Registrieren mit Formular inkl. Bild - in PHP mit Datenbank

PHP, Schüler, Datenbank – Teil 2a

- Formular - registrieren.php erstellen
- Daten von registrieren.php nach registrieren2.php übergeben
- registrieren2.php mit INSERT INTO - Daten aus dem Formular in die Datenbank schreiben

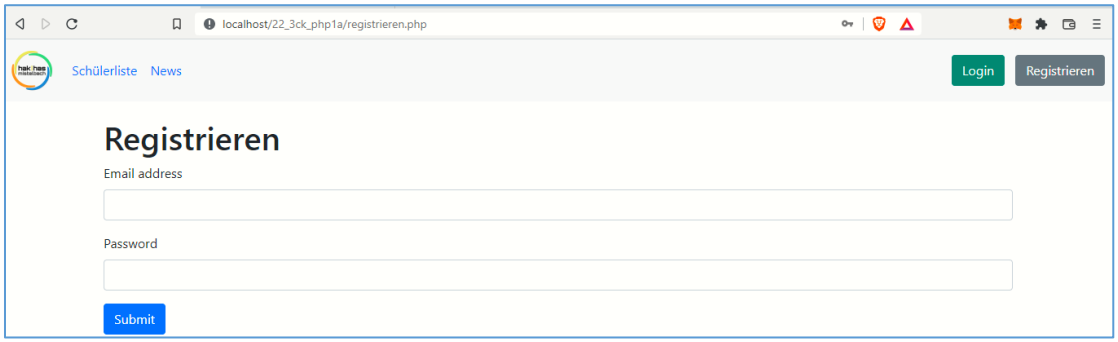
1)Formular erstellen - registrieren.php

Kopiere die oberen Zeilen von 2schueler.php“ und verwende sie in der neu geschaffenen „registrieren.php“:

Erstelle ein einfaches Formular innerhalb des „containers“ unter der Überschrift. Nutze eventuell aus https://getbootstrap.com/docs/5.2/forms/overview/ das Basis-Formular und ändere es entsprechend.

- Entfernen der „emailHelp“
- Entfernen der „checkbox“
- Im „label“ das „for“ und im „input“ die „id“ auf das Feld beziehen. Das in kurzen Begriffen. Wenn diese beiden ident sind, dann ist das für den User ein Vorteil, weil er auch durch Antippen der Feldüberschrift (=Label) das Inputfeld aktiviert und somit befüllen kann. Das ist besonders für die Verwendung am Smartphone praktisch.

Ergebnis:



- Die Klasse „class=“mb-3“ bedeutet hier nur, dass der Abstand zum „bottom“ 3 Pixel ist und somit nicht so sehr am nächsten Feld klebt.

```
<form >
  <div class="mb-3">
    <label for="email" class="form-label">E-Mail</label>
    <input type="email" class="form-control" id="email" name="email">
  </div>

  <div class="mb-3">
    <label for="pw" class="form-label">Passwort</label>
    <input type="password" class="form-control" id="pw" name="pw">
  </div>

  <button type="submit" class="btn btn-primary" >Submt</button>
</form>
```

Beachte:

- Name für E-Mail und Passwort nicht vergessen
- Auf Deutsch umstellen – im Label (E-Mail und Passwort)

2)Elemente gemäß Datenbank-Elemente aufbauen

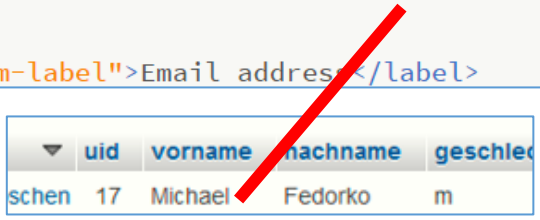
In der Datenbank sind folgende Elemente zu befüllen:

	uid	vorname	nachname	geschlecht	geb	email	passwort	bild
schen	17	Michael	Fedorko	m	2004-03-21	mf@gmail.com	1234	[BLOB - 71,7 KiB]

Kopiere die vom Passwort und füge es mehrfach darunter ein.

