# M151 Mirio Eggmann

## Daten modifizieren

```
UPDATE personen
SET hobbies = ‚Musik, Kino, Sport'
WHERE vorname = ‚Anna' AND nachname= ‚Meier'

UPDATE personen
SET hobbies = ‚Segeln, Sport, Essen'
WHERE vorname= ‚John' AND nachname='Meyer';
```

## Verknüpfung von Bedingungen

```
SELECT * FROM personen
WHERE jahrealt > 18 AND jahrealt < 30;

SELECT * FROM personen
WHERE jahrealt < 20 OR jahrealt > 40;

SELECT * FROM personen
WHERE NOT ort = ‚Zürich' AND jahrealt < 30;

SELECT * FROM personen
WHERE ort = ‚Zürich' OR ( ort = ‚Bern' AND jahrealt < 30);
```

## ▪ Verbindung·aufbauen¶

Hinweis:→ Der·Pfad·von·mysql.exe·muss·in·der·Umgebungsvariablen·PATH·von·Windows·vorhanden·sein,·ansonsten·müssen·Sie·ins·Verzeichnis·C:\xampp\mysql\bin·wechseln,·um·mysql.exe·aufrufen·zu·können. → ¶

```
Set·PATH=%PATH%;C:\xampp\mysql\bin¶

C:\>mysql·-h·localhost·-u·root·–p¶
```

## ▪ MySQL-Verbindung·beenden

```
mysql>·quit;¶
```

## Spalte entfernen

```
ALTER TABLE personen DROP jahrealt;
```

## Tabelle löschen

```
DROP TABLE personen;
```

## Tabellen anzeigen

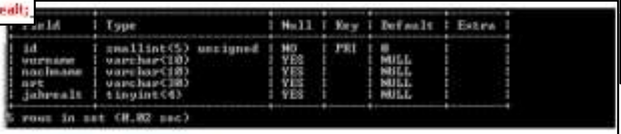
```
SHOW TABLES;
```

## Datenbank erstellen

```
CREATE DATABASE db_test;
```

## Tabellenstruktur anzeigen

```
DESCRIBE personen;
```



```
+----------+----------------------+------+-----+---------+-------+
| Field    | Type                 | Null | Key | Default | Extra |
+----------+----------------------+------+-----+---------+-------+
| id       | smallint(5) unsigned | NO   | PRI | 0       |       |
| vorname  | varchar(10)          | YES  |     | NULL    |       |
| nachname | varchar(10)          | YES  |     | NULL    |       |
| ort      | varchar(30)          | YES  |     | NULL    |       |
| jahrealt | tinyint(4)           | YES  |     | NULL    |       |
+----------+----------------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

## Datentyp einer Spalte ändern

```
ALTER TABLE personen MODIFY vorname VARCHAR(20);
ALTER TABLE personen MODIFY nachname VARCHAR(20);
```

## Gruppierung

```
SELECT ort,COUNT(ort) AS ‚Anzahl' FROM personen
GROUP BY ort;

SELECT ort, COUNT(ort) AS ‚Anzahl' FROM personen
GROUP BY ort
HAVING Anzahl >= 2 ;
```

## Datenbanken auflisten

```
SHOW DATABASES;
```

## Anzahl der Datensätze ermitteln

```
SELECT COUNT(*) AS ‚Anzahl' FROM personen;
```

## Tabelle *personen* anpassen

```
UPDATE personen
SET ort='1'
WHERE ort= ‚Bern';

UPDATE personen
SET ort='2'
WHERE ort='Zürich';

ALTER TABLE personen MODIFY ort TINYINT(4);
```

## Benutzer erstellen

Benutzer auflisten: SELECT * FROM mysql.user; OR desc mysql.user;

```
CREATE USER ‚gibbix'@'localhost' IDENTIFIED BY ‚gibbiX12345';
```

## Summierung und Durchschnitt

```
SELECT SUM(jahrealt) AS ‚Alterssumme' FROM personen;

SELECT AVG(jahrealt) AS ‚Altersdurchschnitt' FROM personen;
```

## Daten einfügen

```
INSERT INTO personen (id, vorname, nachname, jahrealt, ort)
VALUES
(1,'Anna','Meier',19,'Bern'),
(2,'John','Meyer',48,'Bern'),
(3,'Kevin','Müller',25,'Zürich');
```

## Abfrage mit Platzhaltern

```
SELECT * FROM personen
WHERE hobbies LIKE ‚%Musik%';

