1)Einfache Ausgabe und Abfragen in PHP durchführen

Arbeite am Beispiel von „2_db_connection.docx“ weiter:

1a) Verbesserung mit ->fetch_array()

ist eine mächtige Funktion, die sich eine Zeile holt und diese in ein Array umwandelt. Sie liefert aber zwei Sachen, nämlich ein assoziatives (assoc) und ein indiziertes (row) Array zugleich. Der Nachteil ist, dass auch beide angezeigt werden, und somit alles doppelt zählt und jeden key bzw. value zweimal ausgibt.

Die beiden Unterarten sind:

- ->fetch_row() ist das indizierte Array und gibt die Zahlen aus
- ->fetch_assoc() ist das assoziative Array – hier werden als Schlüssel für die einzelnen Elemente die Feldnamen eingesetzt, wie hier z.B. pfl_id, name, bescheibung.

Will man nur eine Angabe haben, kann man sich auf eine Variante konzentrieren.
Das assoziative Array ist leichter verständlich als ein numerisches und wird in der Praxis häufiger eingesetzt, da man mit den Feldnamen meist mehr anfangen kann als mit den Zahlen.

Beispiel mit ->fetch_row()
führt zu dieser Ausgabe:

```
Array
(
    [0] => 1
    [1] => Feldahorn
    [2] => strauchartig, unter g◆nstigen Bedingunge
    [3] => 7.00
    [4] => 1
)
```

```
$erg = $con->query("SELECT * FROM pflanzen");
$zeile = $erg->fetch_row();
echo "<pre>";
print_r($zeile);
echo "<pre>";
```

Beispiel mit ->fetch_assoc()
führt zu dieser Ausgabe:

```
Array
(
    [pfl_id] => 1
    [name] => Feldahorn
    [beschreibung] => strauchartig, unter g◆nstigen Bedin
    [preis] => 7.00
    [liefer_id] => 1
)
```

```
$erg = $con->query("SELECT * FROM pflanzen");
$zeile = $erg->fetch_assoc();
echo "<pre>";
print_r($zeile);
echo "<pre>";
```

Code fetch_assoc:

```php
<?php
/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "garten4");
if ($con->connect_error) {
    echo "Fehler bei der Verbindung: " . mysqli_connect_error();
    exit();
}

$erg = $con->query("SELECT * FROM pflanzen");
$zeile = $erg->fetch_assoc();
echo "<pre>";
print_r($zeile);
echo "<pre>";
```

1b)Verbesserung mit while-Schleife und „fetch_array“

Die Funktion soll in einer **while-Schleife** aufgerufen werden. Gibt es keinen Datensatz mehr, liefert „fetch_array()“ NULL zurück und die Schleife ist damit beendet. Somit hört sie von selbst auf.
Beachte:

- Eingeschlossen in Anführungszeichen
- geschwungene Klammern
- eckige Klammern bei den Elementen der Variable

Code:

```
while($zeile = $erg->fetch_array() ) {
   echo "{$zeile['name']} {$zeile['beschreibung']} <br>";
```

```php
<?php
/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "garten4");
if ($con->connect_error) {
    echo "Fehler bei der Verbindung: " . mysqli_connect_error();
    exit();
}

$erg = $con->query("SELECT * FROM pflanzen");
while($zeile = $erg->fetch_array() ) {
    echo "{$zeile['name']} {$zeile['beschreibung']} <br>";
}
```

Code:

```
$erg = $con->query("SELECT * FROM pflanzen");
while($zeile = $erg->fetch_array() ) {
   echo "{$zeile['name']} {$zeile['beschreibung']} <br>";
```

Ergebnis im Browser:

1c)Ausgabe in einer Tabelle

While eignet sich sehr gut um einzelne Felder eines Datensatzes auszugeben.
Die Schleife wird von einem <table>_Tag umschlossen, innerhalb der Schleife wird jeder Datensatz als Tabellenzeile <tr> und innerhalb einer Zeile werden die einzelnen Werte als Tabellenfeld ausgegeben <td>.

Details:

- Oben steht die Verbindung zur Datenbank.
- Darunter wird PHP beendet und HTML für eine Überschrift 1 genutzt.
- Dann startet PHP wieder, um die Daten aus der Tabelle zu holen.

```php
<?php
/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "garten");
if ($con->connect_error) {
    echo "Fehler bei der Verbindung: " . mysqli_connect_error();
    exit();
}
?>

<h1>Unsere Pflanzen</h1>

<?php
$erg = $con->query("SELECT * FROM pflanzen");

echo "<table border='1'>";
while($zeile = $erg->fetch_array()) {
    echo    "<tr>
             <td> {$zeile['name']} </td>
             <td> {$zeile['bezeichnung']} </td>
             <td> {$zeile['preis']} </td>
             </tr>";
}
echo "</table>";
?>
```

Ergebnis im Browser:

Übung: Gib darunter die Liefertanten aus:

```
25
26  <h1>Unsere Lieferanten</h1>
27
28  <?php
29  $erg = $con->query("SELECT * FROM lieferanten");
30
31  echo "<table border='1'>";
32  while($zeile = $erg->fetch_array()) {
33      echo    "<tr>
34               <td> {$zeile['firma']} </td>
35               <td> {$zeile['strasse_nr']} </td>
36               <td> {$zeile['plz']} </td>
37               <td> {$zeile['ort']} </td>
38               </tr>";
39  }
40
41  echo "</table>";
42
43  $con->close();
44  ?>
```

Ergebnis:

2) Übung „Firma“ und „Personen“

Inhalt:

