POST['name'];
    }

    // Textbereich auswerten
    if (!empty($_POST['kommentar'])) {
        $kommentar = $_POST['kommentar'];
    }

    // Radio Buttons auswerten
    if (!empty($_POST['geschlecht'])
     && ('m' == $_POST['geschlecht']
      || 'w' == $_POST['geschlecht'])) {
        $geschlecht = $_POST['geschlecht'];
    }

    // Select Liste auswerten
    if (!empty($_POST['altersbereich'])
     && strpos($_POST['altersbereich'], '-')) {
        $tmp = explode('-', $_POST['altersbereich']);
        if (is_numeric($tmp[0]) && $tmp[0] >= 0
         && is_numeric($tmp[1]) && $tmp[1] >= 0) {
            $altersbereich['von'] = (int) $tmp[0];
            $altersbereich['bis'] = (int) $tmp[1];
        }
    }

    // Checkbox auswerten
```

```
    if (!empty($_POST['haustiere'])
     && $_POST['haustiere'] == '1') {
        $haustiere = true;
    }

    // Titel
    echo '<h1>Sie haben das Formular mit folgenden Angaben abgeschickt:</h1>';

    // Angaben zusammenstellen
    $ausgabe = '<strong>Name: </strong>';
    $ausgabe .= $name . '<br />' . "\n";
    $ausgabe .= '<strong>Kommentar: </strong>';
    $ausgabe .= nl2br($kommentar) . '<br />' . "\n";
    $ausgabe .= '<strong>Geschlecht: </strong>';
    $ausgabe .= $geschlecht . '<br />' . "\n";
    $ausgabe .= '<strong>Altersbereich: </strong>';
    $ausgabe .= $altersbereich['von'] . ' bis ';
    $ausgabe .= $altersbereich['bis'] . '<br />' . "\n";
    $ausgabe .= '<strong>Haustiere: </strong>';
    $ausgabe .= ($haustiere ? 'Ja' : 'Nein') . '<br />' . "\n";
    echo $ausgabe;

    // Rohdaten
    echo '<hr /><h2>Die Rohdaten sahen so aus:</h2>';
    echo '<pre>';
    print_r($_POST);
    echo '</pre>';
}
// Wenn Formular noch nicht abgeschickt
else {
?>
    <h1>Formular</h1>
    <form name="formular" id="formular" method="post" action="index.php">
        <label for="name">Name</label>
        <input name="name" id="name" type="text" size="10" maxlength="10" />
        <p>
        <label for="kommentar">Kommentar</label>
        <textarea name="kommentar" id="textarea"></textarea>
        <p>
        <label for="geschlecht">Geschlecht</label>
        <input type="radio" name="geschlecht" id="geschlecht" value="m"
checked="checked" />m&auml;nnlich
        <input type="radio" name="geschlecht" id="geschlecht" value="w"
/>weiblich
        <p>
        <label for="altersbereich">Altersbereich</label>
        <select name="altersbereich" id="altersbereich">
            <option value="0-18">bis 18</option>
            <option value="18-25">18 bis 25</option>
            <option value="25-35">25 bis 35</option>
            <option value="35-45">35 bis 45</option>
            <option value="45-55">45 bis 55</option>
            <option value="55-65">55 bis 65</option>
            <option value="65-0">ab 65</option>
        </select>
        <p>
        <input type="checkbox" name="haustiere" id="haustiere" value="1" />
        <label for="haustiere">Ja, ich habe Haustiere</label>
        <p>
        <input type="submit" name="submit" value="Abschicken" />
    </form>
<?php }
?>
</body>
</html>
```

Versuchen Sie auch hier jede einzelne Zeile zu verstehen und schauen Sie alle Ihnen unbekannten

PHP-Funktionen im Manual nach!

4.8. Der REQUEST-Parameter

Der Request Parameter ist nichts anderes, als die Vereinigung von GET- und POST-Parametern. Sie empfangen also alle $_GET und $_POST Variablen auch als $_REQUEST Variable.

Testen Sie folgendes Skript mit verschiedenen Formularen und GET Parametern und sie werden feststellen, dass alle Variablen in $_GET und $_POST auch in $_REQUEST enthalten sind.

```
<?php
echo '<pre>';
echo 'GET';
print_r($_GET);
echo '<hr />';
echo 'POST';
print_r($_POST);
echo '<hr />';
echo 'REQUEST';
print_r($_REQUEST);
echo '</pre>';
?>
```

Wozu ist das nun gut? Ganz einfach: **Nur** dort, wo wir nicht wissen woher die Parameter kommen oder wo es mehrere Möglichkeiten gibt, einen Parameter zu übergeben, verwenden wir $_REQUEST.

4.9. Dateiuploads

Dateiuploads erhalten wir über den Array **$_FILES**. Beim Erstellen des Upload-Formulares ist es wichtig, dass im form-Tag ***enctype="multipart/form-data"*** steht. Ansonsten funktionieren Datei-Uploads nicht.

```
<html>
<head>
<title>Dateiupload</title>
</head>
<body>
    <h1>Dateiupload</h1>
    <form name="dateiupload" id="dateiupload" method="post" action="index.php"
enctype="multipart/form-data">
        <label for="datei">Datei</label>
        <input type="file" name="datei" id="datei" />
        <p>
        <input type="submit" name="submit" value="Upload" />
    </form>
    <hr />
<?php error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

// Wenn die Datei hochgeladen wurde
if (isset($_FILES['datei'])
 && is_uploaded_file($_FILES['datei']['tmp_name'])) {

    // Fehler und Skriptabbruch wenn ein Fehler beim Upload passiert ist
    if (UPLOAD_ERR_OK != $_FILES['datei']['error']) {
        die('Es ist ein Fehler beim Upload aufgetreten!
             Die Fehlernummer ist ' . $_FILES['datei']['error']);
    }

    // Nun wurde die Datei ins temporäre Upload-Verzeichnis des Servers geladen.

    // Wir möchten diese Datei in einem bestimmten Ordner abspeichern

    // Zielverzeichnis
    $zielpfad = './uploads/';

    // Fehler wenn Zielverzeichnis nicht existiert
    if (!is_dir($zielpfad)) {
```

```
        die('Das Zielverzeichnis ' . $zielpfad . ' existiert nicht!');
    }

    // Fehler wenn Zielverzeichnis nicht beschreibbar
    if (!is_writable($zielpfad)) {
        die('Das Zielverzeichnis ' . $zielpfad . ' ist nicht beschreibbar!');
    }

    // Dateiname des Benutzers übernehmen, aber alle Sonderzeichen entfernen!
    $zieldatei = preg_replace('#[^A-Za-z0-9_\-]#i', '', $_FILES['userfile']
['name']);

    // Fehler und Skriptabbruch wenn die Datei nicht in unser Zielpfad-
Verzeichnis verschoben werden konnte
    if (!move_uploaded_file($_FILES['datei']
['tmp_name'], $zielpfad . $zieldatei)) {
        die('Datei konnte nicht ins Verzeichnis ' . $zielpfad . ' abgespeichert werden!');
    }

    // Jetzt wurde die Datei erfolgreich hochgeladen

    // Erfolgsmeldung
    echo 'Die Datei wurde erfolgreich hochgeladen: ';
    echo '<a href="' . $zielpfad . $zieldatei . '">';
    echo $zielpfad . $zieldatei;
    echo '</a>';
}
?>
</body>
</html>
```

Die Fehlermeldungen bei Dateiuploads sind übrigens hier dokumentiert. Lesen bitte Sie auch den Aritkel Steuerung von Dateiuploads auf php.net.

4.10. Server-Variablen

Die Server-Variable $_SERVER ist ein vordefinierter, superglobaler Array, auf dessen Werte Sie von überall aus Zugriff haben. Aber auch dies sind externe Werte, auf die Sie sich nicht 100% verlassen können.