SELECT * FROM personen
WHERE nachname LIKE ‚Me_er';     <--- grundsätzlich nicht ganz korrekt (nicht nur i/y)
```

## Tabelle erstellen

```
CREATE TABLE personen
(
id INT NOT NULL AUTO_INCREMENT,
vorname VARCHAR(10) ,
nachname VARCHAR(10),
ort VARCHAR(30),
jahrealt INT,
PRIMARY KEY(id)
);
```

## Sortierung

```
SELECT * FROM personen
ORDER BY jahrealt asc;

SELECT * FROM personen
ORDER BY jahrealt desc;
```

## Benennung von Spalten

```
SELECT id, vorname, nachname, ort, jahrealt AS ‚alter', hobbies FROM personen;
```

## Abfragen erst

```
SELECT * FROM personen;

SELECT vorname,nachname FROM personen;
```

## Spalte einfügen

```
ALTER TABLE personen ADD hobbies VARCHAR(100);
```

## Einfache Abfrage mit Herausfiltern von Duplikaten

```
SELECT ort FROM personen;

SELECT DISTINCT(ort) FROM personen;
```

## Minimal- und Maximalwert

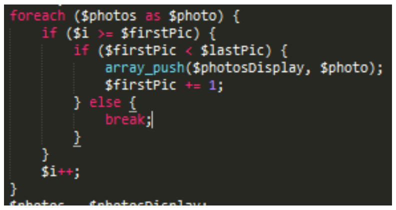
```
SELECT * FROM personen
WHERE jahrealt = (SELECT MIN(jahrealt) FROM personen); ← wäre für die person...

SELECT MIN(jahrealt) AS ‚Jüngste' FROM personen;

SELECT MAX(jahrealt) AS ‚Älteste' FROM personen;
```

## Daten löschen

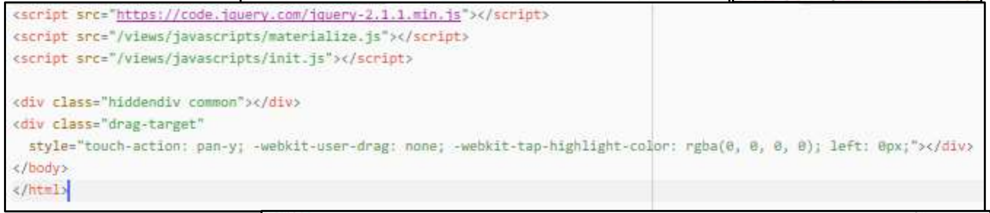
```
DELETE FROM personen
WHERE vorname='kevin' AND nachname='müller';

INSERT INTO personen (id, vorname, nachname, jahrealt, ort, hobbies)
VALUES
(3,'Kevin','Müller',25,'Zürich','Musik, Freunde, Ausgang');
```

## Abfrage mit Bedingung

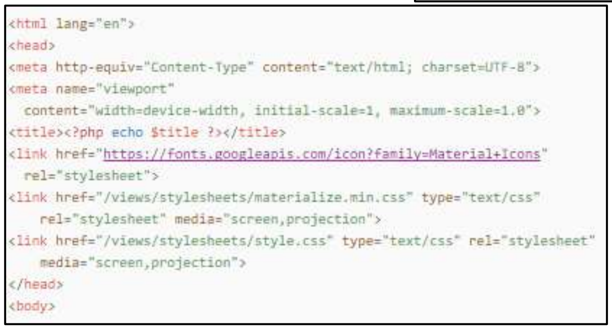
```
SELECT * FROM personen
WHERE id = 1;

SELECT * FROM personen
WHERE ort = ‚Bern';