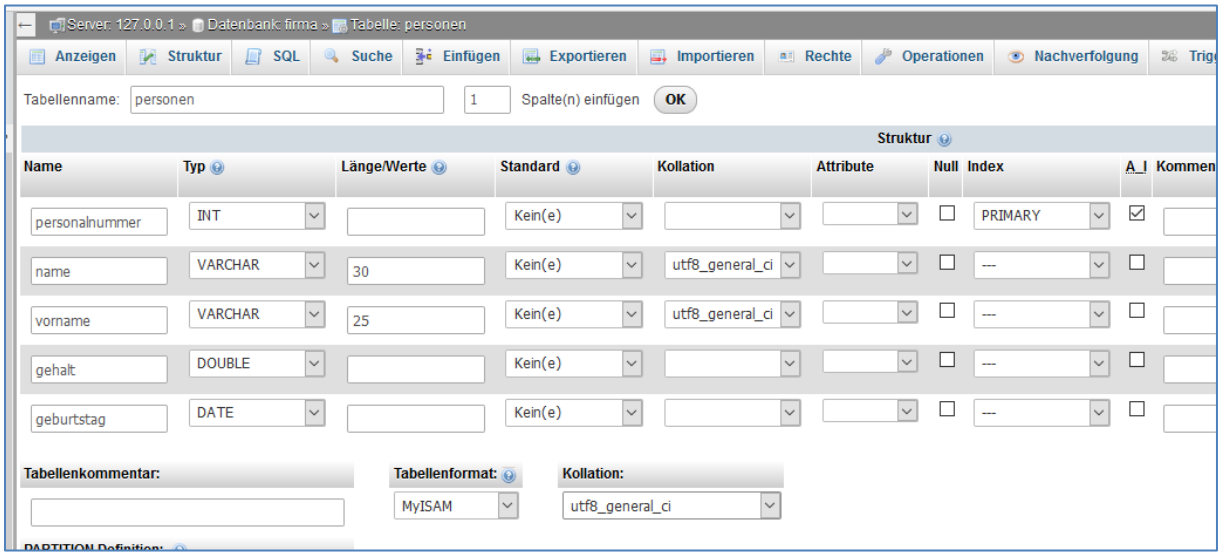
1. Datenbank mit phpMyAdmin anlegen und Datensätze einfügen
2. in PHP Verbindung aufnehmen und Abfragen mit SELECT
3. Ausgabe in HTML-Tabelle
4. Auswahl von Daten über ein Suchformular
5. Auswahl mit Radio-Buttons

2.1)Datenbank anlegen und Datensätze einfügen

- Anlegen der Datenbank
- Anlegen von Tabellen durch Angabe der Struktur
- Einfügen der Datensätze in die Tabellen

Beispieldatenbank und –tabelle

Es soll eine Datenbank „firma“ mit einer Datenbanktabelle „personen“ erzeugt werden.

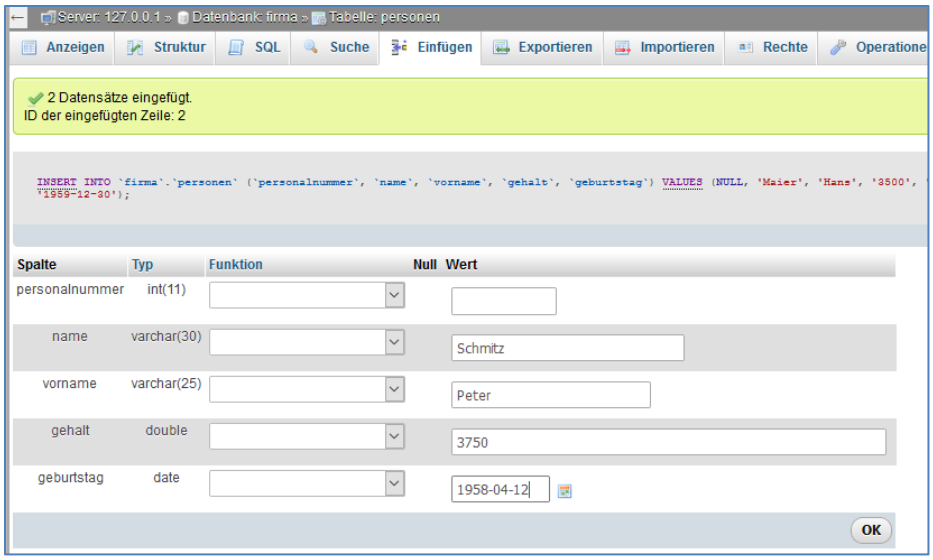


Datensätze eintragen:

Auf der Registrierkarte „Einfügen“ hat man die Möglichkeit die Datensätze einzugeben.

Folgende Daten:

personalnummer	name	vorname	gehalt	geburtstag
1	Maier	Hans	3500	1962-03-15
2	Mertens	Julia	3621.5	1959-12-30
3	Schmitz	Peter	3750	1958-04-12



Ergebnis:

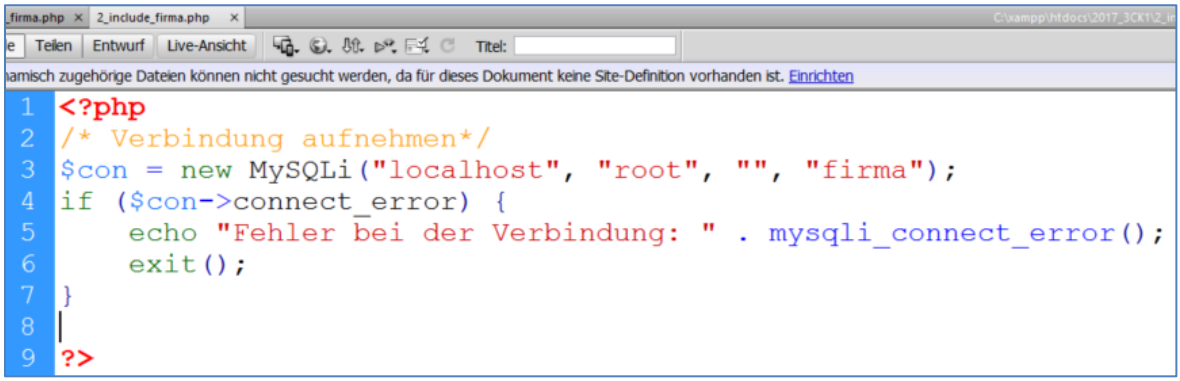
2.2)in PHP Verbindung aufnehmen und Abfragen

Hier wird die dynamische Schnittstelle zwischen dem Betrachter einer Internetseite und den Inhalten einer MySQL-Datenbank erzeugt.

