FH University of Applied Sciences
TECHNIKUM WIEN

Backend Basics

Wiederholung und Datenbanken

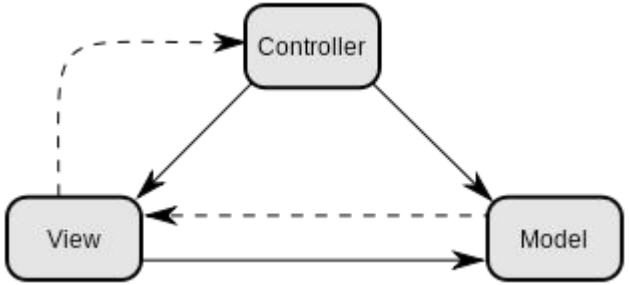
FH University of Applied Sciences
TECHNIKUM WIEN

Agenda

- Diskussion zur Übung 2
- Wiederholung der Konzepte
  - MVC
  - Interface
  - Clean Code
- Einsatz von Datenbanken mit PhP
- Übung zu Datenbanken

MVC: Model View Controller mit PhP

- **Controller** steuert das Zusammenspiel von Model und View
  kümmert sich um Input und Output der
  Web Anwendung
  - hier ist die Businesslogik implementiert



- **Model** datenhaltende und daten-
  verarbeitende Klassen
  - enthält im Falle von PhP viel Logik
  - bindet Services ein (DB, SESSION)

- **View** Repräsentation der Ausgabe je nach Bedarf

Interface

- spezifiziert für Klassen, welche Methoden implementiert werden müssen
- es wird Austauschbarkeit erzeugt
- macht den Softwareentwurf durch Kapselung flexibel

```
// The signatures of the methods for the implementing class are defined
interface View {
    public function setHeaders();
    public function streamOutput($data);
}

// The real class has to implement the methods from the implemented interface
class RealView implements View{
    public function setHeaders(){
      // ... concrete implementation ...
    }
    // ... same for streamOutput($data) ...
}
```

Wiederholung Clean Code

- Achtsamkeit bei der Namenswahl
- Code sollte ohne Kommentare lesbar sein
- eine Methode sollte nur auf einer logischen Ebene agieren
- tiefere Logik-Ebenen erfordern Unterfunktionen (gewöhnlich private)
- Indizien für Dirty Code
  - Methoden mit mehr als 20 Zeilen
  - Methoden die mehr als 2 Parameter erwarten
  - Verschachtelungen tiefer als 2 Ebenen zb: `for( … if( … if(...`
  - Klassen mit mehr als 300 Zeilen deuten auf s.g. Gott-Klassen hin
  - Abkürzungen für Namen z.b.: img für image

FH University of Applied Sciences TECHNIKUM WIEN

Fragen bis hier?

Einsatz von Datenbanken

Warum

- Um Daten zentral zu halten und zu verwalten
- Zugriff durch mehrere Anwendungen möglich

Wie

- Datenbankserver
- Relationale Datenbank
- SQL - Structured Query Language

Datenbanken und PhP

Was

- Schnittstelle zur DB über PDO Objekt
- Datenbank Treiber wählen
  - Datenbank Typ: MySQL, postgres, Oracle, DB2, MSSQL, etc.
- Verbindung aufbauen
  - Host-Adresse des DB Servers, DB-Name, DB-Benutzer
- SQL Statements ausführen
  - zb.: SELECT id, firstname, lastname FROM persons WHERE created >= ‘2018-01-01’

bzgl. MVC: Datenbankobjekte sind Services und werden **von Models genutzt**

Installation: evtl. die Extension in php.ini einkommentieren (extension=php_pdo.dll)

PDO in PhP

```
//PDO Objekt erzeugen
$pdoObject = new PDO("mysql:host=localhost;dbname=$dbName;charset=utf8", "root", "");
//den SQL Query vorbereiten
$sql = “SELECT id, street, zip, location FROM addresses WHERE zip = ‘1010’ ”;

$resultTable = array(); //wir bereiten uns ein leeres Array vor.
try{
    //iterieren für alle moeglichen zeilen
    foreach ($pdoObject->query($sql) as $row) {
        $resultTable[] = $row; //die Zeile in den Array schreiben
    }
} catch (PDOException $ex){
    //error handling, wenn der query schief ging
    error_log("PDO ERROR: querying database: " . $ex->getMessage()."\n".$sql);
}
```

Übung zu Datenbanken

Legen Sie eine Datenbank an, verbinden Sie sich damit und geben Sie die Ergebnisse aus.

- Die Datenbank soll eine Tabelle enthalten: persons(id, firstname, lastname, birthdate, pk: id)
- erstellen Sie selbstständig Testdaten
- Wiederholen Sie in den Unterlagen, wie die Anbindung mittels pdo funktioniert und führen Sie die Schritte selbstständig durch

Stellen Sie im Forum Fragen, wenn etwas unklar ist.

Test in kommender LV

Stoffeingrenzung

- Theoretisches Wissen
  - zur Funktionsweise von Webanwendungen (Slides Einheit 1)
  - zu PhP (Datentypen, Schleifen, auch Einheit 1)
  - zu den besprochenen Konzepten (Einheit 2 und 3)

Es wird ein **Multiple Choice** Test sein: Mehrere Antworten je Frage möglich.

Es gibt 20 Punkte zu erreichen die zusätzlich zu den 100 Gesamtpunkten erreicht werden können.