Westfälische Wilhelms-Universität Münster

Erstellen dynamischer Webseiten mit PHP

WS 2016/2017

wissen.leben
WWU Münster

Dozenten: Patrick Förster, Michael Hasseler

Übersicht der heutigen Inhalte

10. Vorlesung: Einstieg zu MySQL-Datenbanken

- SQL-Befehle
  - Datentypen
  - CREATE
  - INSERT
  - SELECT
  - UPDATE
  - DELETE
- phpMyAdmin als grafische Oberfläche
- PDO

Auszug an Datentypen

TINYINT(1) bzw. BOOL	Boolean mit den Wert 0 oder 1
INT	Ganze Zahl 4 Byte
DOUBLE	Gleitkommazahl 8 Byte
CHAR(m)	Zeichenfolge fester Länge (0-255) Auffüllen mit Blanks, falls Inhalt kürzer ist
VARCHAR(m)	Zeichenfolge variabler Länge (0-65536)
DATETIME	'**1000-01-01 00:00:00**' bis '**9999-12-31 23:59:59**', Darstellung '**YYYY-MM-DD HH:MM:SS**'
TIMESTAMP	Anzahl Sekunden seit '**1970-01-01 00:00:00**',

Vgl. http://dev.mysql.com/doc/refman/5.1/de/data-type-overview.html

Beispiel: Struktur der Tabelle benutzer

Feldname	Datentyp
login	varchar(20)
passwort	varchar(20)
vorname	varchar(30)
nachname	varchar(30)
angemeldet_am	datetime

Beispiel 1: CREATE-Statement

- **CREATE TABLE** benutzer (
  login varchar(20) NOT NULL default '',
  passwort varchar(20) NOT NULL default '',
  vorname varchar(30) NOT NULL default '',
  nachname varchar(30) NOT NULL default '',
  angemeldet_am timestamp NOT NULL default CURRENT_TIMESTAMP,
  PRIMARY KEY (login)
);

Beispiel 2: CREATE-Statement

- **CREATE TABLE** spielbegegnungen (
  jahr int,
  spieltag int,
  heimmannschaft varchar(40),
  auswaertsmannschaft varchar(40),
  heim_tore int,
  auswaerts_tore int);

Beispiel: Inhalt der Tabelle benutzer

login	passwort	vorname	nachname	angemeldet_am
muster	muster123	Max	Mustermann	2016-11-30 14:15:00
fred	fred123	Fred	Feuerstein	2016-12-03 08:45:12
wilma	wilma123	Wilma	Feuerstein	2016-12-09 19:55:30

Datensatz hinzufügen: INSERT

- Syntax:

INSERT INTO <tabellenname> (<feldname1>, <feldname2>, …)
VALUES (<feldname1>, <feldname2>, …);

- Fehlen Felder beim **INSERT INTO**-Statement gegenüber der Tabellendefinition, dann werden Defaultwerte oder null in der Datenbank eingetragen

Beispiel 1: Inhalt der Tabelle benutzer

- **INSERT INTO** benutzer ('login', 'passwort', 'vorname', 'nachname') **VALUES** ('paulchen', 'rosa123', 'Paulchen', 'Panther');

login	passwort	vorname	nachname	angemeldet_am
muster	muster123	Max	Mustermann	2016-11-30 14:15:00
fred	fred123	Fred	Feuerstein	2016-12-03 08:45:12
wilma	wilma123	Wilma	Feuerstein	2016-12-09 19:55:30
paulchen	rosa123	Paulchen	Panther	2016-12-19 16:40:00

Datensatz auswählen: SELECT 1/2

- Syntax:

SELECT [DISTINCT] [col_name1, col_name2, …]
[FROM <tablenname> [AS alias_name]]
[WHERE <condition>] // Vor der Gruppierung
[**GROUP BY** col_name1[, col_name2, …]]
[**HAVING** <condition>] // Nach der Gruppierung
[**ORDER BY** col_name1 [DESC], [col_name2 [DESC], …]
[**LIMIT** row_count];

