Servlet life cycle and thread safety

Fertigstellungstermin: Ihr Code muss spätestens am Dienstag, 26. November 2019 um 6:00 Uhr im git eingecheckt sein. Das gilt für Zug 1 und Zug 2. Die Belegabgabe findet am Dienstag, 26.11. und am Mittwoch, 27.11. in den Übungen statt.

Als PO möchte ich einen Proof of Concept durchführen und einen einfachen Prototype für einen Webservice erstellen.
Anforderungen:

1. Der Webservice soll die Servlet API nutzen.
2. Sie müssen das Projekttemplate in songsservlet.zip nutzen. Alle „TEAMNAME"-Referenzen in der pom.xml und im Code müssen entsprechend angepasst werden.
3. Der Webservice soll auf eine In-Memory-Datensammlung mit Song-Einträgen zugreifen. Die initiale Songssammlung von 10 Songs befindet in der songsservlet/src/main/resources/songs.json-Datei (Quelle für die 10 Songs ist https://www.cheatsheet.com/entertainment/the-worst-songs-of-all-time.html/). Diese Datei soll beim Laden des Servlets gelesen werden und die 10 Songs in eine In-Memory-Datensammlung laden.
4. Der Webservice soll **folgende Anforderungen erfüllen**:

GET-Requests:

- **GET** `http://localhost:8080/songsservlet-TEAMNAME/songs?songId=6` (*)
  `Accept: application/json`
  soll den Song 6 in JSON-Format zurücksenden
- **GET** `http://localhost:8080/songsservlet-TEAMNAME/songs?songId=6` (*)
  `Accept: application/xml`
  soll den Song 6 in XML-Format zurücksenden
- **GET** `http://localhost:8080/songsservlet-TEAMNAME/songs?songId=22` werden von Ihrem Service mit dem Status Code 404 abgewiesen (Einen Song 22 gibt es nicht).
- GET-Requests, die den HTTP-Parameter „`songId`" nicht enthalten, werden von Ihrem Service mit dem Status Code 400 abgewiesen.
- GET-Requests ohne Accept-Header bekommen die angeforderten Daten in JSON zurück.
- GET-Requests mit Accept-Header-Wert, welcher weder „application/json" noch „application/xml" ist, werden von ihrem Webservice mit dem HTTP-Statuscode 406 abgewiesen werden.
- **`htwb.ai.TEAMNAME.controller.GitVersionServlet` in songsservlet.zip ist schon vorimplementiert und darf nicht geändert werden (außer package-Namenanpassung)!**

PUT-Requests: Mit diesem Request

- **PUT** `http://localhost:8080/songsservlet-TEAMNAME/songs?songId=6` (*)
  `Content-Type: application/json`
  mit Payload

```
{
    "id": 6,
    "title": "Wrecking Ball",
    "artist": "MILEY CIRUS",
    "label": "RCA",
    "released": 2013
}
```

möchte der Client den Song 6 in Ihrer DB updaten und speichern. Falls das Update erfolgreich ausgeführt wurde, soll Ihr Service nur den Status Code 204 zurückschicken, mit leerem Response-Body!
PUT akzeptiert nur JSON-Payloads. Payloads die nicht gelesen bzw. geparst werden können, werden von Ihrem Service mit dem Status-Code 400 abgewiesen.

5. Wenn das Servlet vom Container entsorgt wird bzw. wenn der Server herunterfährt, soll der Inhalt Ihrer "Datenbank" wieder in der **gleichen** songs.json-Datei, die bei der Initialisierung des Servlets eingelesen wurde, gespeichert werden. Diese Datei soll die geänderten Songs, die Ihr Service gespeichert hat, enthalten und wird bei der nächsten Initialisierung des Servlets wieder geladen werden.

6. Unit-Tests: Die Servlet-Methoden, "**init**" und "**doGet**" und deren Test-Cases müssen mit Unit-Tests abgedeckt sein. Die Unit-Tests werden bewertet. Hinweis: **Test First!** Überlegen Sie sich erst alle Testfälle und schreiben Sie die entsprechenden Unit-Tests zuerst, wenigstens ansatzweise und erst dann den Servlet-Code.

(*): `songId=6` ist ein Beispiel. Es gilt: $1 \leq songId \leq 10$