2.2.1) Verbindung aufnehmen:

Mit Hilfe von PHP-Programmen kann eine komfortable Schnittstelle zum Erzeugen, Anzeigen, Ändern und Löschen von Datensätzen aus einer MySQL-Datenbank zur Verfügung gestellt werden. Vorher wird die Datenbank, wie hier oben, mit phpMyAdmin erzeugt.

Erstelle die folgende PHP-Datei als externe Verbindung: „2_include_firma.php“

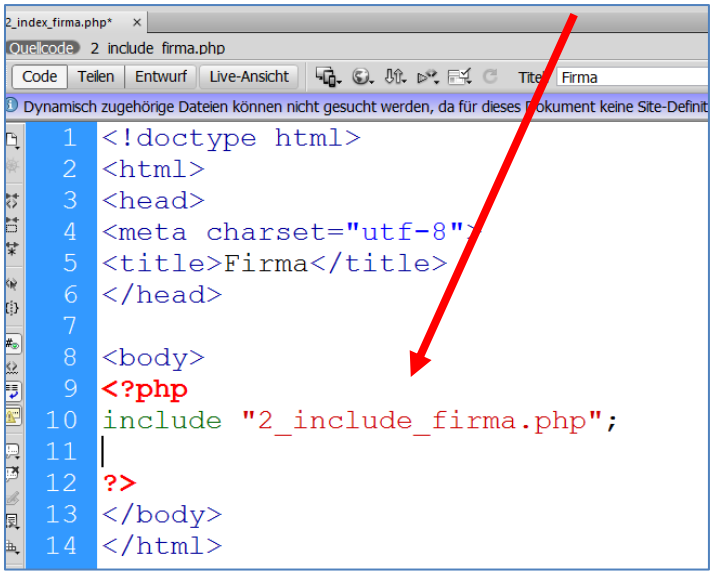


$con = new MySQLi("localhost", "root", "", "firma");

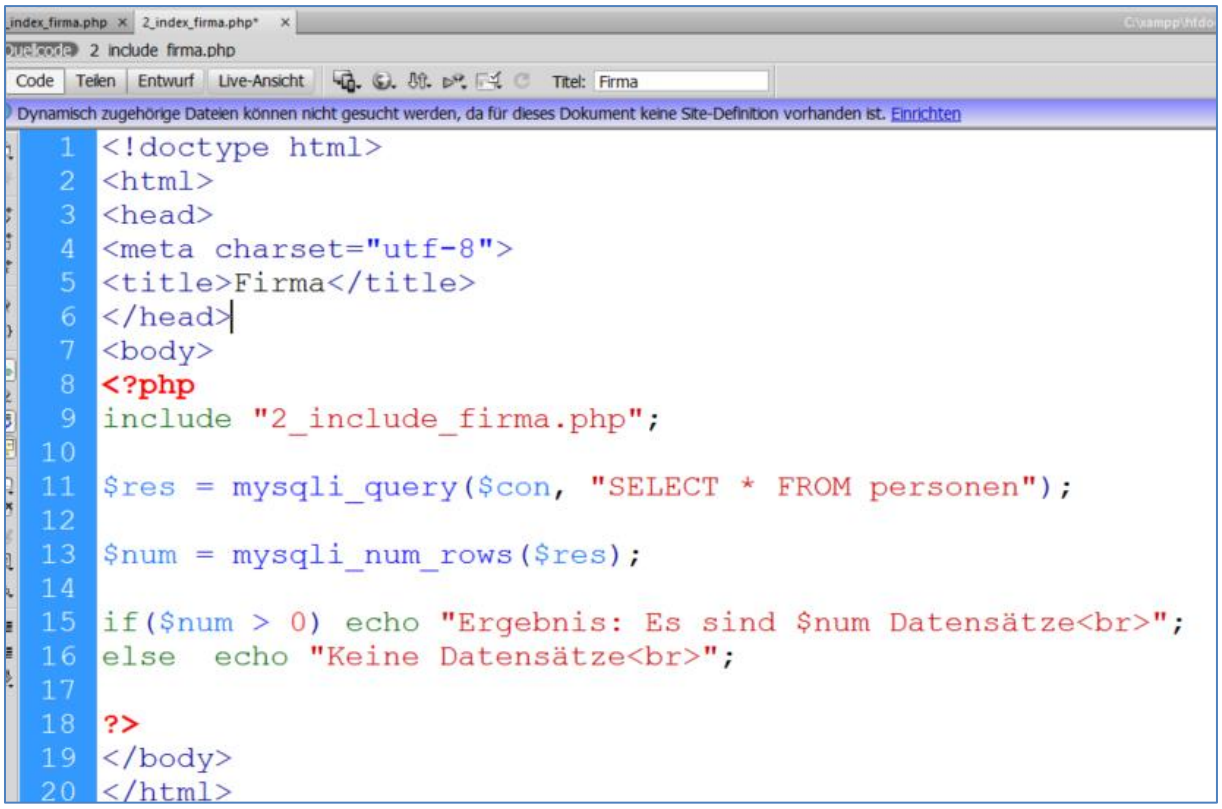
Erklärung:

new MySQLi() Die Funktion „new MySQLi()“ bzw. auch „mysqli_connect()“ öffnet eine Verbindung zum MySQL- Datenbankserver. Der Rückgabewert der Funktion „mysqli_connect()“ ist eine Referenz auf die Verbindung. Diese Referenz wird anschließend für weitere Funktionen benötigt und daher in der Variablen „$con“ gespeichert.