Jetzt muss man genau arbeiten und Änderungen durchführen, jeweils passend zum benötigten neuen Element. Diese sollen sich an den Elementen der Datenbank orientieren.
BEACHTE: auf jeden Fall jeweils einen „name“ in das input-Feld einfügen, der GENAU die gleiche Bezeichnung trägt wie der Eintrag in der Datenbank. Das ist eine Erleichterung für die spätere Übergabe an die Folgedatei, die nach dem Klick auf den „Absenden- bzw. Registrieren-Button“ aufgerufen werden wird.

```
<form>
  <div class="mb-3">
    <label for="vn" class="form-label">Vorname</label>
    <input type="text" class="form-control" id="vn" name="vorname">
  </div>
  <div class="mb-3">
    <label for="email" class="form-label">Email addres</label>
```



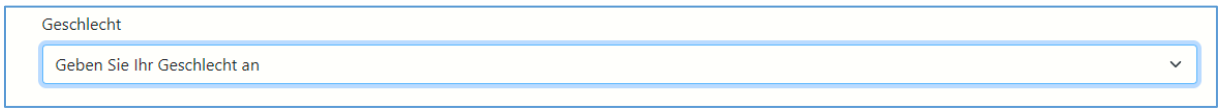
- for im label
- id im input
- Type auf „text“ umstellen
- name = NEU

Geschlecht:

- statt eines „input-Feldes“ benötigt man „select“ und „option“ mit „value“.
- Der Wert innerhalb des „value“ ist wichtig, da er später übergeben wird
- In der ersten <option> steht „selected“, was bedeutet, dass das angezeigt wird. Hier passend mit „Geben Sie Ihr Geschlecht an“

```
23 ▼  <div class="mb-3">
24      <label class="form-label">Geschlecht</label>
25 ▼    <select class="form-select"  type="text" name="geschlecht" >
26          <option selected>Geben Sie Ihr Geschlecht an</option>
27          <option value="w">w (weiblich)</option>
28          <option value="m">m (männlich)</option>
29      </select>
30    </div>
```

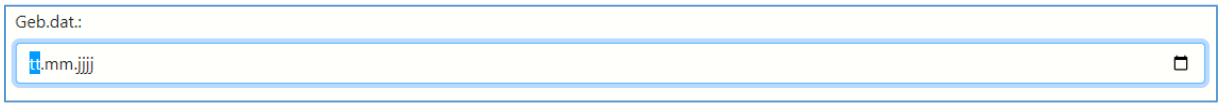
Ergebnis:



Geburtsdatum:

```
32  <!--    Datum-->
33 ▼    <div class="mb-3">
34      <label class="form-label" for="date">Geb.dat.:</label>
35      <input type="date" name="geb" class="form-control" id="date">
36      </div>
```

Ergebnis:



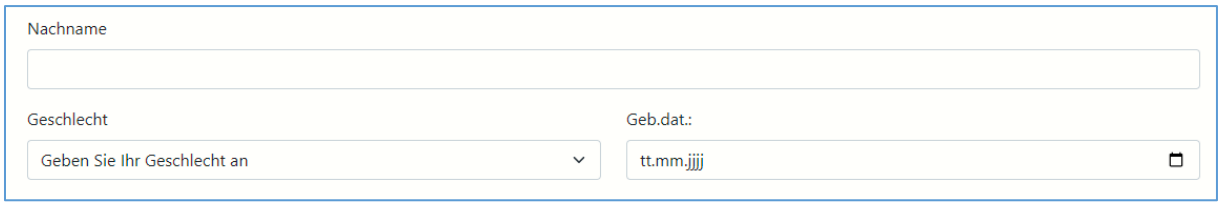
Verbesserung:

Geschlecht und Datum benötigen nicht so viel Platz. Sie könnten nebeneinander in einer Zeile stehen und sich diese gerecht teilen:

Dafür muss man beide in Bootstrap in eine <row> geben und jeden einzelnen Bereich in eine <col>. Wenn man keine Zahl dazuschreibt, wie z.B. col-4, dann teilen sich die Elemente den Platz gerecht auf. Das ist ok so.

```
23
24      <div class="row"> <!-- eine Zeile für 2 Elemente -->
25      <div class="col"> <!-- start erste col-->
26    <!--    Geschlecht-->
27            <div class="mb-3">
28              <label class="form-label">Geschlecht</label>
29              <select class="form-select"  type="text" name="geschlecht" >
30                  <option selected>Geben Sie Ihr Geschlecht an</option>
31                  <option value="w">w (weiblich)</option>
32                  <option value="m">m (männlich)</option>
33              </select>
34            </div>
35      </div> <!-- ende col-->
36      <div class="col">  <!-- start zweite col-->
37    <!--    Datum-->
38      <div class="mb-3">
39      <label class="form-label" for="date">Geb.dat.:</label>
40      <input type="date" name="geb" class="form-control" id="date">
41      </div>
42      </div> <!-- ende col-->
43      </div> <!-- ende row-->
```

Ergebnis:



E-Mail und Passwort ebenfalls in eine Zeile geben

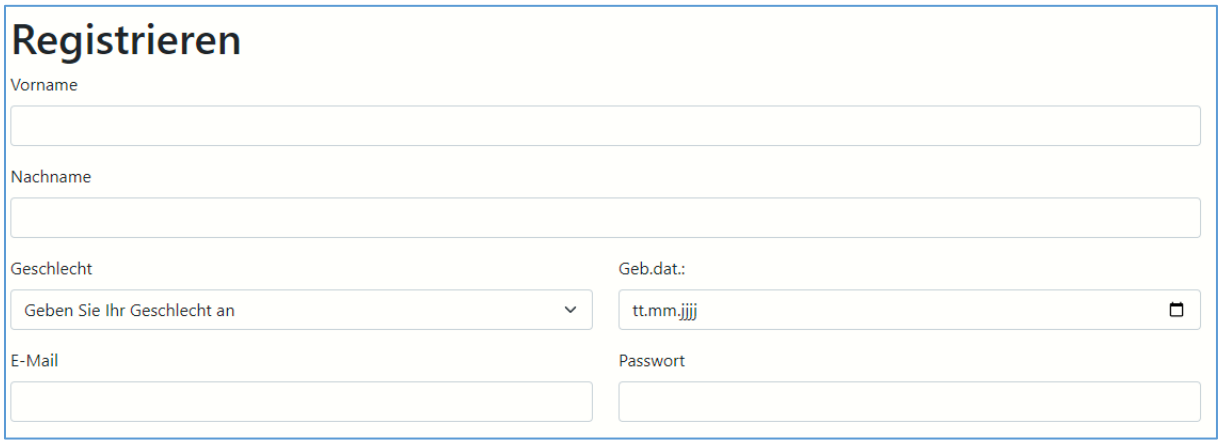
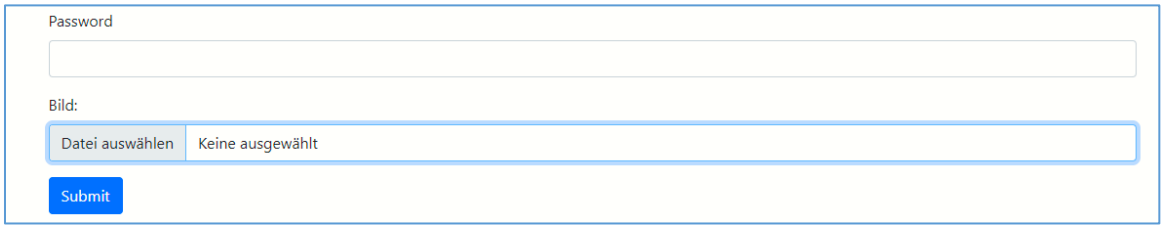


Bild:

Ziel: unterhalb vom Passwort und vor dem „Submit“-Button füge den Bild-Upload ein:



```
53  <!--    Bild auswählen lassen-->
54      <div class="mb-3">
55      <label class="form-label" >Bild:</label>
56      <input type="file" name="bild" class="form-control" enctype="multipart/form-data" >
57      </div>
```

- type muss „file“ sein
- wichtig: enctype="multipart/form-data"

Abschicken Button:

- auch dieser Button soll einen Namen erhalten, den wir später für die Überprüfung benötigen, ob der Button wirklich korrekt betätigt wurde.
- Er soll nicht Submit sondern „Registrieren“ lauten

```
59      <br>
60    <button type="submit" class="btn btn-primary"
      name="abschicken_button">Registrieren</button>
61  </form>
```