SELECT * FROM personen
WHERE jahrealt BETWEEN 18 AND 30;
```



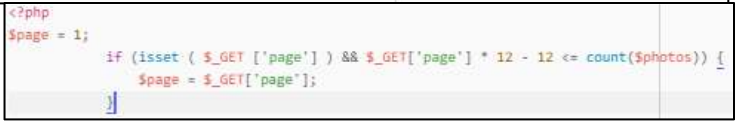
```
foreach ($photos as $photo) {
    if ($i >= $firstPic) {
        if ($firstPic < $lastPic) {
            array_push($photosDisplay, $photo);
            $firstPic += 1;
        } else {
            break;
        }
    }
    $i++;
}
$photos = $photosDisplay;
```



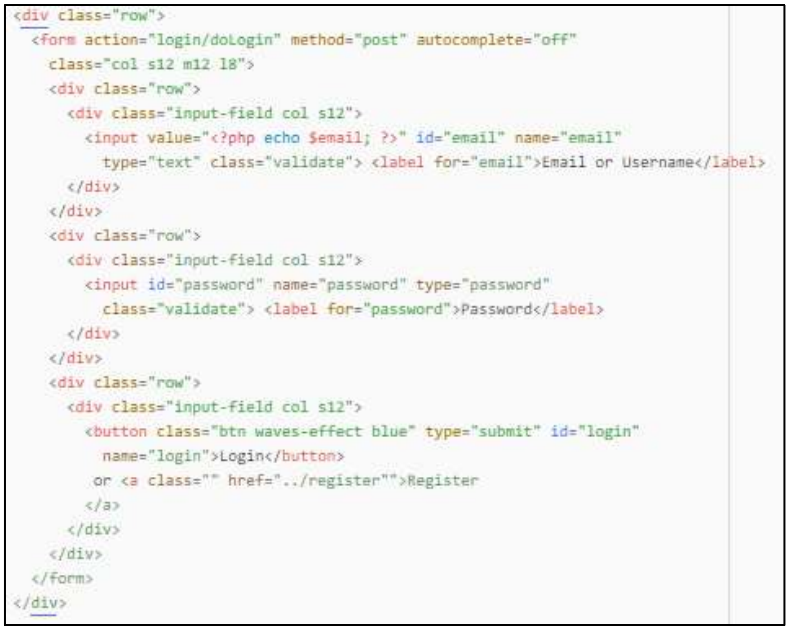
```
<script src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
<script src="/views/javascripts/materialize.js"></script>
<script src="/views/javascripts/init.js"></script>

<div class="hiddendiv common"></div>
<div class="drag-target"
  style="touch-action: pan-y; -webkit-user-drag: none; -webkit-tap-highlight-color: rgba(0, 0, 0, 0); left: 0px;"></div>
</body>
</html>
```



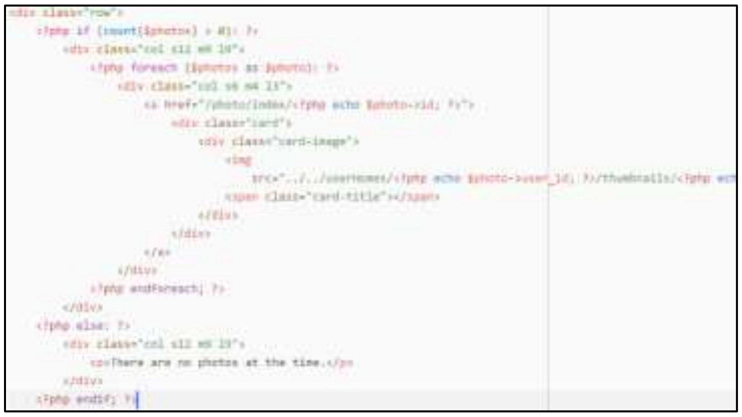
```
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="viewport"
  content="width=device-width, initial-scale=1, maximum-scale=1.0">
<title><?php echo $title ?></title>
<link href="https://fonts.googleapis.com/icon?family=Material+Icons"
  rel="stylesheet">
<link href="/views/stylesheets/materialize.min.css" type="text/css"
    rel="stylesheet" media="screen,projection">
<link href="/views/stylesheets/style.css" type="text/css" rel="stylesheet"
    media="screen,projection">
</head>
<body>
```



```
<?php
$page = 1;
        if (isset ( $_GET ['page'] ) && $_GET['page'] * 12 - 12 <= count($photos)) {
            $page = $_GET['page'];
        }
```



```
<div class="row">
  <form action="login/doLogin" method="post" autocomplete="off"
    class="col s12 m12 l8">
    <div class="row">
      <div class="input-field col s12">
        <input value="<?php echo $email; ?>" id="email" name="email"
          type="text" class="validate"> <label for="email">Email or Username</label>
      </div>
    </div>
    <div class="row">
      <div class="input-field col s12">
        <input id="password" name="password" type="password"
          class="validate"> <label for="password">Password</label>
      </div>
    </div>
    <div class="row">
      <div class="input-field col s12">
        <button class="btn waves-effect blue" type="submit" id="login"
          name="login">Login</button>
         or <a class="" href="../register"">Register
        </a>
      </div>
    </div>
  </form>
</div>
```