Vgl. http://dev.mysql.com/doc/refman/5.5/en/select.html

Datensatz auswählen: SELECT 2/2

- Mit **SELECT** kann man Datenbankspalten selektieren.
  - (* = alle Spalten ausgewählt)
  - **DISTINCT**: bei den Ergebniszeilen werden doppelte Zeilen entfernt
- Mit **FROM** gibt man an, auf welche Tabelle zugegriffen wird.
- Mit **WHERE** wählt man Datensätze aus, die mit Bedingungen eingeschränkt werden können.
- Mit **GROUP BY** werden Zeilen zusammengefasst, die identische Felder haben.
- Mit **HAVING** kann man das Gruppierungsergebnis mit Bedingungen einschränken.
- Mit **ORDER BY** wird die Sortierspalte und Richtung festgelegt.
- Mit **LIMIT** kann die Ergebnismenge beschränkt werden.

Vgl. http://dev.mysql.com/doc/refman/5.5/en/select.html

Auszug an Bedingungsoperatoren in SQL

- übliche Arithmetik mit Klammern, **+, -, *, /**
- Logisch: **NOT, !, OR, ||, AND, &&**
- Vergleich: **=, !=, <=, <, >=, >, IS NULL, IS NOT NULL, expr IN (...), expr NOT IN (...)**
- Zeichenvergleich**: LIKE, NOT LIKE**

Beispiele: SELECT

- Beispiel 1:
- **SELECT** * **FROM** benutzer **WHERE** 'login' = 'muster';

Ergebnis:

login	passwort	vorname	nachname	angemeldet_am
muster	muster123	Max	Mustermann	2016-11-30 14:15:00

- Beispiel 2:
- **SELECT** vorname **FROM** benutzer **WHERE** 'login' = 'muster';
- Ergebnis:

vorname
Max

Datensatz aktualisieren: UPDATE

- Syntax:

UPDATE <tablenname>
SET <col_name1> = expr1 [, <col_name2> = expr2 , …]
[**WHERE** condition]
[**ORDER BY** …]
[**LIMIT** row_count];

Vgl. http://dev.mysql.com/doc/refman/5.5/en/update.html

Beispiel eines UPDATE-Statements

```
UPDATE benutzer
SET angemeldet_am = now()
WHERE login='fred';
```

Ergebnis:

login	passwort	vorname	nachname	angemeldet_am
muster	muster123	Max	Mustermann	2016-11-30 14:15:00
fred	fred123	Fred	Feuerstein	2016-12-19 16:45:10
wilma	wilma123	Wilma	Feuerstein	2016-12-16 19:55:30

Datensatz löschen: DELETE

- Syntax:

DELETE FROM <tablename>
[**WHERE** condition]
[**ORDER BY** …]
[**LIMIT** row_count];

Vgl. http://dev.mysql.com/doc/refman/5.5/en/delete.html

Beispiel eines DELETE-Statements

DELETE FROM benutzer
WHERE login=‘muster‘;

Ergebnis:

login	passwort	vorname	nachname	angemeldet_am
fred	fred123	Fred	Feuerstein	2016-12-03 08:45:12
wilma	wilma123	Wilma	Feuerstein	2016-12-09 19:55:30

PHP Data Objects-Erweiterung (PDO)

- Leichte und konsistente Schnittstelle
- Abstraktionsschicht für den Datenbankzugriff, sodass einheitlich auf Datenbanken zugegriffen werden kann
- PDO ab Version 5.1 in PHP mit ausgeliefert
- PDO verwendet neue OO-Features von PHP 5, daher nicht mit älteren Version nutzbar
- Datenbanktreiber muss die PDO-Schnittstelle implementieren
- Bei Verwendung „reiner“ SQL-Standardbefehle und PDO kann die Datenbank bei vorhandenen Treibern getauscht werden

Quelle: vgl. http://php.net/manual/de/intro.pdo.php

Verbindungsaufbau mit PDO

- Mit dem **new**-Operator erzeugt man von der Klasse PDO ein Objekt, das die Verbindung zur Datenbank aufbaut.
- Dem Klassenkonstruktor werden die Verbindungsdaten als Parameter übergeben.
- Der Verbindungsaufbau sollte in einem **try-catch**-Block stehen, damit nicht die kompletten Datenbankverbindungsdaten ausgegeben werden
  - Wenn es keine Probleme im try-Block gibt, dann erhält man die Datenbankverbindung.
  - Im Fehlerfall wird eine **PDOException** geworfen, die vom catch-Block abgefangen wird. Mit print wird nur die Fehlermeldung ausgegeben und dann mit die() das Skript beendet.

