SER422 Spring 2019 Lab5: Assigned 4/10, due 4/18 @ 11:59:00pm via Canvas

This lab is shorter than your other labs, so notice the shorter deadline.

**Task 1 (67 points): Convert a JAX-RS/Jersey example to Spring MVC / Spring Boot**

The webpage <https://crunchify.com/how-to-build-restful-service-with-java-using-jax-rs-and-jersey/> presents a simple web example of a Fahrenheit to Celsius converter, and vice-versa. For this task, you are to:

1. Convert this JAX-RS/Jersey example to use Spring MVC annotations. The conversion should be such that the EXACT same results are returned, with the lone exception that the String output before the JSON in the F to C result is removed (leaving only JSON returned).
2. Add a query string parameter “precision” to both the F to C and C to F conversions methods that indicates the level of precision (number of digits to the right of the decimal point) that should be in the converted value. The query string parameter is optional.
3. Deploy your service API as a standalone executable in Spring Boot.

**Task 2 (33 points): Deploy your Spring Boot webapp from Task 1 in Docker**

As the title of the Task indicates, take your Spring Boot application from Task 1, #3 and deploy it within a Docker image. When starting a container from the image the service API should be running.

Your submission for this Task is a Dockerfile. You do not need a bash script for this task as in Lab 4.

**Extra Credit (50 points possible):**

As mentioned in class and available on the portal, online SER faculty associate Mr. Diego del Blanco put together a series of Spring Boot walkthrough videos that demonstrate a sample application using Spring Boot with various starters, including Spring MVC and Spring JPA (Java Persistence Architecture). Review these videos, and the sample project. Then create an app using the basic “web” spring boot library (no need of pagination or hateoas links) like in the master branch of the example code (<https://github.com/ddelblanco/restapilab> )

With a REST API for these entities:

* Movie (Id, Title, Year, Plot, Genres. Oscars, Directors)
* Oscars (Id, Year, Category, Movie)
* Director(Id, Name, LastName, DataBirth. Movies)

**Constraints:**

* A movie should have none, one, or more than one director
* A movie should have none, one, or more than one Oscars
* A director can direct none, one, or more movies

**Deliverable:** A complete app (executable jar) with the spring boot complete code. Assume JPA with a MySQL database.

* The jar files for each of the codes that should run with java -jar nameofthejar.jar

A collection of HTTP requests with the needed calls to demonstrate how the API works (all appropriate verbs to all available endpoints). You can get this from postman if you use that tool.

**Submission:**

For submission, you should submit to Canvas a single zipfile named lab5\_<asurite1>.zip that has within it the source tree for Task 1 and the Dockerfile for Task 2. Additionally, it should have a README.txt in the root directory when expanding the zipfile.

* Source trees are the directory contents of your development directories without the compiled artifacts (no “classes” or “bin” subdirectories) and no WAR files. All of your \*.java, properties, and static resources should be in this file. Your clean target should take care of this.
* Given you are using Spring Boot, we expect a pom.xml (Maven) or build.gradle (Gradle) to be provided in accordance with the targets, structure, and dependencies common to a Spring Boot project. Whicever you give us should be self-contained and work “out of the box”.
* Put a README.txt file at the root of your source tree to tell us anything you think we need to know to run your lab.
* If you do the extra credit, please put the project in a directory named EC under the root of your zipfile submission, and name the web app context EC.