```
<?php
echo '<pre>';
print_r($_SERVER);
echo '</pre>';
?>
```

4.11. Dateibasierte Kundendatenbank

Sie wissen nun, wie Sie mit POST und GET zurechtkommen und Arrays serialisieren. Somit sind Sie in der Lage, kleine Applikationen zu schreiben. Hier ein Musterbeispiel, das Sie unter **C:/php/grundlagen/praxis/dateibasierte_kundendatenbank/index.php** speichern können:

```
<?php
error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

// In dieser Datei werden die Daten gespeichert
$datei = './kunden.txt';
?>
<html>
<head>
<title>Dateibasierte Kundendatenbank</title>
</head>
<body>
<?php
// Kundendatenbank auslesen
$kunden = array();
if (is_file($datei)
```

```
 && is_readable($datei)
 && $inhalt = file_get_contents($datei)) {

    // Serialisierter Array in $inhalt unserialisieren
    $kunden = unserialize($inhalt);

    // Variable $inhalt löschen
    unset($inhalt);
}

// Wenn Formular abgeschickt
if ('POST' == $_SERVER['REQUEST_METHOD']) {

    // Fehler und Skriptabbruch wenn Vorname oder Name nicht ausgefüllt
    if (empty($_POST['vorname'])
     || empty($_POST['name'])) {
        die('Sie haben nicht alle Fehler ausgef&uuml;llt!');
    }

    // Kunden Datensatz erstellen
    $kunde['vorname'] = trim($_POST['vorname']);
    $kunde['name'] = trim($_POST['name']);

    // Kunde zur Kundendatenbank hinzufügen
    $kunden[] = $kunde;

    // Variable löschen
    unset($kunde);

    // Kunden-Array serialisieren
    $inhalt = serialize($kunden);

    // Fehler wenn Datei nicht geschrieben werden konnte.
    if (!file_put_contents($datei, $inhalt)) {
        die('Kundendatenbank konnte nicht gespeichert werden!');
    } else {
        echo 'Der Datensatz wurde erfolgreich der Kundendatenbank hinzugefügt.
<br />';
    }
}

?>
    <table width="100%" border="1" cellspacing="0" cellpadding="5">
        <tr>
            <th scope="col">Vorname</th>
            <th scope="col">Name</th>
        </tr>
<?php
foreach ($kunden as $kunde) {
?>
        <tr>
            <td><?php echo htmlentities($kunde['vorname']);?></td>
            <td><?php echo htmlentities($kunde['name']);?></td>
        </tr>
<?php
}
?>
    </table>
    <form id="formular" name="formular" method="post" action="index.php">
        <label for="vorname">Vorname</label>
        <input name="vorname" type="text" size="10" maxlength="50" />
        <label for="name">Name</label>
        <input name="name" type="text" size="10" maxlength="50" />
        <input type="submit" name="submit" value="Speichern" />
    </form>
</body>
```

```
</html>
```

4.12. Sessions

Sessions bieten die Möglichkeit Daten zu einem Benutzer **seitenübergreiffend** zu verwenden. Mit Hilfe einer Session kann also z.B. ein Warenkorb während des ganzen Seitenbesuchs aufrecht erhalten werden, mehrseitige Formulare realisiert oder Login-/Logout-Skripts erstellt werden. Wir fangen mal mit einem kurzen Skript an:

```
<?php
error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

// Session initialisieren, oberhalb dieser Zeile dürfen keine Ausgaben gemacht werden!!!
session_start();

// Wenn Session-Variable $_SESSION['zaehler'] noch nicht definiert
if (!isset($_SESSION['zaehler'])) {
    // Zähler auf 0 setzen
    $_SESSION['zaehler'] = 0;
}

// Bei jedem Seitenaufruf um 1 erhöhen
$_SESSION['zaehler']++;

// Testausgabe
echo 'Sie haben diese Seite schon ' . $_SESSION['zaehler'] . ' mal aufgerufen!';
?>
```

Speichern Sie das Skript unter **C:/php/grundlagen/praxis/sessions/test.php** und beobachten Sie was passiert, wenn Sie das Script unter **http://localhost/grundlagen/praxis/sessions/test.php** aufrufen und Ihren Browser ein paar mal aktualisieren.

Session-Variablen verlieren also ihren Wert auch nach einer Aktualisierung des Browsers nicht. Da Logins eine zentrale Rolle in Webapplikationen darstellen, widmen wir uns diesem Thema:

```
<?php
error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

// Zugangsdaten für Login
$benutzer = 'gast';
$passwort = '1234';

// Session initialisieren
session_start();

// Login
if (!empty($_POST['aktion']) && $_POST['aktion'] == 'login') {
    // Prüfen ob eigegebene Zugangsdaten korrekt sind
    if (!empty($_POST['benutzer']) && $_POST['benutzer'] == $benutzer
     && !empty($_POST['passwort']) && $_POST['passwort'] == $passwort) {
        // Wenn ja -> Session-Variable setzen
        $_SESSION['eingeloggt'] = true;
    }
}
// Logout
else if (!empty($_GET['aktion']) && $_GET['aktion'] == 'logout') {
    // Session-Variable löschen
    unset($_SESSION['eingeloggt']);
    // Allfällige andere Daten aus der Session löschen
    session_destroy();
}
?>
<html>
<head>
<title>Login</title>
</head>
```

