Belegaufgabe 3

RESTful Webservice -
songsWS mit „songs“- und „auth“-Endpoints

Fertigstellungstermin: Dienstag, 17.12.2019 @ 06:00 Uhr

Vorstellung des Belegs: In Ihrer Übung am Di, 17.12. bzw. Mi, 18.12.2019

Der Prototyp für einen Songs-Webservice ist vom Management gut aufgenommen worden. Jetzt geht es darum, den Prototyp in einen vollständigen RESTful Webservice umzuschreiben.

Teil 1: Implementierung des Endpoints /songsWS-TEAMNAME/rest/songs

- Der Endpoint für Ihren neuen Songs-Service ist http://localhost:8080/songsWS-TEAMNAME/rest/songs
- Dieser Endpoint soll die vier HTTP-Methoden GET, POST, PUT, DELETE implementieren und **JSON und XML** als **Ein- bzw. Ausgabeformate** handhaben können. Es gilt:

  GET `http://localhost:8080/`songsWS-TEAMNAME`/rest/songs`
  Ihr Service schickt alle Songs zurück

  GET `http://localhost:8080/`songsWS-TEAMNAME`/rest/songs/1`
  Ihr Service schickt Song mit Id = 1 zurück, 1 <= songId <= 10

  POST `http://localhost:8080/`songsWS-TEAMNAME`/rest/songs`
  mit Song in der Payload
  Wenn erfolgreich, legt Ihr Service den neuen Song an und schickt Statuscode 201 und URL zur neuen Ressource im „Location“-Header zurück

  PUT `http://localhost:8080/`songsWS-TEAMNAME`/rest/songs/1`
  mit Song in der Payload
  Ihr Service führt das Song-Update durch, aber nur wenn songId in URL gleich Id in Payload ist & „title“-Attribute darf nicht leer sein; wenn Update erfolgreich, dann nur Statuscode 204, ansonsten 400er Statuscode zurückschicken

  DELETE `http://localhost:8080/`songsWS-TEAMNAME`/rest/songs/1`
  Ihr Service löscht den Song, aber der Song mit der songID aus der URL muss existieren; wenn Löschen des Songs erfolgreich, dann nur Statuscode 204 zurückschicken, ansonsten 400er Statuscode.

- Als DB bitte eine relationale Datenbank, MySQL oder Postgres, nutzen. Diese Datenbank soll eine Tabelle für die Songs enthalten. In diese Tabelle sollen die 10 Songs aus songs.json geladen werden.

- Bitte immer daran denken, dass die Anfragen Ihrer Clients fehlerhaft sein können und dass diese Fehler, keine Serverfehler (500er Statuscode) erzeugen sollten. Also, **erst Test Cases überlegen**, **dann implementieren!**

- Das muss gemacht werden:
  1. Framework Jersey nutzen
  2. Eine relationale DB und das Framework Hibernate nutzen
  3. Dependency Injection der EntityManagerFactory und der DAOs in die entsprechenden Objekt (Vorlesungen am 3.12. & 10.12.)
  4. Übersichtliche package-Struktur Ihres Projektes, Packages von Klassen und ihren Testklassen müssen übereinstimmen

Teil 2: Implementierung des Endpoints /songsWS-TEAMNAME/rest/auth

Erst wenn Sie mit Teil 1 fertig sind, dürfen Sie mit Teil 2 anfangen. Hier soll es um den 'auth'-Endpoint Ihres Webservices gehen.

a) In Ihrer Datenbank legen Sie sich bitte eine Tabelle für die Nutzer/innen Ihres Webservices an:

```
CREATE TABLE IF NOT EXISTS `user` (
        `userId` VARCHAR(50) NOT NULL ,
        `key` VARCHAR(50) NOT NULL ,
        `firstName` VARCHAR(50) NOT NULL ,
        `lastName` VARCHAR(50) NOT NULL ,
        PRIMARY KEY (`userId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Achtung: Das ist MySQL-SQL und funktioniert in Postgres nicht.

b) Die Tabelle muss diese zwei Nutzerinnen enthalten:

userId = mmuster, key=ENCRYPTEDKEY1, firstName=Maxime, lastName=Muster
userId = eschuler, key= ENCRYPTEDKEY2, firstName=Elena, lastName=Schuler

Sie können sich auch noch andere User anlegen, aber die beiden obigen müssen genau so in Ihrer Datenbank vorhanden sein.

c) Mit der HTTP-Anfrage "GET /songsWS-TEAMNAME/rest/auth?userId=mmuster&key= ENCRYPTEDKEY1" soll sich die Nutzerin mmuster bei Ihrem Webservice anmelden.
Mit der HTTP-Anfrage "GET /songsWS-TEAMNAME/rest/auth?userId=eschuler&key= ENCRYPTEDKEY2" soll sich die Nutzerin eschuler bei Ihrem Webservice anmelden.
Ist die User-Anmeldung erfolgreich, d.h., userId/key -Kombination existiert in Ihrer DB, dann generiert Ihr Service einen Authorization-Token und schickt diesen im Message-Body der Antwort zurück.
Beispiele:

- GET /songsWS-TEAMNAME/rest/auth?userId=mmuster&key= ENCRYPTEDKEY1

  User "mmuster" mit Key " ENCRYPTEDKEY1" existiert in Ihrer DB.

  Antwort des Servers:
  Content-Type: "text/plain"
  qwertyuiiooxd1a245 (Beispiel eines generierten Tokens)

- GET /songsWS-TEAMNAME/rest/auth?userId=mmuster&key= SOMEOTHERKEY
  Die Kombo "mmuster/ SOMEOTHERKEY" gibt es in Ihrer DB nicht. Antwort des Servers: HTTP-Statuscode 401 (User kann nicht authentifiziert werden)

d) Nachdem sich User 'mmuster' erfolgreich bei Ihnen angemeldet hat, muss mmuster dieses Token **in allen nachfolgenden Anfragen an Ihren Service mitschicken**, im HTTP-Header "Authorization". Zum Beispiel, die Anfrage "Gib mir alle Songs in JSON" braucht jetzt auch einen "Authorization"-Header und muss so aussehen:

```
GET /songsWS-TEAMNAME/rest/songs
Accept: application/json
Authorization: qwertyuiiooxd1a245
```

Der Server kann nur dann mit HTTP-code 200 und der Liste aller Songs antworten, wenn ein gültiger Token im "Authorization"-Header vorhanden ist. Falls kein oder kein gültiger Token vorhanden ist, muss Ihr Service mit HTTP-Code 401 antworten.