Quelle: vgl. http://www.php.net/manual/en/pdo.connections.php

Beispiel: PDO-Verbindung zu einer MySQL-Datenbank

- 1. Parameter: Angabe zur Datenbankquelle
- 2. Parameter: Name des Datenbanknutzers
- 3. Parameter: Passwort des Datenbanknutzers

```
<?php
  ..
  try {
    $db = new PDO('mysql:host=localhost;dbname=test;charset=utf8',
$user, $password);
  }
  catch (PDOException $e) {
    echo "Error!: " . $e->getMessage();
    die();
  }
  ..
?>
```

Das PDOStatement-Objekt

1. SQL-Befehl vorbereiten:
   - Aufruf der Methode `prepare` beim Datenbank-Objekt liefert ein `PDOStatement`-Objekt
   - Der SQL-Befehl wird dem `prepare` als Parameter übergeben
2. SQL-Befehl ausführen:
   - Mit Aufruf der Methode `execute` auf dem `PDOStatement`-Objekt wird der vorbereitete SQL-Befehl auf der Datenbank ausgeführt
   - Rückgabe `TRUE` bei Erfolg und `FALSE` bei Fehler
   - Bei TRUE enthält das `PDOStatement`-Objekt die Ergebnismenge
3. Ergebnismenge zugreifen:
   - Aufruf von `fetch` beim `PDOStatement`-Objekt liefert die erste Datenzeile, der zweite Aufruf von `fetch` ggf. die zweite Datenzeile usw.

Quelle: vgl. http://php.net/manual/en/class.pdostatement.php

Auszug einiger fetchMode-Konstanten

- Mit Fetchmode kann der wird mit Klassenkonstanten von `PDO` gesetzt

1. `PDO::FETCH_ASSOC`: liefert ein assoziatives Array mit den Spaltennamen als Schlüssel
2. `PDO::FETCH_NUM`: liefert ein indiziertes Array mit den Spaltennummern als Schlüssel
3. `PDO::FETCH_BOTH` (default): `FETCH_ASSOC` und `FETCH_NUM` zusammen
4. `PDO::FETCH_CLASS`: liefert eine neue Instanz der übergebenen Klasse, wobei die Objektattribute mit den Spaltennamen überstimmen müssen
5. `PDO::FETCH_PROPS_LATE`: sorgt im Fall von `FETCH_CLASS` dafür, dass erst der Konstruktor aufgerufen wird und dann die Objektattribute gemappt werden

Quelle: vgl. http://www.php.net/manual/en/pdostatement.fetch.php

Beispiel: SELECT-Abfrage mit PDO

- Annahme: $db enthält die Verbindung zur Datenbank

```
$sql = 'SELECT * FROM spielbegegnungen';
$statement = $db->prepare($sql);
$statement->execute();
$statement->setFetchMode(PDO::FETCH_ASSOC);
$spiel = $statement->fetch();
while (is_array($spiel)) {
  foreach ($spiel as $key => $value) {
    echo $key.': '.$value.PHP_EOL;
  }
  $spiel = $statement->fetch();
}
```

Bedingungen mit bindParam in PDO angeben

- In Bedingung Parameter mit „:Variablenname“ einfügen
- `Prepare-Statement` mit **`bindParam`** den
- Bei Verwendung von Prepare-Statements werden Parameter vom Treiber automatisch maskiert und mögliche SQL-Injections verhindert
- Beispiel:

```
...
$sql = 'SELECT * FROM spielbegegnungen WHERE spieltag=:tag';
$statement = $db->prepare($sql);
$statement->bindParam(':tag', $spieltag)
$statement->execute();
...
```

