Backend Basics

Implementieren von Beispielen

FH University of Applied Sciences TECHNIKUM WIEN

Agenda

- Diskussion zur Übung 1
- gemeinsames Implementieren mit PhPStorm des Bsp. aus Übung 1
- Besprechen der verwendeten Konzepte
  - Includes
  - MVC
  - Interface
  - Clean Code
- Besprechen der Hausübung
- Planen der zweiten Anwendung

Übung 1

Es soll eine Telefonzentrale als Service implementiert werden, das Anrufe für mehrere Abteilungen entgegen nimmt, verteilt und der Gegenstelle die Leitung, sowie den Namen der Abteilung und eines zuständigen Mitarbeiters zurück gibt.

Folgende **Schnittstelle** soll das Service bereitstellen:

```
GET Parameter String: “departmentPhoneNumber”
```

Der **Output** soll wie in folgender Form sein:

```
JSON Object:{department: <string>, employee: <string>, line: <int>}
```

Details zur Spezifikation

Die Organisation ist wie folgt aufgebaut:

Department A
hat die Rufnummer: 0800 123123
Folgende Mitarbeiter Arbeiten hier und und werden inkl. Durchwahl aufgelistet
- Hr. Huber - DW 1
- Fr. Moser - DW 2
- Fr. Schmidt - DW 3

Department B
hat die Rufnummer: 0800 456456
Folgende Mitarbeiter Arbeiten hier und und werden inkl. Durchwahl aufgelistet
- Fr. Winkler - DW 1
- Hr. Leitner - DW 2

Der Ablauf wäre z.b. wie folgt:
Der Anrufer ruft zb. Department A an:

INPUT http://localhost/index.php?departmentPhoneNumber=0800123123

Planen einer Anwendung

- überlegen Sie welche Objekte wahrscheinlich eine Rolle spielen werden
- welche Parameter müssen weiter gereicht werden?
- Wie sind die Daten organisiert? (DB, CSV, ??)
- wie könnten die Namen der Objekte lauten?
- welche Eigenschaften und Methoden werden die Objekte brauchen?
- wo wird es notwendig sein Schleifen oder Fallunterscheidungen einzuplanen?
- Welche Technologien kommen zum Einsatz?
- zeichnen Sie einen leserlichen Plan auf
- Prüfen Sie mit der Angabe ob, alles berücksichtigt wurde

FH University of Applied Sciences TECHNIKUM WIEN

Live Programming

Praktische Tipps zum Arbeiten

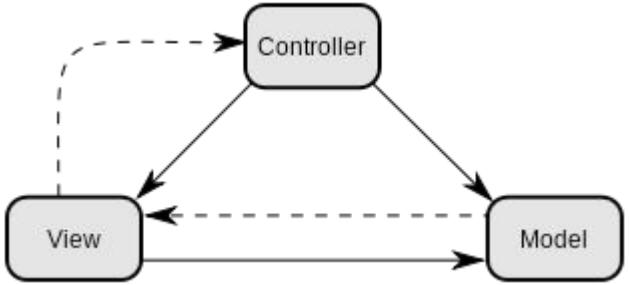
- regelmäßiges Speichern
- immer lauffähigen Code hinterlassen
- möglichst git zur Versionskontrolle verwenden
- Debugging erlernen
- IDE nutzen und sinnvoll konfigurieren (so dass Sie sich wohl fühlen) aber Achtung bei coding standards

Includes

- Zusammenführen von Code
- Aufteilung für bessere Lesbarkeit
- Strukturierung des Sourcecodes

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 vereinfachten Falle auch 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) ...
}
```

Clean Code

- 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

PSR2

- bekanntester Coding Standard in PhP
- vereinbart wie man Code formatiert
- vereinfacht das Zusammenarbeiten und diverse Codeanalysen
- trägt zur Steigerung der Qualität bei
- https://www.php-fig.org/psr/psr-2/

FH University of Applied Sciences TECHNIKUM WIEN

Fragen?

Hausübung

Planen und implementieren Sie eines Service das PhP Schleifen simuliert. Jede der drei Schleifen Varianten soll in einem Objekt gekapselt sein. Das Service kann über einen Parameter gesteuert werden und drei verschiedene Simulationen ausführen. Das Ergebnis wird als JSON zurück geliefert.

Als Input dient ein Array der aus den Buchstaben $characters = [A..Z] besteht.

- Die For-Schleife soll alle geraden Buchstaben in ein Array ablegen.
- Die Foreach Schleifen Simulation soll einen rückwärts sortierten Array per For Schleife erzeugen; also [Z..A].
- Die While Schleife soll alle Zeichen in ein Array Schreiben bis das gewünschte Zeichen gefunden wurde.

Schnittstelle
- GET Parameter String: loopType (mögliche Werte: REVERSE, EVEN, UNTIL)
- GET Parameter String: until (bis zu welchem Zeichen)

Output:
```
JSON Object: {loopName: <string>, result: <array> }
```

Achten Sie darauf, wie die Objekte konzipiert werden, wie sie zusammenarbeiten und welche Eigenschaften von außen nicht manipuliert werden sollten. Überlegen Sie sich wo und wie die Daten zwischen Objekten weiter gegeben werden. Überprüfen Sie Ihre JSON Ausgabe mit https://jsonlint.com/ .

Abgabe der Übung 2

- bis 5.4.2020
- im Moodle unter Uebung 2
- NACHNAME_Uebung2.zip (kein *.rar)
  - mit allen Files des Projektes
  - bitte keine versteckten Files die sonst in Ihrem Filesystem vorkommen *.ts, *.ps, usw.

- Fragen bitte im Diskussionsforum (Keine Scheu! Wir sind hier um zu lernen!)
- versuchen Sie es wirklich alleine
- vergleichen Sie mit der Anwendung aus Uebung1
- versuchen Sie clean zu Implementieren

Planen der zweiten Anwendung

- überlegen Sie welche Objekte wahrscheinlich eine Rolle spielen werden
- wie könnten die Namen der Objekte lauten
- welche Eigenschaften und Methoden werden die Objekte brauchen
- wo wird es notwendig sein Schleifen oder Fallunterscheidungen einzuplanen
- welche Parameter müssen weiter gereicht werden?
- zeichnen Sie einen leserlichen Plan auf