```
<body>
<?php
// Wenn noch nicht eingeloggt -> Login-Formular anzeigen
if (!isset($_SESSION['eingeloggt'])) {
?>
    <h1>Login</h1>
    <form id="login" name="login" method="post" action="login.php">
        <label for="benutzer">Benutzer</label>
        <input type="text" name="benutzer" id="benutzer" size="20" maxlength="50"
/>
        <p>
        <label for="passwort">Passwort</label>
        <input type="password" name="passwort" id="passwort" size="20"
maxlength="50" />
        <p>
        <input type="hidden" name="aktion" id="aktion" value="login" />
        <input type="submit" name="submit" value="Einloggen" />
    </form>
<?php
}
// Wenn schon eingeloggt
else {
?>
    <h1>Sie sind eingeloggt!</h1>
    <a href="login.php?aktion=logout">Logout</a>
<?php
}
?>
</body>
</html>
```

Speichern Sie dieses Skript unter **C:/php/grundlagen/praxis/sessions/login.php** und testen Sie es!

Beim Starten der Session wird eine Session-ID generiert, die beim Besucher normalerweise als **Cookie** abgelegt wird. Lässt der Besucher keine Cookies zu, fügt PHP automatisch die Session-ID der URL hinzu, damit Sie via **GET** weitergereicht werden kann. Im schlimmsten Fall kann es also sein, dass ein anderer User mit einer URL, die eine Session-ID enthält, an die Daten eines anderen Users kommt. Es gibt diverse kleine Tricks, um dieses Risiko zu reduzieren, jedoch sollten Sie **nie** heickle Daten wie z.B. Passwörter in der Session ablegen!

Weitere wichtige Infos über Sessions finden Sie im Manual.

4.13. E-Mails versenden

Um E-Mails via PHP versenden zu können verwenden Sie bitte den PHPMailer. Um die Mail-Funktionen auch lokal nutzen zu können, müssen Sie im XAMPP Control Panel **Mercury** aktivieren und konfigurieren.

Da der PHPMailer aber mit Klassen arbeitet und Vorkenntnisse der objektorientierten Programmierung vorraussetzt, gehen wir hier noch nicht darauf ein. Alternativ können Sie auch die PHP-Funktion mail verwenden, die aber nicht das gelbe vom Ei ist...

4.14. Mehrsprachige Websites

Die Mehrsprachigkeit ist immer wieder ein Thema. Sie können eine Website mehrsprachig machen, indem Sie für jede Sprache eine separate Datei anlegen.

Datei **C:/php/grundlagen/praxis/mehrsprachigkeit/sprachen/de.php**

```
<?php
// Sprachdatei Deutsch
$sprache['welcome'] = 'Willkommen';
$sprache['hello_world'] = 'Hallo Welt';
$sprache['title'] = 'Titel';
?>
```

Datei **C:/php/grundlagen/praxis/mehrsprachigkeit/sprachen/en.php**

```
<?php
// Sprachdatei Englisch
$sprache['welcome'] = 'Welcome';
```

```
$sprache['hello_world'] = 'Hello World';
$sprache['title'] = 'Title';
?>
```

Datei **C:/php/grundlagen/praxis/mehrsprachigkeit/sprachen/fr.php**

```
<?php
// Sprachdatei Französisch
$sprache['welcome'] = 'Bienvenue';
$sprache['hello_world'] = 'Bonjour Monde';
$sprache['title'] = 'Titre';
?>
```

In diesen Dateien werden also alle Sprachdaten in einem einheitlichen Array $sprache abgelegt. Bei den Array-Schlüsseln haben wir uns für Englisch entschieden.

Diese werden wir nun für unser Skript
C:/php/grundlagen/praxis/mehrsprachigkeit/index.php verwenden:

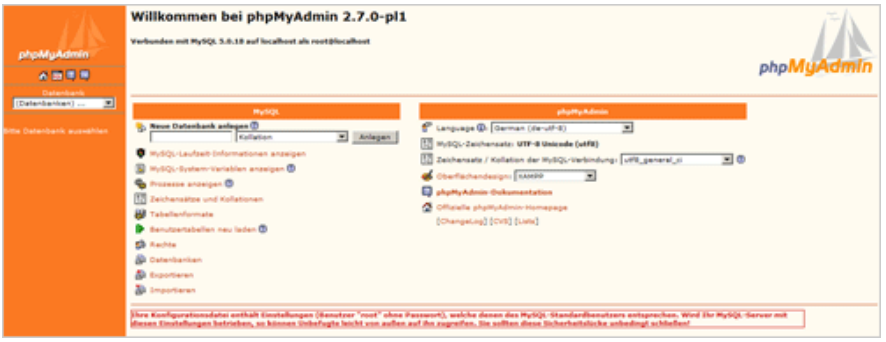
```
<?php
error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

echo '<h1>Mehrsprachige Website</h1>';

// Sprach-Array initialisieren
$sprachen = array();

// Alle Dateien mit der Endung .php aus dem Verzeichnis ./lang/ auslesen
$sprachDateien = glob('./sprachen/*.php');

// Alle Sprachdateien durchlaufen und Sprachcode in $sprachen speichern
foreach ($sprachDateien as $sprachDatei) {
    // Mach aus "./de.php" nur noch "de"
    $sprachen[] = str_replace(basename($sprachDatei), '.php');
}

// Sprachlinks ausgeben
foreach ($sprachen as $sprache) {
    echo '<a href="index.php?sprache=' . urlencode($sprache) . '">';
    echo strtoupper($sprache);
    echo '</a>';
}

echo '<hr />';

// Standardsprache
$aktuelleSprache = 'de';

// Wenn gültige Sprache als GET-Parameter definiert
if (!empty($_GET['sprache'])
 && in_array($_GET['sprache'], $sprachen)) {
    // Standardsprache überschreiben
    $aktuelleSprache = $_GET['sprache'];
}

// Sprachdatei der aktuellen Sprache einbinden
requier_once('./sprachen/' . $aktuelleSprache . '.php');