Diese binde mit „include“ in die neue Datei ein: „2_index_firma.php“.



Dann erstelle den Code der darunter abgebildet ist.



mysqli_query() Diese Funktion führt eine Abfrage mit der SQL-Anweisung SELECT in der aktuellen Datenbank aus. Die Abfrage soll alle Datensätze der betroffenen Tabelle liefern. Auch hier wird als erster Parameter die Referenz auf die Verbindung benötigt.

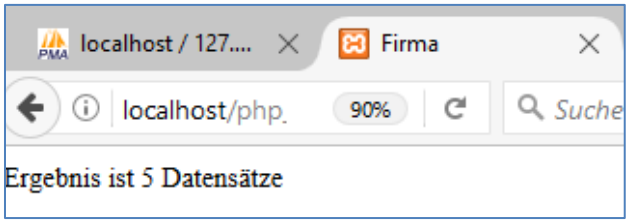
$con->query() Als Schreibweise könnte man alternativ, wie bis her verwendet, auch schreiben:

Ergebniskennung Falls die Abfrage erfolgreich ist, liefert die Funktion eine Ergebniskennung zurück - hier in der Variablen „$res“. Diese Ergebniskennung wird anschießend benötigt, um die einzelnen Komponenten des Ergebnisses zu ermitteln.

mysqli_num_rows() wird aufgerufen, wenn die Anzahl der Datensätze im Abfrageergebnis ermittelt werden soll. Als Parameter wird die Ergebniskennung übergeben, die man ermitteln möchte.

Kein Ergebnis Falls die Abfrage nicht erfolgreich ist, z.B. wegen eines SQL-Fehlers, hat die Variable „$num“ keinen Wert. Falls es keine Datensätze gibt, die der Abfrage genügen, hat die Variable „$num“ den Wert 0. In beiden Fällen wird die Meldung „Keine Ergebnisse“ ausgegeben.

Ergebnis:



<u>Danach soll folgendes ausgegeben werden:</u> Familienname, Vorname, PS-Nr, Gehalt Geburtsdatum

```
17 else echo "Keine Ergebnisse";
18
19 /*Datensätze aus Ergebnis ermitteln,*/
20 /*in Array speichern und ausgeben*/
21 while ($dsatz = mysqli_fetch_assoc($res))
22 {
23     echo     $dsatz["name"] . ", "
24              . $dsatz["vorname"] . ", "
25              . $dsatz["personalnummer"] . ", "
26              . $dsatz["gehalt"] . ", "
27              . $dsatz["geburtstag"] . "<br>";
28 }
29 ?>
30 </body>
31 </html>
```

<u>Code zum kopieren:</u>

```
while ($dsatz = mysqli_fetch_assoc($res))
```

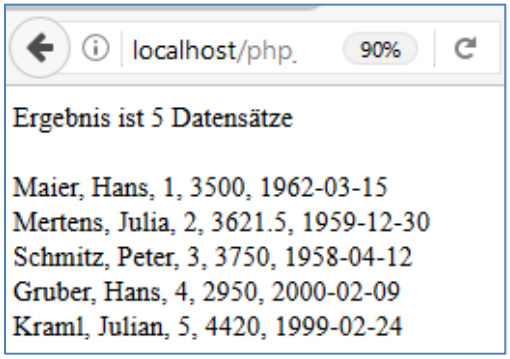
```
{
        echo    $dsatz["name"] . ", "
                        . $dsatz["vorname"] . ", "
                        . $dsatz["personalnummer"] . ", "
                        . $dsatz["gehalt"] . ", "
                        . $dsatz["geburtstag"] . "<br>";
}
```

Erklärung:

mysqli_fetch_assoc() Diese Funktion wird verwendet, um einen Datensatz des Ergebnisses zu ermitteln und ihn in einem assoziativen Feld (hier $dsatz) zu speichern. Dabei stellt der Datenbankfeldname den Schlüssel des Feldes dar. Die Funktion führt dazu, dass ein sogenannter Datensatzzeiger auf den nächsten Datensatz des Ergebnisses gesetzt wird.

while Die Zuweisung des Datenbankfeldes an das assoziative Feld „$dsatz“ wird gleichzeitig dazu verwendet, eine while-Schleife zu steuern. Diese dient dazu, alle Datensätze des Ergebnisses auszugeben.

Ergebnis:



2.2.2) Datensätze auswählen

Auswahl z.B. mit „ORDER BY“ oder „LIKE“.

Übung a:

Übung mit ausgewählten Feldern, WHERE-Klausel, Vergleichsoperatoren, logischen Operatoren und sortierter Ausgabe.

Arbeite einfach unter der oben erstellten Abfrage weiter.
Ziel: Es sollen alle Personen angezeigt werden, deren Gehalt zwischen 3.000 € und 3.700 € liegt, sortiert nach absteigendem Gehalt.

Beachte die Namen der Variablen, die natürlich sich nicht wiederholen sollen und somit einen anderen Namen tragen sollen, z.B. $res2.

Achtung: Beachte die Leerzeichen zwischen den einzelnen Angaben vor WHERE und vor ORDER BY, vor allem nach dem Verknüpfungspunkt. Ohne Leerzeichen nach dem Anführungszeichen, welches einem Punkt folgt, gibt es eine FEHLERMELDUNG!

```
26              . $dsatz["geburtstag"] . "<br>";
27  }
28
29  echo "<br>";
30
31  $sql = "SELECT name, gehalt FROM personen";
32  $sql .= " WHERE gehalt >= 3000 AND gehalt <= 3700";
33  $sql .= " ORDER BY gehalt DESC";
34
35  $res2 = mysqli_query($con, $sql);
36
37  while ($dsatz2 = mysqli_fetch_assoc($res2))
38  { echo $dsatz2["name"] . ", " . $dsatz2["gehalt"] . "<br>";
39  }
40
41  ?>
42  </body>
43  </html>
```

Beachte: Die Abfrage besteht aus einer längeren SQL-Anweisung. Aus Gründen der Übersichtlichkeit wird sie in mehreren Schritten in einer PHP-Variablen ($sql) gespeichert.