Formular weiterleiten

Am Beginn des Formulars müssen die Daten eingefügt werden, die reagieren, wenn der User den „Submit“-Button drückt. Das sind sehr wichtige Informationen:

- wohin werden die Daten gesendet - ACTION
- mit welcher Methode – POST oder GET
- wenn ein Bild oder PDF mitgesendet werden sollen, muss noch das enctype vorhanden sein: enctype="multipart/form-data

```
12  <h1>Registrieren</h1>
13
14  <form method="post" action="functions/registrieren2.php"
15        enctype="multipart/form-data">
16    <div class="mb-3">
17      <label for="vn" class="form-label">Vorname</label>
```

Daher wird später eine Datei im Ordner „functions“ zu erstellen sein, die „registrieren2.php“ heißen wird.

Zur Erinnerung:
Damit das Bild übergeben werden kann muss man an 2 Stellen diesen Code verwenden:

enctype="multipart/form-data"

1. Gleich am Beginn im <form>-Bereich.
2. Und direkt im Input-Feld beim Aufrufen der Bilddatei.

3)registrieren2.php – Daten in die Datenbank schreiben

Nun muss aber auch die Datei, welche die Daten aus den ausgefüllten Input-Felder aus dem Formular annimmt und in die Datenbank übertragen soll.

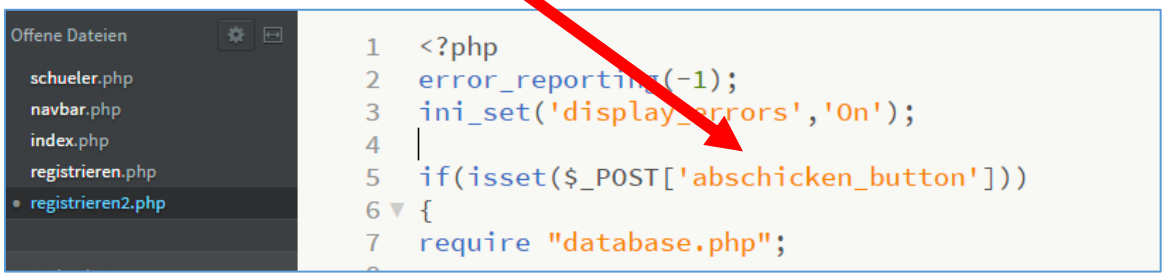
Erstelle diese im Ordner „functions“, da sie mit der Datenbank agiert, aber nicht vom Benutzer direkt befüllt bzw. ausgefüllt werden wird.

3a)Sicherheit – wurde der Submit-Button gedrückt?

Damit kein Betrug erfolgen kann, sollen die Daten nur dann weiterverarbeitet werden, wenn der User auch wirklich den Submit-Button im Formular gedrückt hat. Ein direkter Aufruf in der URL soll somit ins Leere führen.

- IF-Verzweigung erlaubt den Zugriff auf die „database.php“ nur wenn „true“
- isset – bedeutet, ob der Wert in den Klammern „gesetzt wurde“?
- mit $_POST[] wird ein Wert, der übergeben wird, übernommen
- der Button im Formular hat den entsprechenden Namen. Vergleiche diesen:

```
66      <button type="submit" class="btn btn-primary"
        name="abschicken_button">Registrieren</button>
67  </form>
```



Ansonsten wird eine Fehlermeldung ausgegeben:

```
 7  require "database.php";
 8  }
 9  else
10  {
11      echo "Bitte korrekt ausfüllen.";
12  }
13  ?>
```

3b)Variablen erstellen

- Variablen erstellen, die mit der Methode (method=“POST“) die per „Submit-Button“ weitergeleiteten Daten („name“ im Input-Feld) befüllt werden
- Meist verwendet man die gleichen Variablen-Namen wie auch die Bezeichnung der „name“ in den Inputfeldern des Formulars, welche in den eckigen Klammern der Variable „$_POST“ übernommen werden.