// Test
echo $sprache['welcome'] . '<br />';
echo $sprache['hello_world'] . '<br />';
echo $sprache['title'] . '<br />';
?>
```

Testen Sie das Skript unter
http://localhost/grundlagen/praxis/mehrsprachigkeit/index.php und gehen Sie Zeile für Zeile durch bis Sie das Skript verstehen.

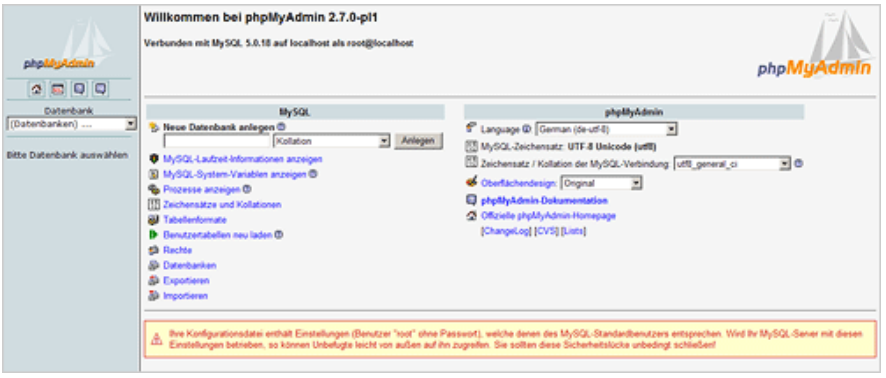
4.15. MySQL Datenbanken / phpMyAdmin

MySQL ist die wohl die meist verbreiteste Datenbank unter den PHP-Programmierern. Datenbanken bieten im Gegensatz zu Text-Dateien viel mehr Möglichkeiten was die Speicherung, Sortierung, Filterung usw. von Daten betrifft.

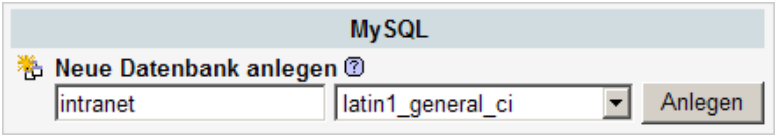
Als erstes öffnen Sie bitte Ihr XAMPP Control Panel und starten MySQL. Danach öffnen Sie bitte Ihren Browser und laden die Seite **http://localhost/phpmyadmin**



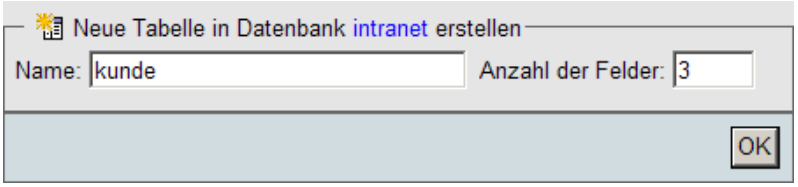
phpMyAdmin ist ein Tool, mit dem Sie MySQL-Datenbanken verwalten können. Schalten Sie das "Oberflächendesign" auf "Original", damit Sie mit dem Standard-Layout arbeiten.



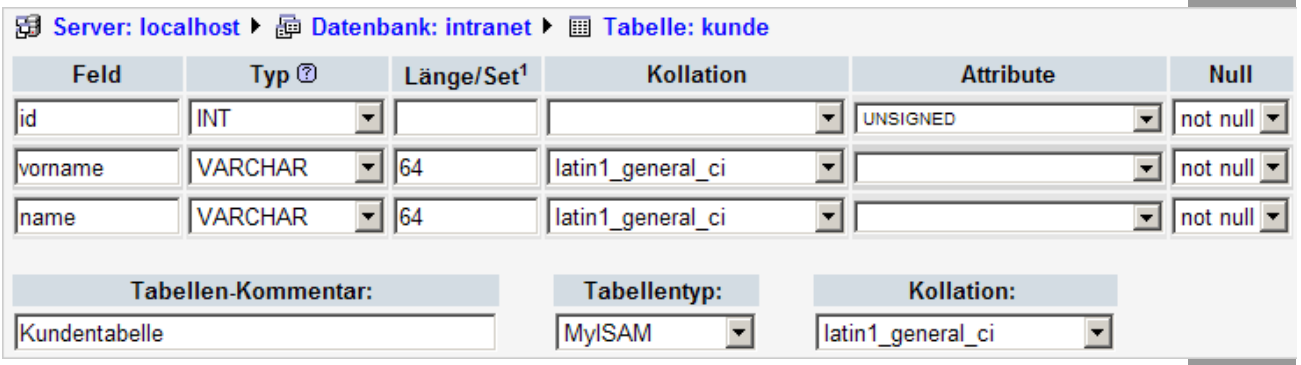
Legen Sie eine neue Datenbank mit dem Namen "**intranet**" und Kollation "**latin1_general_ci**" an.



Die Datenbank wurde nun erzeugt. Erstellen Sie nun eine neue Tabelle mit dem Namen "**kunde**" mit **3** Feldern. In dieser Tabelle werden später unsere Kunden gespeichert.



Legen Sie die Spalten wie folgt an: Teil 1



Teil 2

Die restlichen Angaben lassen Sie leer und klicken Sie auf **speichern**.

INT: Spaltentyp für Zahlen mit einem Maximalwert von 4'294'967'295, bestimmt in diesem Falle die maximale Anzahl Datensätze.

VARCHAR: Spaltentyp für beliebige Zeichen, maximal 255 Zeichen lang (hier auf 64 begrenzt)

Kollation: Zeichensatz der Spalte (normal *latin1_general_ci*, für Chinesisch o.ä. verwenden Sie bitte *utf8_general_ci*)

Attribut UNSIGNED: Stellt sicher, dass die Werte, die eingefügt werden, grösser gleich 0 sind.

Not Null: Verhindert, dass eine Spalte einen Nullwert erhält (sondern einen leeren String).

