SOEN 487: Web Services and Applications

Lab Instructions

Handling Complex Data

February 2, 2022

1 General Information

Lab Date: Wednesday, February 2rd, 2022.
Your lab instructor will provide you with instructions on how to do this lab activity.

2 Introduction

The purpose of this lab is to modify the existing REST API to handle more complex data. This lab will be preferrably performed in pairs, and at the end, you will demonstrate your results as outlined, to your lab instructor.

3 Project Setup

There are 2 ways to set up your project to start this lab.

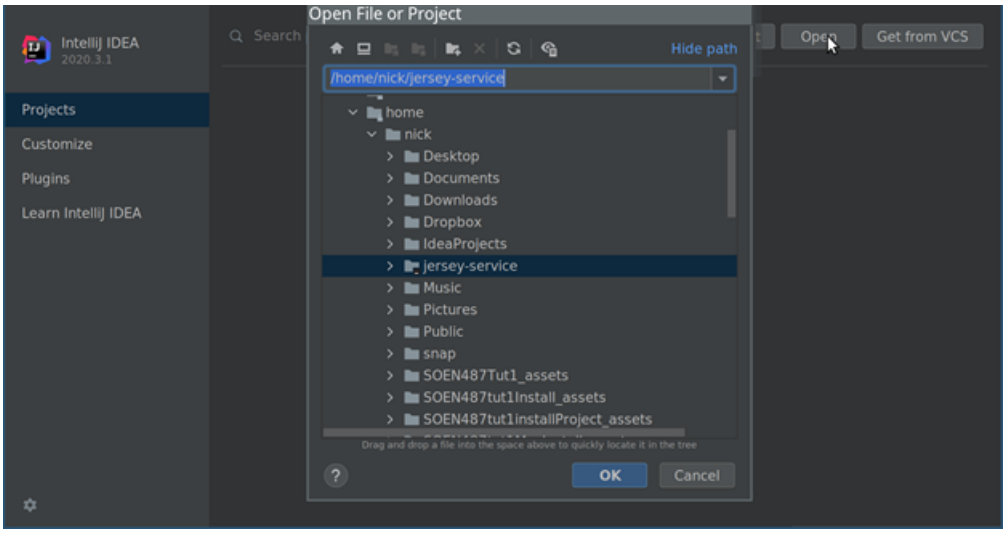
1. If your code worked in the previous lab as verified by your lab instructor then you can reuse that code, but you will need to add the following dependency to your pom.xml file within the dependencies tag:

```
 <dependency>
   <groupId>org.glassfish.jersey.media</groupId>
   <artifactId>jersey-media-moxy</artifactId>
</dependency>
<dependency>
   <groupId>org.glassfish.jersey.media</groupId>
   <artifactId>jersey-media-multipart</artifactId>
   <version>2.17</version>
</dependency>
```

2. Clone the template for the lab that your instructors made for you:
   `https://github.com/SOEN487/LAB03`

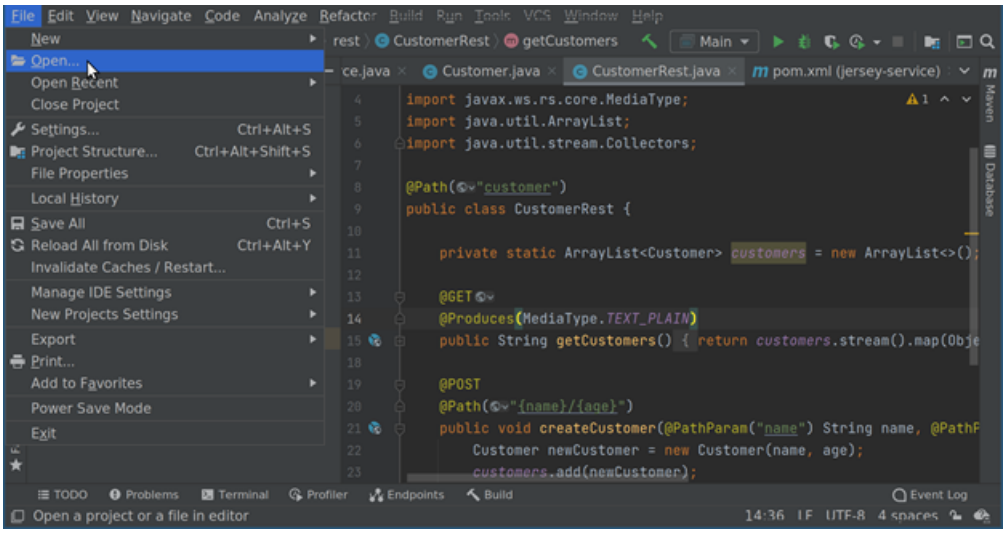
4 Opening the Project with IntelliJ IDEA

If you don't have a project open already, you can open a new one by clicking on open on the top right of your window and then proceed to choose the location where the directory of your generated template is located.



If a project is already open, you may open a new one by navigating to the top left of

your IDE, clicking on File and, subsequently on open. It will then be possible to navigate to the location of your generated web server template, to open it.



5 Instructions

In this tutorial, you will need to add the following to your previously started REST API:

- Add a new POST request that will take **form parameters** as input with the fields **"title"**, **"author"** and **"isbn"** all of which will be of type `String`.
- Add marshalling and unmarshalling from XML (or JSON) for the already implemented **PUT** and **GET** requests.

6 Bonus

For this tutorial, the bonus will be to create a new API with the ability to upload and download files. The file or file(s) (it is not necessary to hold more than 1 file) will be held in a however you would like as long as it is held in memory instead of in the file system. An example of a possible class to hold it in is the `ByteArrayStream` (see references for details).

7 Testing Your Lab

To demonstrate that your API works, you can open up Postman, or use Curl to demonstrate that all your endpoints function properly. Once everything is working, please advise your lab instructor to demonstrate your results.

References

1. REST with Java (JAX-RS) using Jersey:

   `https://www.vogella.com/tutorials/REST/article.html`

2. Apache HTTP Client:

   `https://www.tutorialspoint.com/apache_httpclient/apache_httpclient_overview.htm`

3. ByteArrayStream:

   `https://docs.oracle.com/javase/8/docs/api/java/io/ByteArrayInputStream.html`

   `https://docs.oracle.com/javase/7/docs/api/java/io/ByteArrayOutputStream.html`