Ergebnis:

Mertens, 3621.5
Maier, 3500

Übung b

Beispiel mit dem LIKE-Operator.

Schreibe einfach unterhalb der oben durchgeführten Arbeiten weiter. Zur Abgrenzung füge ein: **echo „
“;**

LIKE: ist sehr nützlich beim Suchen nach Zeichenketten oder Teilen von Zeichenketten. Dabei können auch Platzhalter (Wildcards) eingesetzt werden. Ein % (Prozentzeichen) steht für eine beliebige Anzahl unbekannter Zeichen, ein _ (Unterstrich) für genau ein unbekanntes Zeichen. Die untersuchte Zeichenkette muss dabei weiterhin in einfache Hochkommata gesetzt werden.

Ziel: Es sollen alle Personen angezeigt werden, deren Name mit dem Buchstaben „M“ beginnt. Dabei ist besonders auf das Hochkomma bei „name LIKE `M%´ zu achten, da es sich bei dem Namen um eine Zeichenkette handelt.

Beachte die Namen der Variablen, die natürlich sich nicht wiederholen sollen und somit einen anderen Namen tragen sollen, z.B. $res3.

```
41 echo "<br>";
42
43 /*Nun folgt die LIKE-Abfrage*/
44 $sql_3 = "SELECT name, vorname FROM personen";
45 $sql_3 .= " WHERE name LIKE 'M%' ORDER BY name";
46
47 $res3 = mysqli_query($con, $sql_3);
48
49 while ($dsatz3 = mysqli_fetch_assoc($res3))
50     echo $dsatz3["name"] . ", " . $dsatz3["vorname"] . "<br>";
51
52 ?>
53 </body>
54 </html>
```

Ergebnis:

Maier, Hans
Mertens, Julia

2.3)Ausgabe in einer HTML-Tabelle

Eine Ausgabe wird in Tabellenform wesentlich übersichtlicher. Dafür muss man nur die HTML-Markierungen zur Erzeugung einer Tabelle an geeigneter Stelle in das PHP-Programm integrieren.

Ziel: Anzeige aller Datensätze der Tabelle „personen“ in der Datenbank „firma“ in Tabellenform mit Überschrift.

```
52 echo "<br>";
53
54 //Start für dei Tabellen-Ansicht:
55 $res4 = mysqli_query($con, "SELECT * FROM personen");
56 //Tabellenbeginn mit <table>
57 echo "<table border='1'>";
58
59 //Überschriften zwecks Übersichtlichkeit geteilt auf 2 Zeilen
60 echo "<tr> <td>Lfd. Nr.</td> <td>Name</td> <td>Vorname</td>";
61 echo "<td>Personalnummer</td> <td>Gehalt</td> <td>Geburtstag</td></tr>";
62
63 $positionsnummer = 1;
64 while ($dsatz = mysqli_fetch_assoc($res4))
65 {
66     echo "<tr> <td>$positionsnummer</td>";
67     echo "<td>" . $dsatz["name"] . "</td>";
68     echo "<td>" . $dsatz["vorname"] . "</td>";
69     echo "<td>" . $dsatz["personalnummer"] . "</td>";
70     echo "<td>" . $dsatz["gehalt"] . "</td>";
71     echo "<td>" . $dsatz["geburtstag"] . "</td>";
72     echo "</tr>";
73     $positionsnummer = $positionsnummer + 1;
74 }
75
76 //Ende mit </table>
77 echo "</table>";
78
79 ?>
80 </body>
81 </html>
```

Ergebnis:

Lfd. Nr.	Name	Vorname	Personalnummer	Gehalt	Geburtstag
1	Maier	Hans	1	3500	1962-03-15
2	Mertens	Julia	2	3621.5	1959-12-30
3	Schmitz	Peter	3	3750	1958-04-12
4	Gruber	Hans	4	2950	2000-02-09
5	Kraml	Julian	5	4420	1999-02-24

Erklärung: Es wird das Abfrageergebnis ermittelt. Dann folgt der Tabellenbeginn (<table border=“1“>) und eine Zeile mit einer Überschrift (<tr> bis </tr>). Innerhalb der Schleife wird zunächst zu den Feldinhalten eine laufende Positionsnummer ermittelt. Diese wird gemeinsam mit den Feldinhalten Zeile für Zeile ausgegeben.

2.4)Auswahl von Daten über ein Suchformular (html + php)

Ein Benutzer möchte selbst eine Auswahl treffen. Dies wird ihm durch die Eingabe von Werten in Formulare ermöglicht.

Typischer Ablauf einer Internetdatenbankanwendung:

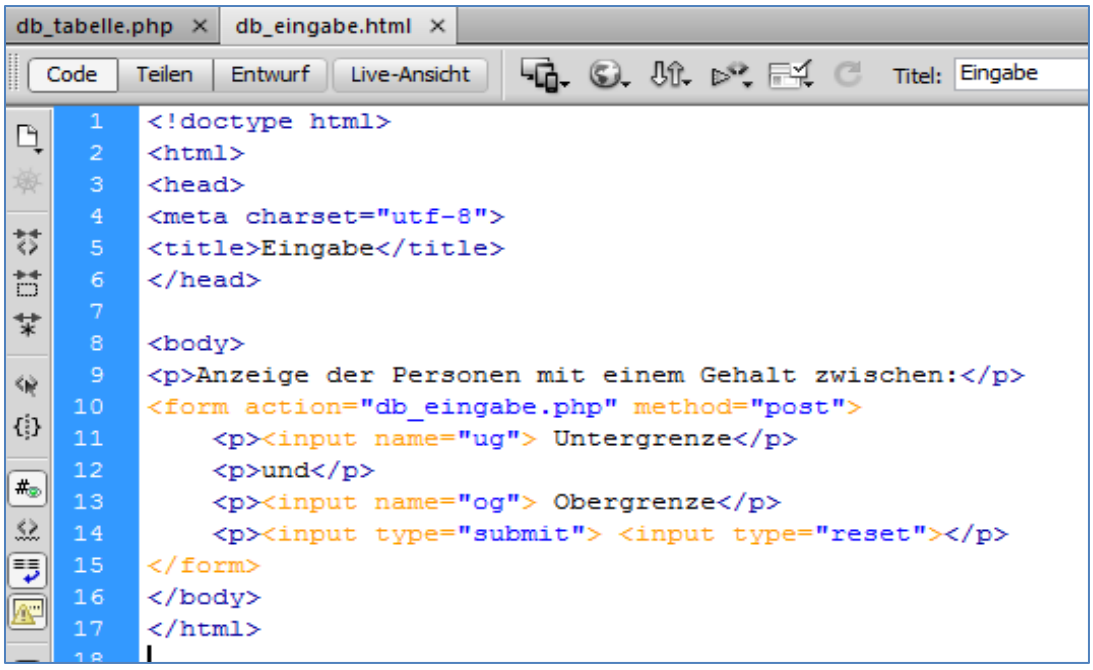
a) Der Benutzer gibt eine Anfrage ein, in dem er Daten in ein Formular einträgt und diese an den Webserver sendet.
b) Beim Webserver werden die Daten von einem PHP-Programm ausgewertet und mithilfe einer SQL-Anweisung an den Datenbankserver gesendet.
c) Der Datenbankserver ermittelt eine Antwort zur SQL-Anweisung und sendet diese an den Webserver zurück.
d) Das PHP-Programm verarbeitet die Antwort, kleidet sie in eine Internetseite und sendet dem Benutzer die Antwort.

Für den Benutzer ist es nicht sichtbar, welche Programme, Sprachen oder Dienste im Hintergrund für ihn tätig sind.

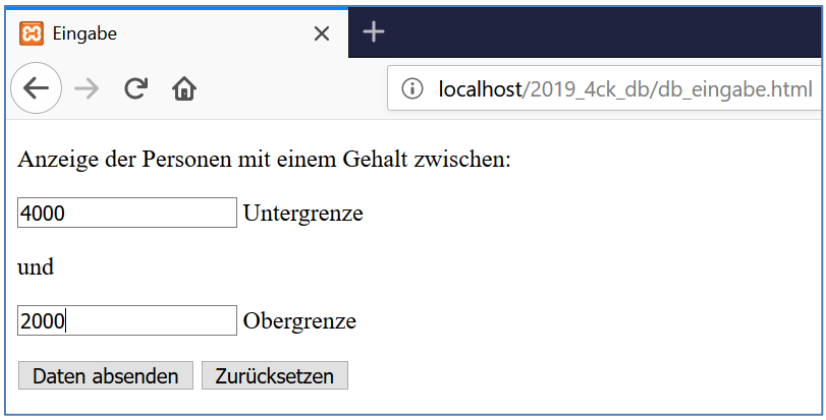
Beispiel Gehalt:

Der Benutzer soll zwei Zahlen eingeben, die als Ober- und Untergrenze dienen. Damit kann er festlegen, in welchen Bereich diese Gehaltsgruppenabfrage liegen soll.

Dafür erstelle zuerst das HTML-Formular mit dem Namen „**db_eingabe.html**“:



Innerhalb des Formulars werden die beiden Werte in die Eingabefelder mit dem Namen „ug“ und „og“ aufgenommen. Die Inhalte stehen dem PHP-Programm nach dem Absenden zur Verfügung.



Dann erstelle NEU die dazu passende PHP-Site mit dem Namen „db_eingabe.php“:
Dabei verwende wieder die Möglichkeit mit dem „include“.

BEACHTE: Beim Verwenden von Verkettung mit Punkt wird vor und nach dem Punkt immer ein Anführugszeichen gesetzt. VORALLEM NACH dem Punkt MUSS nach dem Anführungszeichen ein LEERZEICHEN verwendet werden. Ansonsten kommt es zu einer Fehlermelung!!!

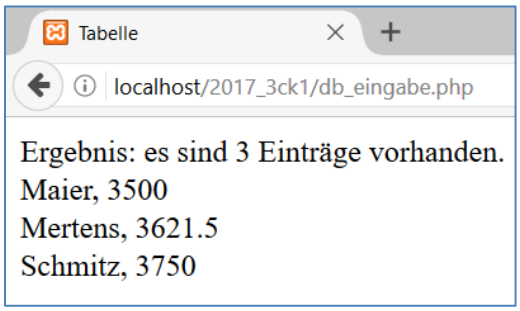
Beispiel:

```
$sql = "SELECT name, gehalt FROM personen" . " WHERE gehalt >= " .
$_POST["ug"] . " AND gehalt <= " . $_POST["og"] . " ORDER BY gehalt";
```

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>Tabelle</title>
</head>
<body>
<?php
/*Verbindung aufnehmen*/
$con = mysqli_connect("localhost", "root", "", "schueler");

$sql = "SELECT name, gehalt FROM personen"
        . " WHERE gehalt >= " . $_POST["ug"]
        . " AND gehalt <= " . $_POST["og"]
        . " ORDER BY gehalt";