auto_increment: Für Integer-Spalten, sorgt für eine automatische Vergabe der id, sobald ein neuer Datensatz eingefügt wird.

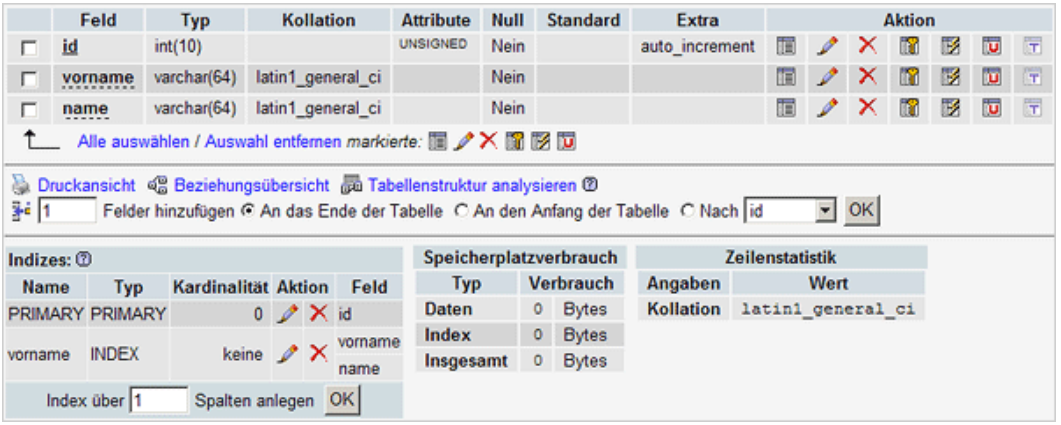
Primärschlüssel: Anhand des Primärschlüssels wird ein Datensatz eindeutig identifiziert. Dieser ist in den meisten Fällen *auto_increment* und wird von MySQL selbst verwaltet.

Index: Setzen Sie bei all jenen Spalten einen Index, nach denen später eine Sortierung oder eine Suche erfolgen soll.

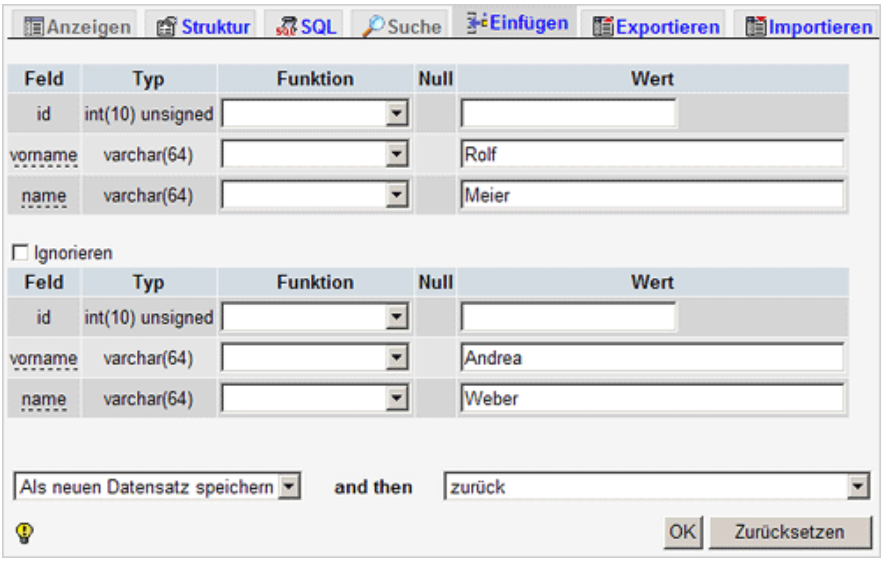
Unique: Setzen Sie all jene Spalten auf Unique, dessen Werte für alle Datensätze immer unterschiedlich sein müssen.

Kommentare: Hier haben Sie die Möglichkeit Kommentare für die einzelnen Spalten zu definieren, sie helfen aber nur Ihnen selbst.

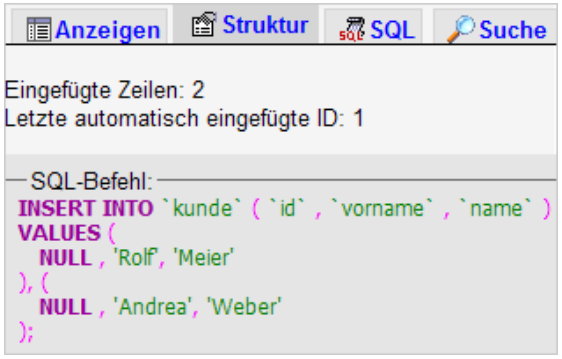
Sie sehen nun die **Struktur** der angelegten Tabelle "kunde":



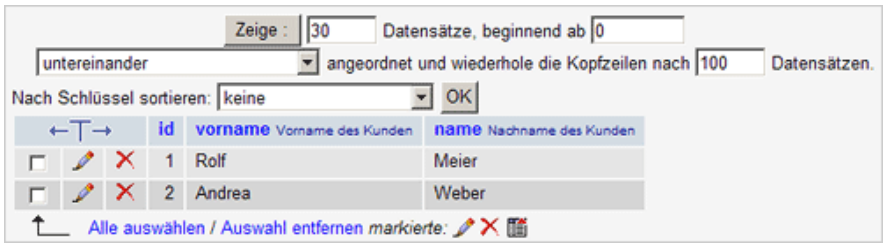
Klicken Sie nun auf den Reiter "**Einfügen**" und erfassen Sie 2 Datensätze:



Die **id** lassen wir bewusst **leer**, da diese ja **auto_increment** ist und von MySQL selbst verwaltet wird.

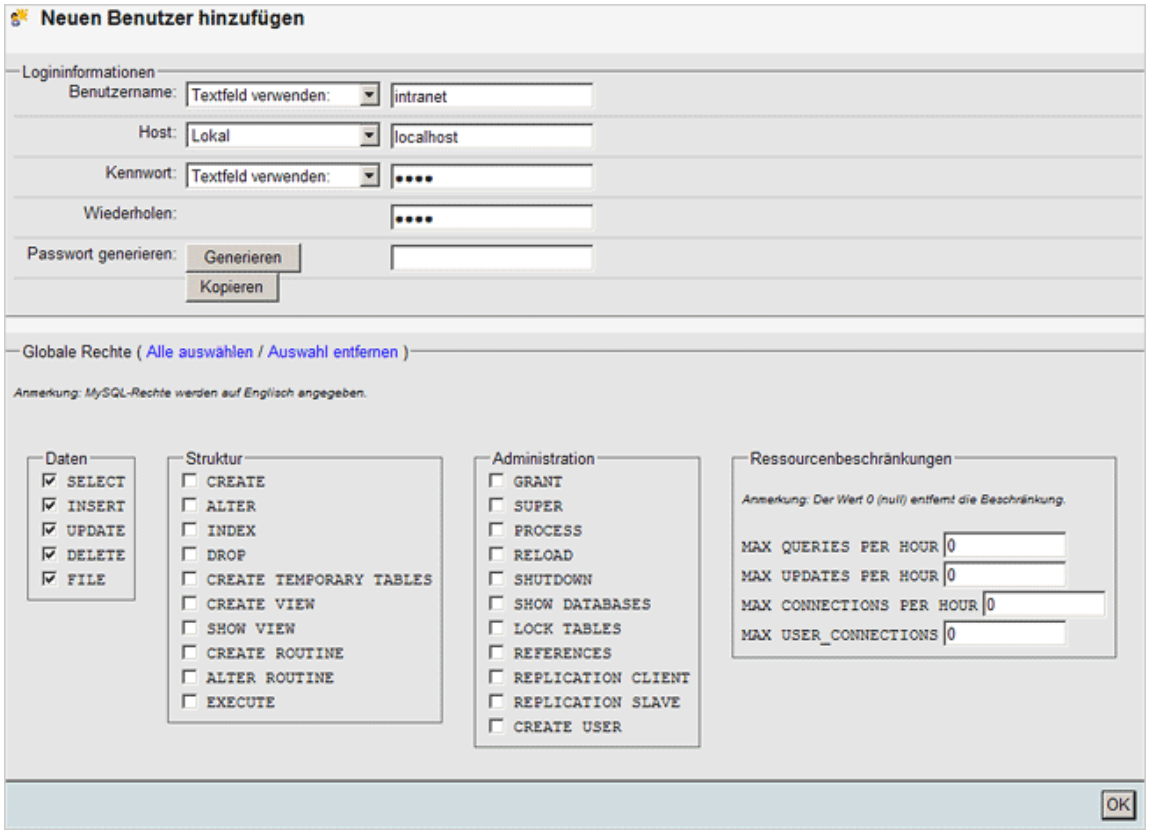


Die 2 Datensätze wurden nun erfolgreich gespeichert. Klicken Sie nun auf den Reiter "**Anzeigen**".



Wie Sie sehen erhielten die beiden Kunden automatisch die Id's 1 und 2.

Nun haben wir die Datenbank fertig eingerichtet und 2 Datensätze zum Test einfefügt. Bevor wir aber nun mit PHP diese Daten auslesen können, benötigen wir noch einen Datenbankbenutzer, der auf die Datenbank "intranet" auch Zugriff hat. Klicken Sie dazu zuoberst auf den Link "**Server: localhost**". Klicken Sie anschliessend auf "**Rechte**" und auf "**Neuen Benutzer hinzufügen**".



Erstellen Sie einen Benutzer mit dem Namen "**intranet**" und dem Passwort "**1234**" mit allen Rechten unter "**Daten**". Klicken Sie auf **OK** fahren Sie mit dem nächsten Kapitel fort.

4.16. Datensätze auslesen / einfügen / aktualisieren / löschen

Die **Verbindung zur Datenbank** speichern Sie bitte unter
C:/php/grundlagen/praxis/mysql/verbindung.php