NUR: die Variable des Passworts verträgt kein Sonderzeichen, weswegen könnte diese Variable

nicht „$pwd#“ lauten könnte, sondern „$pw“ lautet. Der „name“ im Formular kann gerne eine Raute # aufweisen.

```
 7  require "database.php";
 8
 9  $vorname = $_POST['vorname'];
10  $nachname = $_POST['nachname'];
11  $geschlecht = $_POST['geschlecht'];
12  $geb = $_POST['geb'];
13  $email = $_POST['email'];
14  $pw = $_POST['pw'];
15  // $bild = $_POST['bild']; geht so nicht, daher ->
16  $bild = addslashes(file_get_contents($_FILES['bild']['tmp_name']));
```

Da jedoch die Übernahme des Bildes aus dem Eingabeformular so einfach nicht funktioniert, muss man folgenden Code verwenden:

$bild = addslashes (file_get_contents($_FILES['bild']['tmp_name']));

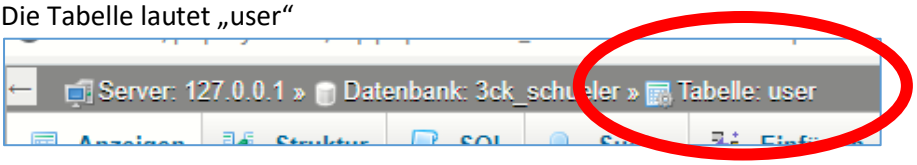
Info: später, wenn man eine gewisse Sicherheit einbaut, muss man „addslashes“ weglassen!!! (prepared statements)

In PHP werden SQL-Abfragen durch Strings repräsentiert. Es ist üblich, dass man Abfragen in einer Variablen speichert, bevor man sie an die Funktion $con->query () übergibt.

- $sql ist die PHP-Variable, die den String mit der Abfrage festhält

Danach folgt der SQL-Befehl, der die Daten aus den eben erstellten Variablen in die Datenbank schieben möchte:

- INSERT INTO
- Die Tabelle lautet „user“



- Die Bezeichnungen in der folgenden Klammer müssen EXAKT mit den Elementen in der Datenbank übereinstimmen. Eine Abweichung würde dazu führen, dass keine Speicherung möglich wäre.
- Daher vergleiche nochmal die Elemente aus der Datenbank direkt

	uid	vorname	nachname	geschlecht	geb	email	passwort	bild
schen	17	Michael	Fedorko	m	2004-03-21	mf@gmail.com	1234	[BLOB - 71,7 KiB]

- Nicht vergessen, die einfachen Anführungszeichen zu setzen bei jeder Variablen in den VALUES

```
18
19  $sql = "INSERT INTO user (vorname, nachname, geschlecht, geb, email, passwort, bild)
20      VALUES ('$vorname','$nachname','$geschlecht','$geb','$email','$pw','$bild')";
21
```

Im INSERT INTO müssen immer die korrekten Bezeichnungen stehen, die auch so in der Datenbank verwendet werden. Die Variablen des „values“ sind die, die wir ein paar Zeilen darüber erstellt haben und müssen ebenfalls passen. Zum Glück werden sie beim Beginn der Eingabe in manchen Editoren automatisch vorgeschlagen.

Nachdem die INSERT-Abfrage in einem String gespeichert ist, ist man bereit, sie an die Funktion mysqli_query() zu übergeben, um die Einfüge-Operation tatsächlich durchzuführen.

$ergebnis = $con->query ($sql)
or die("Fehler bei der Datenbankabfrage.");

```
21
22  $ergebnis = $con->query ($sql)
23      or die("Fehler bei der Datenbankabfrage.");
24
```

Beachte: Kein Strichpunkt nach der ersten Zeile, da noch die Anweisung „or die“ folgt.

Info: Die Funktion $con->query() braucht zwei Informationen, um eine Abfrage durchzuführen: eine
1. Datenbankverbindung und einen
2. SQL-Abfrage-String.

Ad_1) Die Datenbankverbindung haben wir in “$con“ gelegt.
Ad_2) Den SQL-Abfrage-String haben wir gerade erstellt und “$sql“ getauft und ebenfalls in PHP geschrieben.

Dieser Code zeigt, dass ein Aufruf von $con->query() keine einseitige Angelegenheit ist. Auf den Aufruf der Funktion werden Informationen zurückgeliefert, die in der Variablen $ergebnis gespeichert werden. Auf eine INSERT-Abfrage wird aber kein richtiges Ergebnis zurückgeliefert – die Variable $ergebnis speichert nur, ob die Abfrage erfolgreich war oder nicht.