```
<div class="row">
    <?php if (count($photos) > 0): ?>
        <div class="col s12 m8 l9">
            <?php foreach ($photos as $photo): ?>
                <div class="col s6 m4 l3">
                    <a href="/photo/index/<?php echo $photo->id; ?>">
                        <div class="card">
                            <div class="card-image">
                                <img
                                    src="../../userhomes/<?php echo $photo->user_id; ?>/thumbnails/<?php ec
                                <span class="card-title"></span>
                            </div>
                        </div>
                    </a>
                </div>
            <?php endforeach; ?>
        </div>
    <?php else: ?>
        <div class="col s12 m8 l9">
            <p>There are no photos at the time.</p>
        </div>
    <?php endif; ?>
```

```php
<?php
public function create($firstName, $lastName, $userName, $email, $password) {
    $password = password_hash ( $password, PASSWORD_BCRYPT );
    $query = "INSERT INTO $this->tableName (firstName, lastName, userName, email, password) VALUES (?, ?, ?, ?, ?)";
    $statement = ConnectionHandler::getConnection ()->prepare ( $query );
    $statement->bind_param ( 'sssss', $firstName, $lastName, $userName, $email, $password );
    if (! $statement->execute ()) {
      throw new Exception ( $statement->error );    }}
```

```php
<?php
class ConnectionHandler{
  private static $connection = null;
  public static function getConnection(){
    if (self::$connection === null) {
      $config = require('config.php');
      $host     = $config['database']['host'];
      $username = $config['database']['username'];
      $password = $config['database']['password'];
      $database = $config['database']['database'];
      self::$connection = new MySQLi($host, $username, $password, $database);
      if (self::$connection->connect_error) {
        $error = self::$connection->connect_error;
        throw new Exception("Verbindungsfehler: $error");    }
      self::$connection->set_charset('utf8'); }

    return self::$connection;
  }
}
```

```php
class MySessionHandler{
    public function isUserLoggedIn() {
        if (session_status() == PHP_SESSION_NONE) {
            session_start();      }
        if (isset ( $_SESSION ['loggedIn'] ) && $_SESSION ['loggedIn'] == true) {
            $time = time();
            $timeout_duration = 1800;
            $sessionid_update_duration = 600;
            if (isset($_SESSION['lastActivity']) && ($time - $_SESSION['lastActivity']) > $timeout_duration) {
                session_unset();
                setcookie(session_name(), "", 1);
                setcookie(session_name(), false);
                unset($_COOKIE[session_name()]);
                session_destroy();
                return false;      }
            if (isset($_SESSION['lastSessionUpdate']) &&
             ($time - $_SESSION['lastSessionUpdate']) > $sessionid_update_duration) {
                session_regenerate_id();
                $_SESSION['lastSessionUpdate'] = $time;          }
            $_SESSION['lastActivity'] = $time;
            return true;
        } else {
            return false;      }  }}
```

```php
<?php
public function readById($id)  {
        $query = "SELECT * FROM $this->tableName WHERE id=?";
        $statement = ConnectionHandler::getConnection()->prepare($query);
        $statement->bind_param('i', $id);
        $statement->execute();
        $result = $statement->get_result();
        if (!$result) {
            throw new Exception($result->error);      }
        $row = $result->fetch_object();
        $result->close();
        return $row;  }
```

```php
lic function deleteById($id)  {
    $query = "DELETE FROM $this->tableName WHERE id=?";
    $statement = ConnectionHandler::getConnection()->prepare($query);
    $statement->bind_param('i', $id);
    if (!$statement->execute()) {}  }
```

```php
<?php
class FileService {
    public function createUsereHome($userId) {
        if (!file_exists('./userHomes/'.$userId)) {
            mkdir('./userHomes/'.$userId.'/photos', 0777, true);
            mkdir('./userHomes/'.$userId.'/thumbnails', 0777, true);        }   }
    public function delete($path)     {
        if (is_dir($path) === true)         {
            $files = array_diff(scandir($path), array('.', '..'));
            foreach ($files as $file)             {
                $this->delete(realpath($path) . '/' . $file);         }
            return rmdir($path);       }
        else if (is_file($path) === true)       {
            return unlink($path);        }
        return false;    }
blic function deleteFile($path) {
        if (is_file($path) === true)        {
            return unlink($path);        }
        return false;    }}
```