```
<?php
error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

// MySQL Zugangsdaten / Konfiguration
```

```php
define('MYSQL_HOST', 'localhost');
define('MYSQL_BENUTZER', 'intranet');
define('MYSQL_PASSWORT', '1234');
define('MYSQL_DATENBANK', 'intranet');

// Verbindungsaufbau zum MySQL-Server
if (!$link = mysql_connect(MYSQL_HOST, MYSQL_BENUTZER, MYSQL_PASSWORT)) {
    die('Verbindung zur Datenbank fehlgeschlagen: ' . mysql_error());
}

// Datenbank selektieren
if (!mysql_select_db(MYSQL_DATENBANK)) {
    die('Datenbank ' . MYSQL_DATENBANK . ' konnte nicht ausgewählt werden: ' . mysql_error()
}
?>
```

Stellen Sie sicher, dass das Skript unter **http://localhost/grundlagen/praxis/mysql/verbindung.php** keine Fehlermeldungen ausgibt!

WICHTIG: Debuggen Sie jede MySQL-Funktion mit mysql_error()! Nur so finden Sie heraus, warum ein Skript nicht so tut wie es sollte!

Nun wollen wir unsere 2 Datensätze, die wir im vorherigen Kapitel manuell eingefügt haben, auslesen. Dies erfolgt folgendermassen:

```php
<?php
// Datenbankverbindung einbinden
require_once('./verbindung.php');

// Abfrage Befehl
$query = 'SELECT `vorname`, `name`
          FROM `kunde`
          ORDER BY `name`, `vorname`';

/*
Zu Deutsch: Lese die Werte der Spalten `vorname` und `name`
aller Kunden aus und sortiere Sie alphabetisch aufwärts nach
Name und innerhalb des gleichen Nachnamens nach dem Vornamen.
*/

// Befehl ausführen und Resultat in $result speichern
if (!$result = mysql_query($query)) {
    die('Befehl fehlgeschlagen: ' . mysql_error());
}

// Meldung und Skriptabbruch wenn keine Datensätze gefunden wurden
if (0 == mysql_num_rows($result)) {
    echo 'Es wurden keine Datensätze gefunden!';
    exit;
}

// Ergebnis durchlaufen und Spaltenwerte in Array $row speichern
while ($row = mysql_fetch_assoc($result)) {
    echo htmlentities($row['vorname']);
    echo ' ';
    echo htmlentities($row['name']);
    echo '<br />';
}
?>
```

Speichern Sie dieses Skript unter **C:/php/grundlagen/praxis/mysql/select.php** und führen Sie es aus. Vertauschen Sie beim **ORDER BY** Befehl mal `**name**` und `**vorname**` und sehen Sie was passiert. Lesen Sie alle verwendeten MySQL-Funktionen im Manual nach! Nun wissen Sie, wie man Daten aus einer MySQL-Datenbank auslesen kann

Um einen **neuen Datensatz** in die Tabelle "kunde" **einzufügen** gehen wir folgendermassen vor:

```php
<?php
```

```php
// Datenbankverbindung einbinden
require_once('./verbindung.php');

// Der Kunde, den wir speichern möchten
$kunde['vorname'] = 'Peter';
$kunde['name'] = 'Koch';

/*
Da dieser MySQL-Befehl (String) viele ' beinhaltet, verwenden wir hier
ausnahmsweise die doppelten Anführungszeichen damit der Code lesbarer wird:
*/
$query = "INSERT INTO `kunde` SET
              `vorname` = '" . mysql_real_escape_string($kunde['vorname']) . "',
              `name` = '" . mysql_real_escape_string($kunde['name']) . "'";

// Befehl ausführen und Fehlermeldung bei Misserfolg
if (!mysql_query($query)) {
    die('Befehl fehlgeschlagen: ' . mysql_error());
}

// Die zuletzt eingefügte auto_increment id ausgeben
echo 'Die soeben eingefügte id ist: ' . mysql_insert_id();
?>
```

Speichern Sie dieses Skript unter **C:/php/grundlagen/praxis/mysql/insert.php** und führen Sie es aus. Beim ersten Skriptaufruf erhält der Kunde also die id 3, beim dritten die 4 usw. Stellen Sie sicher, dass die Datensätze eingefügt wurden, in dem Sie wieder **http://localhost/grundlagen/praxis/mysql/select.php** aufrufen.

Um einen **bestehenden Datensatz** in die Tabelle "kunde" zu **ändern** gehen wir folgendermassen vor:

```php
<?php
// Datenbankverbindung einbinden
require_once('./verbindung.php');

// Der Kunde, den wir ändern möchten
$kunde['id'] = 2;
$kunde['vorname'] = 'Andreas'; // (vorher Andrea)
$kunde['name'] = 'Wenger'; // (vorher Weber)

// Update-Befehl
$query = "UPDATE `kunde` SET
              `vorname` = '" . mysql_real_escape_string($kunde['vorname']) . "',
              `name` = '" . mysql_real_escape_string($kunde['name']) . "'
          WHERE `id` = " . $kunde['id'];

// Befehl ausführen und Fehlermeldung bei Misserfolg
if (!mysql_query($query)) {
    die('Befehl fehlgeschlagen: ' . mysql_error());
}
?>
```

Speichern Sie das Skript unter **C:/php/grundlagen/praxis/mysql/update.php** und führen Sie es unter **http://localhost/grundlagen/praxis/mysql/update.php** aus. Überprüfen Sie ob das Update erfolgreich war, in dem Sie wieder **http://localhost/grundlagen/praxis/mysql/select.php** aufrufen.

Als letztes möchten wir noch den **Datensatz** mit der **id 3** aus der Tabelle "kunde" **löschen**. Dies geschieht folgendermassen:

```php
<?php
// Datenbankverbindung einbinden
require_once('./verbindung.php');

// Die id des Kunden, den wir löschen möchten
$id = 3;

// Update-Befehl
```

```php
$query = "DELETE FROM `kunde` WHERE `id` = " . $id;