Somit werden die in das Formular eingegebenen Daten in die Datenbank übermittelt.

Nun sollte man einmal TESTEN.

	uid	vorname	nachname	geschlecht	geb	email	passwort	bild
hen	17	Michael	Fedorko	m	2004-03-21	mf@gmail.com	1234	[BLOB - 71,7 KiB]
hen	18	Felix	Vollmair	m	2004-01-13	felix@live.at	1234	[BLOB - 59,0 KiB]
hen	19	Josef	Eberhart	m	2022-08-02	josefeberhart@gmx.at	1234	[BLOB - 55,5 KiB]

Funktioniert.

Zusätzliche Theorie: Daten schreiben mit „INSERT INTO“

Die Details – also welche Daten in welche Felder von welcher Tabelle eingefügt werden sollen – werden über verschiedene Parameter definiert. So wird mit INTO die Tabelle bestimmt, und mit VALUES legt man die einzufügenden Werte fest.

Da jede Tabelle auch einen Primärschlüssel hat, der automatisch hochgezählt wird, sollte man diesen nicht beeinflussen. Daher beginnt man in der Klammer nach dem Namen der Tabelle (hier: test) mit den Namen des ersten Feldes, das einen neuen Wert aufnehmen soll.

Diese Feldnamen werden nach dem Tabellennamen in Klammern genannt und durch ein Komma voneinander getrennt. Soll mehr als ein Datensatz eingefügt werden, werden die Daten der einzelnen Datensätze ebenfalls durch ein Komma getrennt. Das klingt in der Beschreibung vielleicht ein wenig seltsam, ist aber in der Praxis ganz einfach. In seiner allgemeinen Form sieht das etwa so aus:

```
INSERT INTO test (feld1, feld2, feld3)
  VALUES (wert1, wert2, wert3), (wert4, wert5, wert6), (wert7, wert8, wert9)
```

Beispiel:

```
INSERT INTO test (name, vorname)
  VALUES ('Lebeau', 'Madeleine'), ('Wilson', 'Dooley'), ('Page', 'Joy')
```

Wie bei `SELECT` und `UPDATE` wird auch dieses Kommando mit der Methode `query()` an den Datenbankserver geschickt. Das Datenbankobjekt wird hier z.B. in der Variablen `$sql` abgelegt:

```
$insert = "INSERT INTO test (name, vorname)
  VALUES  ('Lebeau', 'Madeleine'), ('Wilson', 'Dooley'), ('Page', 'Joy')";
$sql->query($insert);
```

Dieses INSERT fügt der Tabelle „test“ neue Datensätze ein.

5.)Die Verbindung mit „mysqli_close()“ schließen.

Da wir nur die eine INSERT-Abfrage durchführen möchten, ist für uns, d.h. das Skript, die Datenbankinteraktion beendet. Und wenn man eine Datenbankverbindung nicht mehr benötigt, sollte man sie schließen. Datenbankverbindungen werden automatisch geschlossen, wenn der Benutzer die Seite verlässt, aber ebenso, wie es sich gehört,
die Tür zu schließen, durch die man gerade gekommen, gehört es sich, eine Datenbankverbindung zu schließen, die man geöffnet hat.
Die PHP-Funktion mysqli_close() schließt eine MySQL-Datenbankverbindung.

```
mysqli_close($con);
```

Warum sollte man die Verbindung schließen?
Verbindungen benötigen Ressourcen, und mit Ressourcen sollte man **sparsam** und verantwortungsbewusst umgehen.
Der Datenbankserver muss für die Verbindung Ressourcen (Speicherplatz, Rechenzeit usw.) bereitstellen. Da seine physischen Ressourcen beschränkt sind, muss er die Anzahl von Verbindungen beschränken, die er zu einem Zeitpunkt annehmen kann. Ist diese Zahl erschöpft, kann er keine weiteren Verbindungen annehmen. Um die Gefahr zu reduzieren, dass Skripten aus diesem Grund keine Verbindung aufbauen können und fehlschlagen, sollte man die geöffneten Verbindungen schließen, sobald man sie nicht mehr benötigen.

Formular mit Bild an DB übergeben:

https://www.youtube.com/watch?v=hklOcrUhggg