Quelle: vgl. http://www.php.net/pdo.prepared-statements

Beispiel: SELECT-Abfrage mit PDO

- Annahme: $db enthält die Verbindung zur Datenbank
- `query` führt den SQL-Befehl im Gegensatz zu `prepare` direkt aus

```
$sql = 'SELECT jahr AS spieljahr, spieltag, heimmannschaft,
auswaertsmannschaft, heim_tore AS heimtore, auswaerts_tore AS
auswaertstore FROM spielbegegnungen';
$statement = $db->query($sql);
$statement->setFetchMode(PDO::FETCH_CLASS|PDO::FETCH_PROPS_LATE,
'Spielbegegnung');
while ($spiele = $statement->fetch()) {
  echo $spiele.PHP_EOL;
}
```

phpMyAdmin mit grafischer Oberfläche

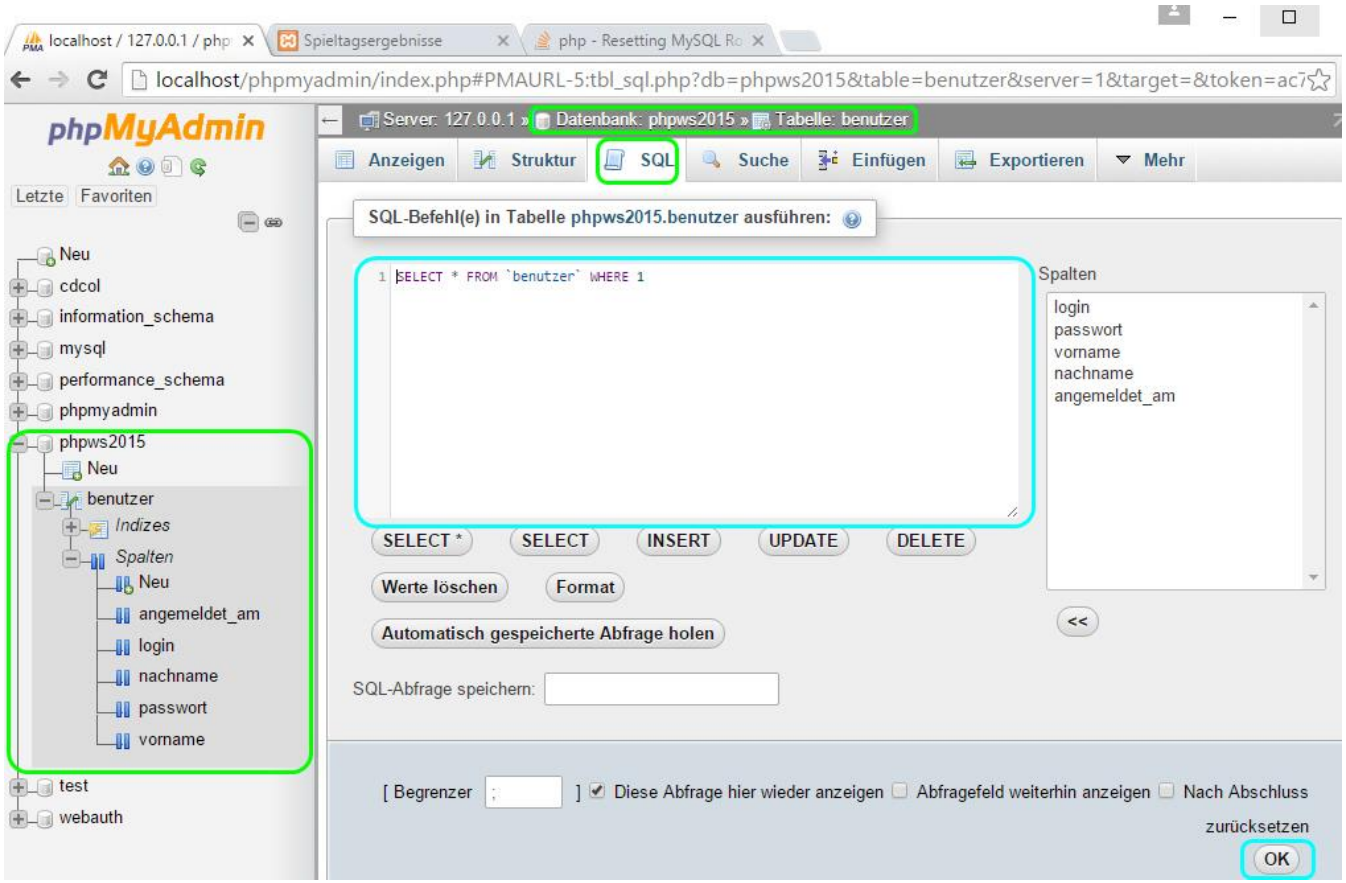
- Aufruf mit dem Browser
- Ist bei Providern die Standardschnittstelle zum Verwalten einer MySQL-Datenbank.
- In die Adressleite bei der virtuellen Maschine eingeben:

  localhost/phpmyadmin/index.php

Anhang: phpMyAdmin verwenden

- Im Browser **localhost/phpmyadmin/index.php** aufrufen
  - Nutzer: **root**
  - Passwort: **initial keins vergeben**
- Befehle 1 bis 4 (siehe Anhang) in SQL-Bereich von phpMyAdmin ausführen (nächste Folie)
  - Datenbank **php2016ws** anlegen
  - Datenbank-Nutzer **user_phpws2016** anlegen
  - Datenbanktabelle **benutzer** anlegen
  - Fügen Sie in der Tabelle Datensätze hinzu(siehe Anhang)
- Links in phpMyAdmin sollte nun die Datenbank **php2016ws** mit der Tabelle benutzer aufgelistet werden (siehe nächste Folie)

Ansicht von phpMyAdmin im Browser



Zusammenfassung

- Verwendung der SQL-Befehle CREATE, SELECT, INSERT, UPDATE und DELETE
- Aufbauen einer PDO-Connection mit try-catch-Block
- Ausführen eines Prepared-Statements mit bindParams in PDO

Aufgaben

1. Das Beispiel mit den Fußballergebnissen soll umgeschrieben werden, sodass die Speicherung der Spielbegegnungen nun auch in der MySQL-Datenbank erfolgt (hierfür die SQL-Befehle 1-6 im Anhang vorher ausführen):
   a) Schreiben Sie die Funktion `addToDB($spielbegegnung)`, die als Parameter ein Objekt der Klasse Spielbegegnung übergeben bekommt. Diese Funktion soll die bisherige Funktion `addToFile()` ersetzen.
   b) Ersetzen Sie im php-Skript ergebnisse_auflisten.php den Aufruf `fopen` durch einen Datenbankzugriff mittels PDO, welches die Spielbegegnungen zurückliefert.
2. Erweitern Sie die Anzeige der Fussballergebnisse um eine Combobox für das Spieljahr, die zur Filterung verwendet wird. Die Box soll mit den aktuell vorhandenen Spieljahren aus der Datenbank gefüllt werden. Des Weiteren soll es den Eintrag ‚Alle' in der Combobox geben, der initial ausgewählt ist oder falls der Post-Wert zur combobox-Auswahl leer ist. Bei der Auswahl ‚Alle' werden die Spielbegegnungen zu allen Jahren angezeigt.

Anhang: SQL-Befehle (sql_befehle.txt)

- 1) CREATE DATABASE phpws2016;
- 2) GRANT SELECT, INSERT, UPDATE, DELETE ON phpws2016.*
  TO 'user_phpws2016'@'localhost'
  IDENTIFIED BY 'phpFNeanW16';
- 3) CREATE TABLE benutzer(login varchar(20) NOT NULL default '', passwort varchar(20) NOT NULL default '', vorname varchar(30) NOT NULL default '', nachname varchar(30) NOT NULL default '', angemeldet_am timestamp NOT NULL default CURRENT_TIMESTAMP, PRIMARY KEY (login));

Anhang: SQL-Befehle (sql_befehle.txt)

- 4) INSERT INTO benutzer(login, vorname, nachname, passwort) VALUES
  ('muster', 'Max', 'Mustermann', 'muster123'),
  ('fred', 'Fred', 'Feuerstein', 'fred123'),
  ('wilma', 'Wilma', 'Feuerstein', 'wilma123'),
  ('paulchen', 'Paulchen', 'Panther', 'rosa123');
- 5) CREATE TABLE spielbegegnungen(jahr int, spieltag int, heimmannschaft varchar(40), auswaertsmannschaft varchar(40), heim_tore int, auswaerts_tore int);
- 6) INSERT INTO spielbegegnungen(jahr, spieltag, heimmannschaft, auswaertsmannschaft, heim_tore, auswaerts_tore) VALUES
(2013, 1, 'Hamburg', 'Bremen', 1, 2);