// Befehl ausführen und Fehlermeldung bei Misserfolg
if (!mysql_query($query)) {
    die('Befehl fehlgeschlagen: ' . mysql_error());
}
?>
```

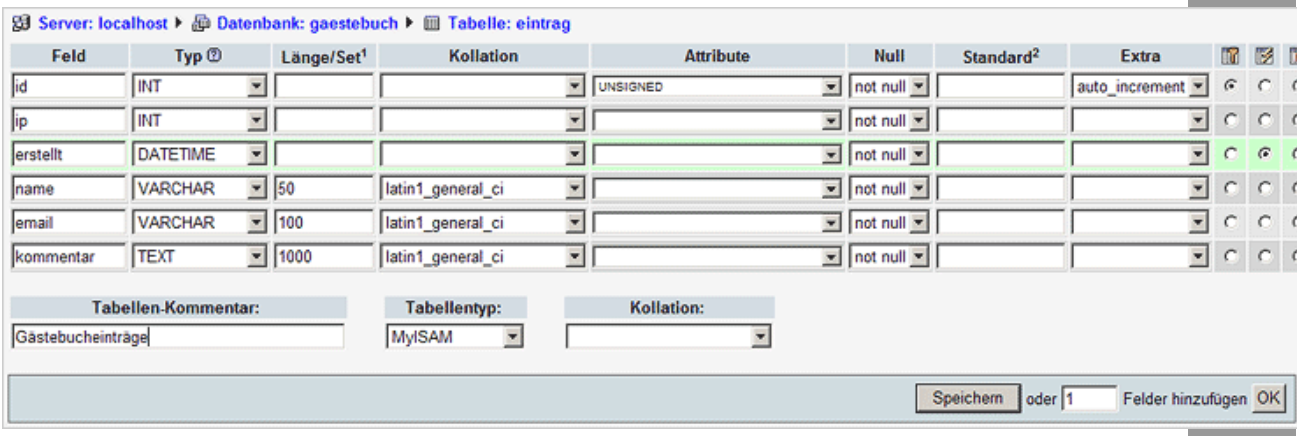
Speichern Sie das Skript unter **C:/php/grundlagen/praxis/mysql/delete.php** und führen Sie es aus. Überprüfen Sie wieder ob die Löschung erfolgreich war, in dem Sie wieder **http://localhost/grundlagen/praxis/mysql/select.php** aufrufen.

Beachten Sie, dass Sie alle nicht numerischen Variablen immer mit der Funktion mysql_real_escape_string behandeln sollten, bevor Abfragen gestartet werden. Dies verhindert SQL-Injections. **Tabellen- und Spaltennamen** werden immer in **Backticks** (`) geschrieben und alle nicht numerische **Spaltenwerte** werden zwischen **einfachen Anführungszeichen** (') geschrieben. Die **Tabellennamen** sollten in **Singular** definiert werden (nicht "kunden"). Dies wird uns später das Leben beim Arbeiten mit einem ORM erleichtern.

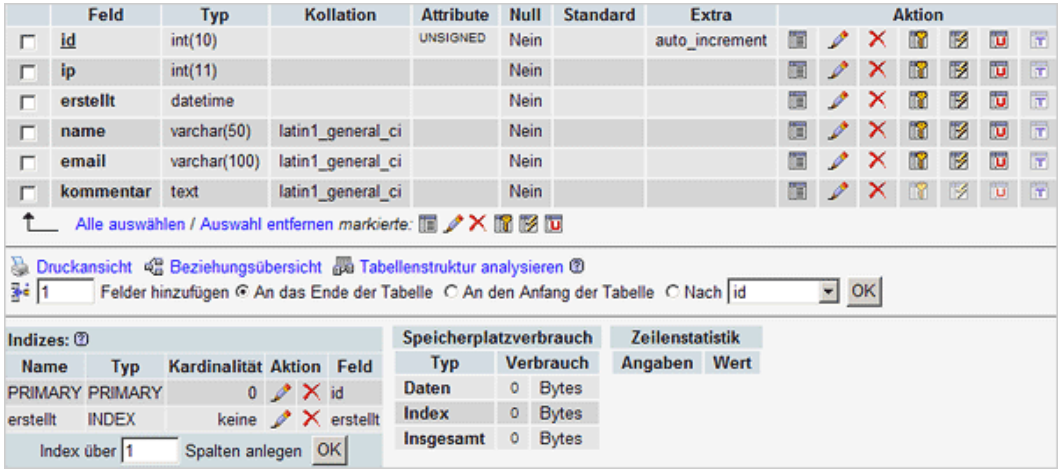
Das arbeiten mit MySQL gehört bei PHP zum Alltag. Deshalb sollten Sie sich auch die Links zu MySQL genau durchlesen.

4.17. MySQL Gästebuch

Erstellen Sie in phpMyAdmin eine neue **Datenbank** mit dem Namen "**gaestebuch**" und einen neuen **Benutzer** (Benutzername: **gaestebuch**, Passwort: **1234**) und legen Sie eine neue **Tabelle** "**eintrag**" mit folgenden **6 Spalten** an:



Wie Sie sehen, ist **id** wieder Primärschlüssel und **erstellt** definieren wir als Index, da später nach Datum sortiert werden soll.



Hier das zugehörige Skript, das auch gegen Attacken einigermassen gewidmet sein sollte

```php
<?php
error_reporting(E_ALL|E_STRICT);
ini_set('display_errors', true);

// Konfiguration
define('EINTRAEGE_PRO_SEITE', 4);
define('ZEITSPERRE', 60); // in Sekunden
```

```
// MySQL Zugangsdaten / Konfiguration
define('MYSQL_HOST', 'localhost');
define('MYSQL_BENUTZER', 'gaestebuch');
define('MYSQL_PASSWORT', '1234');
define('MYSQL_DATENBANK', 'gaestebuch');

// Diverse Header Einstellungen
session_cache_limiter('none');
session_start();
header('Cache-Control: no-store');
header('Pragma: no-cache');

// Standard Zeitzone festlegen
date_default_timezone_set('Europe/Berlin');

// Formular freigeben
if (!isset($_SESSION['gesperrt'])) {
    $_SESSION['gesperrt'] = false;
}

// Verbindungsaufbau zum MySQL-Server
if (!$link = mysql_connect(MYSQL_HOST, MYSQL_BENUTZER, MYSQL_PASSWORT)) {
    die('Verbindung zur Datenbank fehlgeschlagen: ' . mysql_error());
}

// Datenbank selektieren
if (!mysql_select_db(MYSQL_DATENBANK)) {
    die('Datenbank ' . MYSQL_DATENBANK . ' konnte nicht ausgewählt werden: ' . mysql_error()
}

//-------------------------------------------------------------------------------
-

// Fehlermeldungen für den Besucher initialisieren
$fehler = array();

// Werte initialisieren (leer stellen)
$name = '';
$email = '';
$kommentar = '';

// Wenn Formular abgeschickt
if ('POST' == $_SERVER['REQUEST_METHOD']
 && isset($_SESSION['sicherheitscode'])
 && $_SESSION['sicherheitscode'] == $_POST['sicherheitscode']
 && !empty($_POST['name'])
 && !empty($_POST['email'])
 && !empty($_POST['kommentar'])) {

    // Allfällige Whitespaces am Anfang und am Ende entfernen
    $name = trim($_POST['name']);
    $kommentar = trim($_POST['kommentar']);
    $email = trim($_POST['email']);

    // E-Mail (grob) überprüfen
    $regex = "/^([a-z0-9àáâãäåæçèéêëìíîïðñòóôöøùúûüýþÿ]";
    $regex .= "+([-_\.]?[a-z0-9àáâãäåæçèéêëìíîïðñòóôöøùúûüýþÿ])+)";
    $regex .= "@[a-z0-9àáâãäåæçèéêëìíîïðñòóôöøùúûüýþÿ]+([-_\.]?";
    $regex .= "[a-z0-9àáâãäåæçèéêëìíîïðñòóôöøùúûüýþÿ])+\.[a-z]{2,4}$/i";
    // Prüfen ob E-Mail Adresse einigermassen korrekt
    if (!preg_match($regex, $email)) {
        $fehler[] = 'Sie haben eine falsche E-Mail-Adresse angegeben!';
    }
    /*
    ACHTUNG: Um eine E-Mail RICHTIG zu überprüfen, sollte normalerweise das
```

```
            PEAR-Packet Mail (http://pear.php.net/package/Mail) verwendet
            werden. Die Klasse Mail/RFC822.php bietet diese Möglichkeit.
            Da wir aber in diesem Einsteigerkurs noch nichts mit Klassen
            und PEAR machen, beschränken wir uns auf einen einfachen Regex!
    */