$res = mysqli_query($con, $sql);
$num = mysqli_num_rows($res);
if($num > 0)    echo "Ergebnis:<br>";
else            echo "Keine Ergebnisse<br>";

while ($dsatz = mysqli_fetch_assoc($res))
    echo $dsatz["name"] . ", " . $dsatz["gehalt"] . "<br>";

mysqli_close($con);
?>
</body>
</html>
```

Innerhalb der SQL-Anweisung findet sich nach der WHERE-Klausel die Feldelemente $_POST[„ug“] und $_POST[„og“], um die auszugebenden Datenmenge einzuschränken. Die Feldelemente beinhalten die beiden Eingabewerte des Benutzers.

Ergebnis bei Eingabe 3000 – 4000:

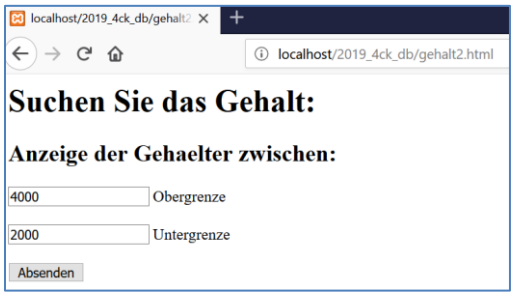


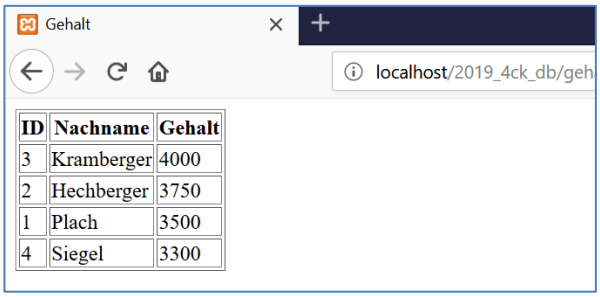
HÜ2: Datenbank „schueler“ – Abfrage Unter- und Obergrenze

Erstelle die Abfrage wie oben angeführt aber mit Ausgabe in eine schöne Tabelle. Es sollen in der Tabellenkopfzeile die ID, der Nachname und der Gehalt ausgegeben werden.

```
5   </html>
6   <?php
7   include "verbindung.php";
8
9   $erg = $con->query("SELECT s_id,nachname, gehalt FROM schueler
    WHERE gehalt >= " . $_POST['unter'] . " AND gehalt <= " .
    $_POST['ober'] . " ORDER BY gehalt DESC");
10
11  echo "<table border='1'>";
12  echo "<tr><th>ID</th><th>Nachname</th><th>Gehalt</th></tr>";
13  while($zeile = $erg->fetch_array()){
14      echo "<tr>
15              <td>{$zeile['s_id']}</td>
16              <td> {$zeile['nachname']} </td>
17              <td> {$zeile['gehalt']} </td>
18          </tr>";
19  }
20  echo "</table>";
21  ?>
```

Ergebnis:

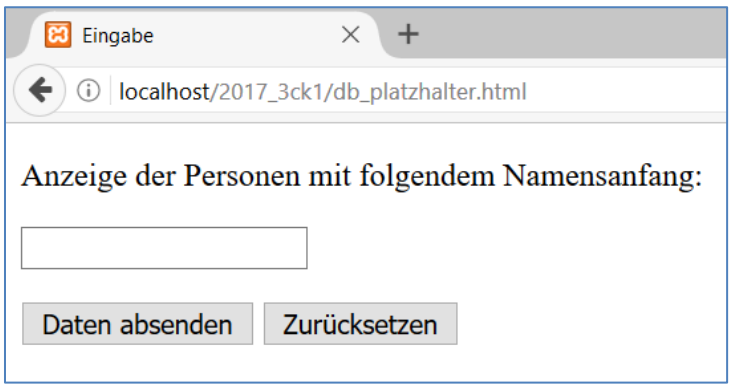




Übung:

Erstelle ein Formular mit dem der Benutzer nach Personen suchen kann, deren Anfangsbuchstaben mit dem angegebenen Buchstaben beginnen.
Speichere diese als „db_platzhalter.html“ und „db_platzhalter.php“.

So soll die „db_platzhalter.html“ aussehen:



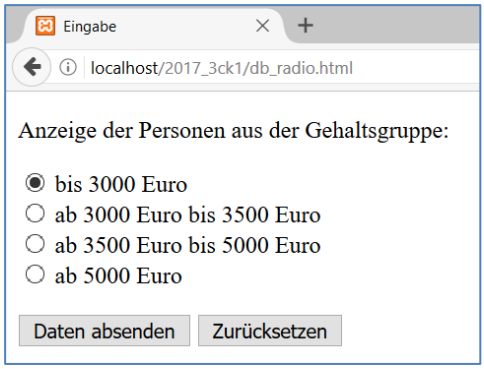
Ergebnis bei Eingabe von „m“:

2.5)Auswahl mit Radiobuttons aus der Datenbank „schueler“

Eine Abfrage kann dem Benutzer durch die Verwendung von diversen Formularelementen erleichtert werden.

Beispiel:
Mithilfe von Radiobuttons kann der Benutzer aus bestimmten Gehaltsgruppen wählen.

Erstelle das HTML-Formular „db_radio.html“