```php
blic function readPasswordByEmail($email) {
    $query = "SELECT password FROM $this->tableName WHERE email=?";
    $statement = ConnectionHandler::getConnection ()->prepare ( $query );
    $statement->bind_param ( 's', $email );
    $statement->execute ();
    $result = $statement->get_result();
    $row = $result->fetch_assoc();
    $value = $row['password'];
    $result->close();
    return $value;
}
```

```php
public function updateEmailById($email, $id)  {
    $query = "UPDATE $this->tableName SET email =? WHERE id=?";
    $statement = ConnectionHandler::getConnection ()->prepare($query);
    $statement->bind_param ( 'si', $email, $id );
    if (! $statement->execute ()) {
      throw new Exception ( $statement->error );  } }
```

```php
$userModel = new UserModel();
$id = $userModel->readIdByUsername($_SESSION['userName']);
$uniqueId = uniqid();
$newFilePath = "./userHomes/" . $id . "/photos/" . $uniqueId . "_" . basename($_FILES ['files'] ['name'] [$i]);
$thumbnailPath = "./userHomes/" . $id . "/thumbnails/" . $uniqueId . "_" . basename($_FILES ['files'] ['name'] [$i]);
$splittedName = explode(".", basename($_FILES['files']['name'][$i]));
$extension = end($splittedName);
if (in_array($extension, $validExtentions, true)) {
    if (move_uploaded_file($tmpFilePath, $newFilePath)) {
        array_pop($splittedName);
        $name = $uniqueId."_".implode(".", $splittedName);
        $size = $_FILES['files']['size'][$i];
        $height = getimagesize($newFilePath)[1];
        $width = getimagesize($newFilePath)[0];
      date_default_timezone_set('Europe/Zurich');
        $date = date("Y-m-d H:i:s");
        $photoModel = new PhotoModel();
        $photoModel->create($name,$extension,$height,$width,$size,$date,$id);
        $thumbnailCreator = new ThumbnailCreator();
        $thumbnailCreator->createThumbnail($newFilePath, $thumbnailPath);
```

Reflected XSS: .z.B. über Suchanfrage

<p>Sie suchten: <script>alert('XSS');</script></p>

Persistent XSS: Serverseitig abgespeichert, z.B. in blogbeitrag.

t in einem Blogbeitrag. <script>alert('XSS');</script>

htmlspecialchars(), htmlentities() oder strip_tags().

```php
<?php foreach($tags as $tag): ?>
    <div class="chip">
        <?php echo $tag->name; ?>
    </div>
<?php endforeach;?>
```

```php
public function doLogin() {
    if (isset ( $_POST ['login'] )) {
            if (session_status() == PHP_SESSION_NONE) {
                session_start();     }
        $formValues = $this->getFormValues ();
         $userModel = new UserModel ();
        if ($userModel->readIsEmailUsed ( $formValues ['email'] ) || $userModel->readIsUsernameUsed( $formValues ['email'] )) {
          if (password_verify ( $formValues ['password'], $userModel->readPasswordByEmail ( $formValues ['email'] ) )
          || password_verify ( $formValues ['password'], $userModel->readPasswordByUsername( $formValues ['email'] ) )) {
            session_start ();
            session_regenerate_id();
                   $id = (strpos($formValues['email'], '@') !== false ? $userModel->readIdByEmail($formValues['email']): $userModel->readIdByUsername($formValues['email']));
            $_SESSION ['userName'] = $userModel->readUserNameById($id);
                   $_SESSION['lastActivity'] = time();
                   $_SESSION['lastSessionUpdate'] = time();
            $_SESSION ['loggedIn'] = true;
            $this->clearFormValues ();
            header ( 'Location: /photos' );
          } else {
            header ( 'Location: /login' );      }
        } else {
          header ( 'Location: /login' );       }
    } else {
      header ( 'Location: /login' );    }   }
 private function getFormValues() {
 $values = array (
    'email' => (isset ( $_POST ['email'] ) ? $_POST ['email'] : ""),
    'password' => (isset ( $_POST ['password'] ) ? $_POST ['password'] : "")
 );
 return $values;
}
```

```php
require_once('libraries/Dispatcher.php');
require_once('libraries/View.php');
require_once('libraries/Model.php');
$dispatcher = new Dispatcher();
$dispatcher->dispatch();
```

```php
<?php
public function isValid($regex, $value) {
    if (preg_match ( $regex, $value )) {
        return true;
    } else {
        return false;
    }
}
```