Today’s Agenda

• Repetition: Powerpoint Karaoke
• Introduction to Databases and (My)SQL
• Break Out: Music-Albums Organization Table
• Quiz
• Discussion of “Hangman” Solution (Assignment 03)
Powerpoint Karaoke:
PHP Sessions
Sessions

• Sessions maintain “states” on the server side
• Sessions store current state of variables as long as connected to the client
• On the client side, sessions are identified with a session ID cookie:
  – default cookie name in PHP: PHPSESSID
  – renaming possible with session_name()
Sessions with PHP

- Sessions need to be started **before any output occurs**
- Create session ID cookie:
  ```php
  session_start()
  ```
- Delete the session ID cookie:
  ```php
  session_destroy()
  ```
- Read / write session values:
  - superglobal `$_SESSION` array
  - immediately reset session like this `$_SESSION = array();`
Interaction with Databases
Databases and SQL

• Data can be stored **permanently** in databases

• There are a number of database management systems (DBMS). In this lecture & tutorial we use **MySQL**

• SQL (= Structured Query Language) is a language that allows us to access databases. We can retrieve and manipulate data with it.

• With SQL you can:
  – Create databases
  – Create tables
  – Retrieve data from a database
  – Store data in a database
  – ...
Tables in relational databases

- A relational database usually consists of one or more **tables**
- Each table has a unique name with one or more **columns**
- Each table can have multiple entries (or none).
- A table **row** represents an entry

<table>
<thead>
<tr>
<th>PersonID</th>
<th>FirstName</th>
<th>LastName</th>
<th>PhoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Max</td>
<td>Mustermann</td>
<td>089455544431</td>
</tr>
<tr>
<td>2</td>
<td>Laura</td>
<td>Stern</td>
<td>070815643593</td>
</tr>
<tr>
<td>3</td>
<td>Tanja</td>
<td>Baumann</td>
<td>0895673138</td>
</tr>
<tr>
<td>4</td>
<td>Felix</td>
<td>Maurer</td>
<td>0894562897</td>
</tr>
</tbody>
</table>

*Table: Contacts*
MySQL at the CIP-Pool

• Access “Datenbank Management” here: https://tools.rz.ifi.lmu.de/
• Create a new account (required)
• Create a new database (required)
• Connect to db2.cip.ifi.lmu.de
MySQL at the CIP-Pool (II)

To work with the database, you have to connect to the database server:

1. Start a SHELL (Ctrl+Alt+T)
2. Enter the following command:
   ```
   mysql -h db2.cip.ifi.lmu.de -u [username] -p
   ```
3. Provide your password
4. If successfull you should see something like this:
MySQL with your local database(I)

- **XAMPP** lets you work with your own, local MySQL database
- Make sure you start the MySQL Service in the control center
MySQL with your local database (II)

- Connect to a local database server:
  1. Change to the „.../xampp/mysql/bin“ directory
  2. Enter the following command:
     ```
     mysql -h localhost -u [username] -p
     ```
  3. Enter the password (usually “root”, “admin”, “password” or none)
  4. You should see something like the following:

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.5.34 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```
MySQL with your local database (III)

• You can perform work with MySQL through a very common web interface: phpMyAdmin

• Once you’ve started the Apache & MySQL Servers in XAMPP, enter the following URL in a web browser:
  – http://localhost/phpmyadmin
SQL: Creating a database

- Get an overview on all existing databases:
  `SHOW DATABASES;`

- Create a new database:
  `CREATE DATABASE mydb;`

- Select a database for further usage:
  `USE mydb;`

- Delete a database (be careful!):
  `DROP DATABASE mydb;`
SQL: Creating a table (I)

- Get an overview on all existing tables (of a database):
  
  ```sql
  SHOW TABLES;
  ```

- Create a new table
  
  ```sql
  CREATE TABLE myTable
  (column_name1 data_type(size),
  column_name2 data_type(size),
  column_name3 data_type(size),
  ...
  );
  ```

Table: myTable

<table>
<thead>
<tr>
<th>column_name1</th>
<th>column_name2</th>
<th>column_name3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SQL: Creating a table (II)