```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Eingabe</title>
</head>

<body>
<p>Anzeige der Personen aus der Gehaltsgruppe:</p>
<form action="db_radio.php" method="post">
    <p> <input type="radio" name="geh" value="1" checked="checked"> bis 3000 Euro
        <br>
        <input type="radio" name="geh" value="2"> ab 3000 Euro bis 3500 Euro <br>
        <input type="radio" name="geh" value="3"> ab 3500 Euro bis 5000 Euro <br>
        <input type="radio" name="geh" value="4"> ab 5000 Euro </p>
        <p><input type="submit"> <input type="reset"></p>
</form>
</body>
</html>
```

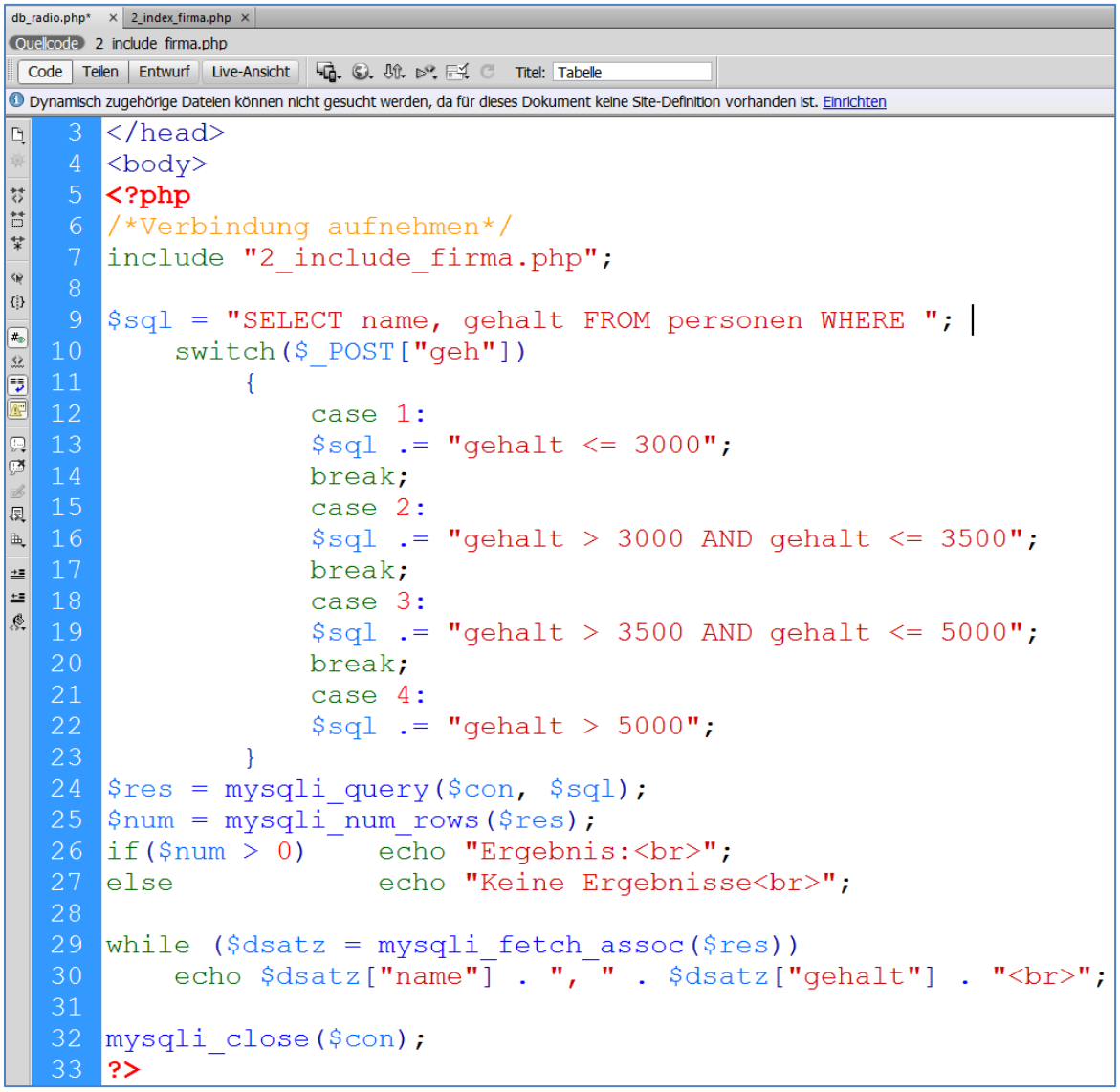
Code:

```
<p>Anzeige der Personen aus der Gehaltsgruppe:</p>
<form action="db_radio.php" method="post">
        <p><input type="radio" name="geh" value="1" checked="checked"> bis 3000 Euro <br>
        <input type="radio" name="geh" value="2"> ab 3000 Euro bis 3500 Euro <br>
    <input type="radio" name="geh" value="3"> ab 3500 Euro bis 5000 Euro <br>
    <input type="radio" name="geh" value="4"> ab 5000 Euro </p>
    <p><input type="submit"> <input type="reset"></p>
</form>
```

Ergebnis:

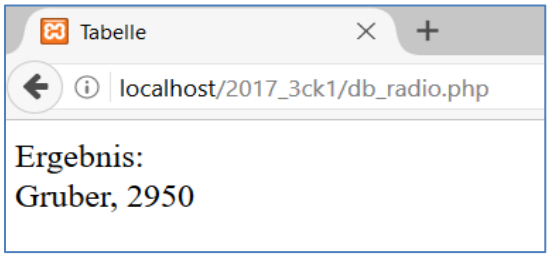
Die verwendeten Radiobuttons haben alle den gleichen Namen („geh“), dadurch bilden sie eine Gruppe und somit kann immer nur ein Button ausgewählt werden. Der vom Benutzer ausgewählte Button ist mit dem Wert 1,2,3 oder 4 verbunden. Dieser Wert wird dem PHP-Programm beim Absenden übermittelt.

db_radio.php:



Info: switch https://www.w3schools.com/php7/php7_switch.asp

Das übermittelte Feldelement $_POST[“geh“] wird mithilfe einer switch/case-Verzweigung untersucht. Je nach Wert des Feldelements wird eine von mehreren möglichen SQL-Anweisungen gebildet. Diese wird ausgeführt und liefert die gewünschten Daten.



Quellen:

Florence Maurice in PHP 5.6 und MySQL 5.7, dpunkt Verlag, 2015, S.357-376

Giesbert Damaschke in PHP & MySQL, Wiley-Vch Verlag, 2015, S.281-288
und S.284-285, S.319-321