Belegaufgabe 3

A RESTful Web Service with Jersey (JAX-RS)
songsRX mit „songs“- und „auth“-Endpoints

Abgabetermin: Ihre Übung am Mittwoch, 12.12. am Donnerstag, 13.12. und am Montag, 17.12.

Teil 1: Implementierung des Endpoints /songsRX/rest/songs

- Die Songs-DB besteht wieder aus den 10 Einträgen in songs.json
- Dieses JSON-File sollte geladen werden und als eine In-Memory-“Datenbank” Ihrem REST-Service zur Verfügung stehen.
- Der Endpoint für Ihren Service soll sein: http://localhost:8080/songsRX/rest/songs
- Ihr Service soll die vier HTTP-Methoden “GET, POST, PUT, DELETE” implementieren, und JSON und XML als Ein- bzw. Ausgabeformate handhaben können. Zum Beispiel:

  GET Accept:application/json http://localhost:8080/songsRX/rest/songs - schickt alle Songs in JSON zurück
  GET Accept:application/xml http://localhost:8080/songsRX/rest/songs/7 - schickt Song 7 in XML zurück

  POST Content-Type:application/xml http://localhost:8080/songsRX/rest/songs (mit einer XML-Payload) - trägt den Song in der DB ein. Falls erfolgreich, dann Status-Code 201 und liefert die URL für den neuen Song im “Location”-Header an den Client zurück.

  POST Content-Type:application/json http://localhost:8080/songsRX/rest/songs (mit einer JSON-Payload) - trägt den Song in der DB ein. Falls erfolgreich, dann Status-Code 201 und liefert die URL für den neuen Song im “Location”-Header an den Client zurück.

  PUT Content-Type:application/xml http://localhost:8080/songsRX/rest/songs/7 (mit einer XML-Payload) - erneuert den Eintrag für Id 7 in der DB und schickt “on Success” den HTTP-Statuscode 204.

  PUT Content-Type:application/json http://localhost:8080/songsRX/rest/songs/7 (mit einer JSON-Payload) - erneuert den Eintrag für Id 7 in der DB und schickt “on Success” den HTTP-Statuscode 204.

  DELETE http://localhost:8080/songsRX/rest/songs/7 - löscht den Eintrag für Id 7 in der DB und schickt on Success einen HTTP-Statuscode 204.

- **Unit-Tests:** Da es mal wieder sehr viele Testfälle gibt, reicht es mir, wenn Sie die **Testfälle für die PUT-Methode** abdecken.

- Was ist noch wichtig:

  1. Nutzen Sie Dependency Injection
  2. Übersichtliche package-Struktur Ihres Projektes
  3. packages von Klassen und ihren Testklassen müssen übereinstimmen
  4. Klassennamen müssen großgeschrieben werden.

Teil 2: Implementierung des Endpoints /songsRX/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. Bitte legen Sie sich einen User-Datenbestand in einem JSON-File an. Dieser könnte so aussehen:

{“id”:1, “userId”:”mmuster”,”lastName”:”Muster”,”firstName”:”Maxime”}

{“id”:2, “userId”:”eschuler”,”lastName”:”Schuler”,”firstName”:”Elena”}

Mit der HTTP-Anfrage “GET /songsRX/rest/auth?userId=mmuster” soll sich der User ‘mmuster’ für unseren Service einen Authorization-Token generieren lassen. Dieser Token soll dann bei allen nachfolgenenden Anfragen mitgeschickt werden und von Ihrem Service auf seine Gültigkeit geprüft werden. Also:

> GET /songsRX/rest/auth?userId=mmuster
>
> Fall 1 - mmuster gibt es als User nicht: Antwort des Servers: HTTP-Statuscode 403 (User hat keine Berechtigung auf Ihrem Server)
>
> Fall 2 - mmuster ist ein bestehender User: Antwort des Servers: 123456789qwertyuiiooxd1a245 als “text/plain” (Ihr Service generiert diesen Token.)

User ‘mmuster’ muss diesen Token **in allen nachfolgenenden Anfragen an Ihren Service mitschicken** und zwar im HTTP-Header “Authorization”. Zum Beispiel, die Anfrage “Gib mir alle Songs” braucht jetzt auch einen “Authorization”-Header und sollte so aussehen:

> GET /songsRX/rest/songs HTTP/1.1
>
> Authorization: 123456789qwertyuiiooxd1a245

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

- **Unit-Tests** für die GET-Methode Ihres “auth”-Endpoints müssen vorhanden sein.
- Was ist noch wichtig:

    1. Nutzen Sie Dependency Injection
    2. Übersichtliche package-Struktur Ihres Projektes
    3. packages von Klassen und ihren Testklassen müssen übereinstimmen
    4. Klassennamen müssen großgeschrieben werden.