• Problems with the statement from previous slide:
  – You can add empty entries to the table
  – Entries could be duplicates

• Solution: Create a table with certain constraints. Define certain rules for columns

• Most important constraints (among many others):
  – NOT NULL
  – PRIMARY KEY [often in conjunction with] AUTO_INCREMENT
SQL: Creating a table (III)

• Create a table with certain constraints

```
CREATE TABLE myTable
(
  column_name1 data_type(size) NOT NULL,
  column_name2 data_type(size) NOT NULL,
  column_name3 data_type(size),
  ...
);
```
Example: Creating a table

```
CREATE TABLE Contacts
(
    PersonID int NOT NULL PRIMARY KEY AUTO_INCREMENT,
    FirstName varchar(255) NOT NULL,
    LastName varchar(255) NOT NULL,
    PhoneNumber int NOT NULL,
);
```

Table: Contacts

<table>
<thead>
<tr>
<th>PersonID</th>
<th>FirstName</th>
<th>LastName</th>
<th>PhoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SQL: Adding & Retrieving data

- **Add entries:**
  
  ```sql
  INSERT INTO myTable
  (column_name1, column_name2, ...)
  VALUES
  (value1, value2, ...);
  ```

- **Retrieve all entries from a table:**
  
  ```sql
  SELECT * FROM myTable;
  ```

- **Retrieve only a subset of entries**
  - Entries that fulfill certain conditions with the **WHERE** keyword
    
    ```sql
    SELECT * FROM myTable WHERE column_name=value;
    ```
  - Entries from specific columns:
    
    ```sql
    SELECT column_name1 FROM myTable;
    SELECT column_name1, column_name2 FROM myTable;
    ```
Example: Add an entry

```
INSERT INTO Contacts
    (FirstName, LastName, PhoneNumber)
VALUES
    ("Max", "Mustermann", 089455544431);
```

Table: Contacts

<table>
<thead>
<tr>
<th>PersonID</th>
<th>FirstName</th>
<th>LastName</th>
<th>PhoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Max</td>
<td>Mustermann</td>
<td>089455544431</td>
</tr>
</tbody>
</table>
Example: Retrieve data

- Retrieve all data from a table
  
  ```sql
  SELECT * FROM Contacts
  ```

- Retrieve entries that fulfill a certain condition:
  
  ```sql
  SELECT * FROM Contacts WHERE FirstName="Laura";
  ```

<table>
<thead>
<tr>
<th>PersonID</th>
<th>FirstName</th>
<th>LastName</th>
<th>PhoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Max</td>
<td>Mustermann</td>
<td>089455544431</td>
</tr>
<tr>
<td>2</td>
<td>Laura</td>
<td>Stern</td>
<td>070815643593</td>
</tr>
<tr>
<td>3</td>
<td>Tanja</td>
<td>Baumann</td>
<td>0895673138</td>
</tr>
<tr>
<td>4</td>
<td>Felix</td>
<td>Maurer</td>
<td>0894562897</td>
</tr>
</tbody>
</table>

*Table: Contacts*
Break Out

• Use SQL to create a table to store information about music albums

• Each album has:
  – An artist
  – A title
  – A track count
  – A runtime
  – A price
  – A link to a cover image
    (e.g. https://upload.wikimedia.org/wikipedia/en/0/0c/Velvet_Underground_and_Nico.jpg)
  – A Universal Product Code (UPC)

• If you have time, insert some data!
Round-up Quiz

1. True or False: Databases store information permanently.
2. Describe the result:
   \[
   \text{SELECT firstName, lastName FROM contacts;}
   \]
3. Spot the error:
   \[
   \text{INSERT INTO contacts VALUES (John, Smith, 5555320039);}
   \]
4. What is a “relational” Database?
Thanks!

What are your questions?
Discussion of Assignment 03

Hangman

Word: _ _ _ _ _ _ E _ _ A

Guess
Let’s begin with the Assignment!

- Download the assignment sheet
- Start with task 1
- You can collaborate with your neighbor
- Turn in the assignment by November 18th, 12:00 noon via UniWorX