    // IP ermitteln
    $ip = isset($_SERVER['REMOTE_ADDR']) ? $_SERVER['REMOTE_ADDR'] : '';

    // Wenn alles ok
    if (empty($fehler) && !$_SESSION['gesperrt']) {
        // Befehl
        $query = "INSERT INTO `eintrag` SET
                    `ip` = " . ip2long($ip) . ",
                    `erstellt` = NOW(),
                    `name` = '" . mysql_real_escape_string($name) . "',
                    `email` = '" . mysql_real_escape_string($email) . "',
                    `kommentar` = '" . mysql_real_escape_string($kommentar) . "'";
        // Befehl ausführen und bei Fehler, Meldung speichern
        if (!mysql_query($query)) {
            $fehler[] = 'Eintrag konnte nicht gespeichert werden!';
        }
        // Wenn erfolgreich gespeichert
        else {
            // Variablen wieder leeren
            $name = '';
            $email = '';
            $kommentar = '';
            // Formular sperren
            $_SESSION['gesperrt'] = true;
            // Zeitpunkt speichern
            $_SESSION['erstellt'] = time();
        }
    }
}

//--------------------------------------------------------------------------------
-

// Totale Anzahl Einträge ermitteln
$query = 'SELECT COUNT(`id`) AS anz FROM `eintrag`';

// Befehl ausführen und Resultat in $result speichern
if (!$result = mysql_query($query)) {
    die('Befehl fehlgeschlagen: ' . mysql_error());
}
$eintraegeTotal = 0;
while ($row = mysql_fetch_assoc($result)) {
    $eintraegeTotal = (int) $row['anz'];
}

// Einträge starten ab Eintrag Nr...
$start = 0;
if (!empty($_GET['seite']) && is_numeric($_GET['seite'])) {
    // GET Hacking verhindern
    $start = ((int) $_GET['seite'] * EINTRAEGE_PRO_SEITE)-EINTRAEGE_PRO_SEITE;
    if ($start >= $eintraegeTotal) {
        $start = $eintraegeTotal-EINTRAEGE_PRO_SEITE;
    }
}

// Anzahl Seiten berechnen
$seitenTotal = ceil($eintraegeTotal/EINTRAEGE_PRO_SEITE);

// Einträge auslesen und nach Datum abwärts sortieren
$query = "SELECT `id`, `name`, `email`, `kommentar`,
```

```
                DATE_FORMAT(`erstellt`, '%d.%m.%Y') AS datum,
                DATE_FORMAT(`erstellt`, '%H:%i:%s') AS uhrzeit
          FROM `eintrag`
          ORDER BY `erstellt` DESC, id DESC
          LIMIT " . $start . ', ' . EINTRAEGE_PRO_SEITE;

// Befehl ausführen und Resultat in $result speichern
if (!$result = mysql_query($query)) {
    die('Befehl fehlgeschlagen: ' . mysql_error());
}

// Ergebnis durchlaufen und Spaltenwerte in Array $row speichern
$eintraege = array();
while ($row = mysql_fetch_assoc($result)) {
    $eintraege[] = $row;
}
?>
<html>
<head>
<title>G&auml;stebuch</title>
<style type="text/css">
label {width:100px;}
</style>
</head>
<body>
    <h1>G&auml;stebuch</h1>
    <table width="100%" border="1" cellspacing="0" cellpadding="5">
<?php
// Wenn Einträge vorhanden
if (count($eintraege) > 0) {
    // Einträge anzeigen
    foreach ($eintraege as $eintrag) {
?>
        <tr>
            <td bgcolor="#CCCCCC">Eintrag <strong><?php echo $eintrag['id'];?>
</strong>
            geschrieben von <a href="mailto:<?
php echo urlencode(htmlentities($eintrag['email']));?>">
            <?php echo htmlentities($eintrag['name']);?></a>
            am <?php echo $eintrag['datum'];?> um <?
php echo $eintrag['uhrzeit'];?> Uhr</td>
        </tr>
        <tr>
            <td bgcolor="#EEEEEE"><?
php echo nl2br(htmlentities($eintrag['kommentar']));?></td>
        </tr>
<?php
    }
}
// Wenn keine Einträge vorhanden
else {
?>
        <tr>
            <td bgcolor="#CCCCCC">Keine Eintr&auml;ge vorhanden!</td>
        </tr>
<?php
}
?>
        <tr>
            <td colspan="2">Seite
<?php
// Seitenlinks anzeigen
for($i=1; $i<=$seitenTotal; $i++) {
    echo '<a href="index.php?seite=' . $i . '">' . $i . '</a> | ';
}
?>
```

```
            </td>
        </tr>
    </table>
    <p> </p>
<?php
// Fehlermeldungen anzeigen
foreach ($fehler as $meldung) {
    echo '<span style="color:#FF0000">';
    echo htmlentities($meldung);
    echo '</span><br />' . "\n";
}

// Doppelposts verhindern
if ($_SESSION['gesperrt'] && (time()-$_SESSION['erstellt']) < ZEITSPERRE) {
    echo htmlentities('Danke für Ihren Eintrag! Sie können erst in ' .
                      (ZEITSPERRE-(time()-$_SESSION['erstellt'])) .
                      ' Sekunden wieder einen Eintrag machen!');
} else {
    // Formular wieder freigeben
    if ($_SESSION['gesperrt']) {
        $_SESSION['gesperrt'] = false;
        unset($_SESSION['erstellt']);
        // Variablen wieder leeren
        $name = '';
        $email = '';
        $kommentar = '';
    }
    // neue Sicherheitscode setzen
    $_SESSION['sicherheitscode'] = md5('Fischers Fritz ' . rand());
?>
    <form id="formular" name="formular" method="post" action="index.php">
        <label for="name">Name</label>
        <input type="text" name="name" id="name" value="<?php echo $name;?>"
size="20" maxlength="50" />
        <p> </p>
        <label for="email">E-Mail</label>
        <input type="text" name="email" id="email" value="<?php echo $email;?>"
size="20" maxlength="100" />
        <p> </p>
        <label for="kommentar">Kommentar</label>
        <textarea name="kommentar" id="kommentar" cols="20" rows="5"><?
php echo $kommentar;?></textarea>
        <p> </p>
        <label for="submit"> </label>
        <input type="hidden" name="sicherheitscode" id="sicherheitscode" value="
<?php echo $_SESSION['sicherheitscode'];?>" />
        <input type="submit" name="submit" value="Eintragen" />
    </form>
<?php
}
?>
</body>
</html>
```

Sie können das Skript unter **C:/php/grundlagen/praxis/gaestebuch/index.php** abspeichern. Wenn Sie so ein Gästebuch selbst schreiben könnten, haben Sie ein gutes PHP-Grundwissen!

So... jetzt dürfen Sie sich für die **2421** Zeilen Code bedanken **...viel Spass beim Programmieren!**

---

5. Links

- PHP
  - PHP Manual
  - phpforum.de
  - php-faq.de
  - phpbar.de
  - quakenet:#php
- MySQL

- MySQL 5.1 Manual
- Datenbanken entwickeln

- Sonstiges
  - XAMPP
  - modrewrite.de