Backend Basics

Wiederholung

FH University of Applied Sciences TECHNIKUM WIEN

Agenda

- Diskussion zur Übung 4
- Wiederholung und Vorbereitung auf die Klausur
  - Konzeption
  - Datenbankanbindung
  - JSON, GET, POST
  - RESTFul
  - Implementieren
- Übung 5

Wiederholung: Konzeption

- Wichtige Fragestellungen?

Wiederholung: Konzeption

- Wichtige Fragestellungen?
  - Welche Funktionen werden implementiert?
  - Wie sind die Funktionen ab zu grenzen!?
  - Wie wird der Ablauf im Controller sein? Welche Models werden notwendig?
  - Welche Datenbank Einträge werden benötigt? (Was ist mein Inuput, was mein Output?)
  - welche Parameter müssen weiter gereicht werden?
  - Welche Objekte wird es geben?
  - Wie könnten die Namen der Objekte lauten?
  - Welche Eigenschaften und Methoden werden die Objekte brauchen?

- MVC: Model View Controller
  - Controller: steuert
  - Model: verarbeitet
  - View: repräsentiert die Ausgabe

Wiederholung: Datenbankanbindung

- mit PhPMyAdmin arbeiten
- Abfragen auf DB
- DB Ergebnis verarbeiten
- Schreiben auf der DB

Wiederholung: Datenbankanbindung

- mit PhPMyAdmin arbeiten
  - Wo finde ich den phpmyadmin
  - DB anlegen, SQL Importieren
- Abfragen auf DB
  - PDO Objekt in PhP nutzen
- DB Ergebnis verarbeiten
  - klassisch durch foreach Schleife
- Schreiben auf der DB
  - query per DPO
  - last inserted id (AUTO_INCREEMENT)

Wiederholung: JSON

- Unterschied von GET und POST
- JSON Input verarbeiten
- JSON Output an Schnittstellen Design anpassen

Wiederholung: JSON, GET, POST

- Unterschied von GET und POST
  - bei GET sind Parameter in der URL (limitiert durch Client)
  - bei POST sind Parameter im HTTP Paket (limitiert durch Server)
  - Validieren von Input durch Browser ist immer notwendig
- JSON Input verarbeiten
  - json_decode()
  - Struktur ist durch Schnittstellenspezifikation klar
- JSON Output an Schnittstellen Design anpassen
  - json_encode()
  - korrekte Header setzen
  - Struktur ist durch Schnittstellenspezifikation klar

Wiederholung: RESTFul

- URLs Routing im Konzept bedenken
- MVC
- gegen Schnittstelle programmieren

Wiederholung: RESTFul

- URLs Routing im Konzept bedenken
  - URLs in Rückgabe für Features der Anwendung
  - PhP ist im Unterschied zu zB. Java stateless
- MVC
  - Controller steuert die Anwendung (Businesslogik)
  - Model verarbeitet Daten und verwendet Services und externe RESTFul Schnittstellen
  - View repräsentiert die Ausgabe im JSON Format
- gegen Schnittstelle programmieren
  - die Schnittstellen sind nach außen bekannt und können “konsumiert” werden

Wiederholung: Implementieren

- Objekte und Konzepte zu Objekten
- MVC Einsatz
- Nameing von Methoden und Variablen
- Code Struktur
- Methoden längen
- Datei Struktur und Namen
- Code Verständlichkeit

Wiederholung: Implementieren

- Objekte und Konzepte zu Objekten
  - Eigenschaften und Methoden
  - Interfaces
- MVC Einsatz
  - Controller steuert die Anwendung (Businesslogik)
  - Model verarbeitet Daten und verwendet Services
  - View repräsentiert die Ausgabe und enthält keine Logik mehr
- Nameing von Methoden und Variablen
  - Der Variablenname spiegelt die Rolle und den Inhalt einer Variable
  - Der Methodenname sagt aus, was in der Methode geschieht und gibt Rückschlüsse auf einen Return Wert
  - Die Methode arbeitet nur auf der logischen Ebene des Namen

Wiederholung: Implementieren

- Code Struktur
  - Unterteilen der Probleme in Klassen und Sub-Methoden
  - Aufbau wie eine Geschichte mit Unterkapiteln
  - Einrücken zur Übersichtlichkeit
- Methoden längen
  - 20 lines of Code
  - nur eine logische Ebene
  - nicht zu tief verschachteln
  - max 2 Parameter übergeben
- Datei Struktur und Namen
  - auch bei Dateien Namen verwenden die rasch auf den Inhalt schließen lassen
- Code Verständlichkeit
  - Code soll lesbar sein wie eine Geschichte, Kommentare sollten nicht notwendig sein
  - KISS: Keep It Simple Stupid

Übung 5

Es soll eine Applikation implementiert werden, die es Flughafenpersonal ermöglicht Flüge und zugehörige Passagierlisten aus einer Datenbank heraus zu lesen. Die Datenbank und ein Service dazu besteht bereits (Download auf Moodle) und die Schnittstellen wurden wie folgt definiert

Definition von Schnittstellen

Folgende Schnittstelle soll das Service bereitstellen:

```
GET Parameter string: action (Mögliche Werte “GET-FLIGHTS”, “GET-PASSENGERS”)

GET Parameter(wenn action “GET-PASSENGERS” ist) int: flightId
```

Output

Fall 1: GET-FLIGHTS (Liste der Flüge, mit RESTFul URL zur jeweiligen Passagierliste):

```
JSON Object:
{
   flights: [
     {flightId: <int>, name: <string>, url: <string>},
     . . .
     ]
}

Bsp. der URL: http://localhost/uebung5/index.php?action=GET-PASSENGERS&flightId=1
```

Output

Fall 2: GET-PASSENGERS (Liste der Passagiere, inkl. Flugname):

```
JSON Object:
{
   fligthName: <string>
   passengers: [
     {lastname: <string>, firstname: <string>},
     . . .
    ]
}
```

Hinweise

- Die Ausgabe wird an den Client als JSON Objekt mit JSON Header übertragen
- Verwenden Sie eine MVC Architektur und überlegen Sie welche Aufgaben Sie den unterschiedlichen Akteuren zuteilen.
- Achten Sie auf klare Namensgebung
- Implementieren Sie sauber
- Machen Sie sich mit der Datenbank vertraut und überprüfen Sie wie die Fieldnames sind.

TODOs

- Konzeption: Entscheidungen, Objekte, etc.
- DB Import und mit den Fields vertraut machen
- Filesystem anlegen - Comments in Files, wofür sie sind
- Schritt für Schritt Implementierung und Tests/Checks
- Anwendungsfälle ganz durchspielen
- Wie könnte das FE aussehen?

Abgabe der Übung 5

- bis 15.06.2020
- im Moodle unter Uebung 5
- NACHNAME_Uebung5.zip (kein *.rar)
  - mit allen Files des Projektes
  - bitte keine versteckten Files die sonst in Ihrem Filesystem vorkommen *.ts, *.ps, usw.
  - Ihr Entwurfskonzept, wie üblich als Scan/Foto/etc.

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