Kuwait University
College of Computing Sciences and Engineering
Department of Information Science

COMP 3095 – Web Application Development with Java

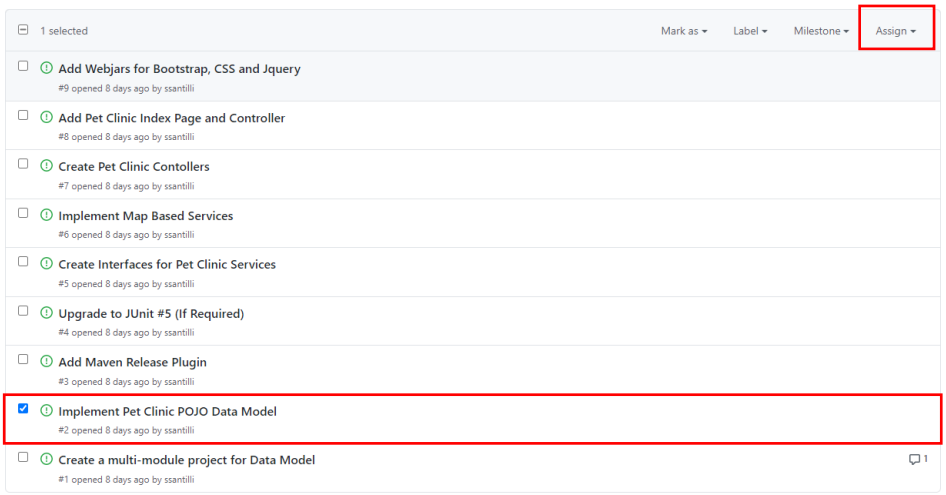
Lab 3
Spring Pet Clinic
POJO Data Model / Multi-Module Maven Builds

Table of Contents

Laboratory #3 – POJO Data Model

1. Laboratory Objective

The goal of this lab is to extend our previous lab and implement the Pet-Clinic data model. Eventually the POJOs we create, will become entities in the backend pet-clinic database.



2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:
   a. Create a POJO data model for a application project
   b. Conduct a multi-model maven build

3. Laboratory Instructions

a. Open the COMP3095 - Pet-Clinic lab 2 in your IDE

b. Start by creating a **Person** class in a new package called **model** (add new class to your git). The Person class should contain the following:

   i. first name
   ii. last name
   iii. setters/getters

c. Create a class to represent Vetranarians called **Vet**, vetranarians are people, so as a consequence, this class should extend Person. No additional properties are required for Vet at this point.

d. Create a class to represent owners called **Owner**, this class should also extend Person. No additional properties are required for Owner at this point.

e. Create a class called **PetType**. PetType should contain the following properties:
   i. Name
   ii. Setter/Getter

f. Create a class to represent pets called **Pet**. A Pet should have the following properties (at a minimum):
   i. petType
   ii. owner
   iii. birthdate

g. Commit your code (if you chose to create a github account). Add "**Closes #2**" – to commit comments to close issue #2 in your git repository automatically.

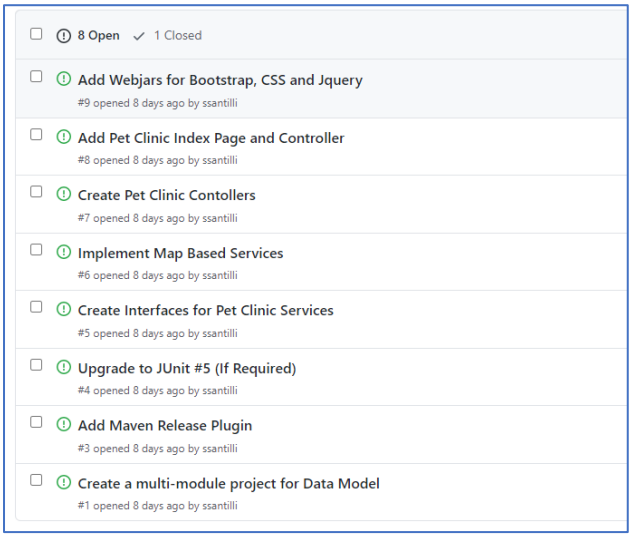
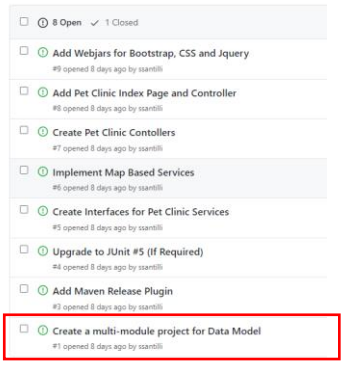


Figure 1: Issue #2 is now closed

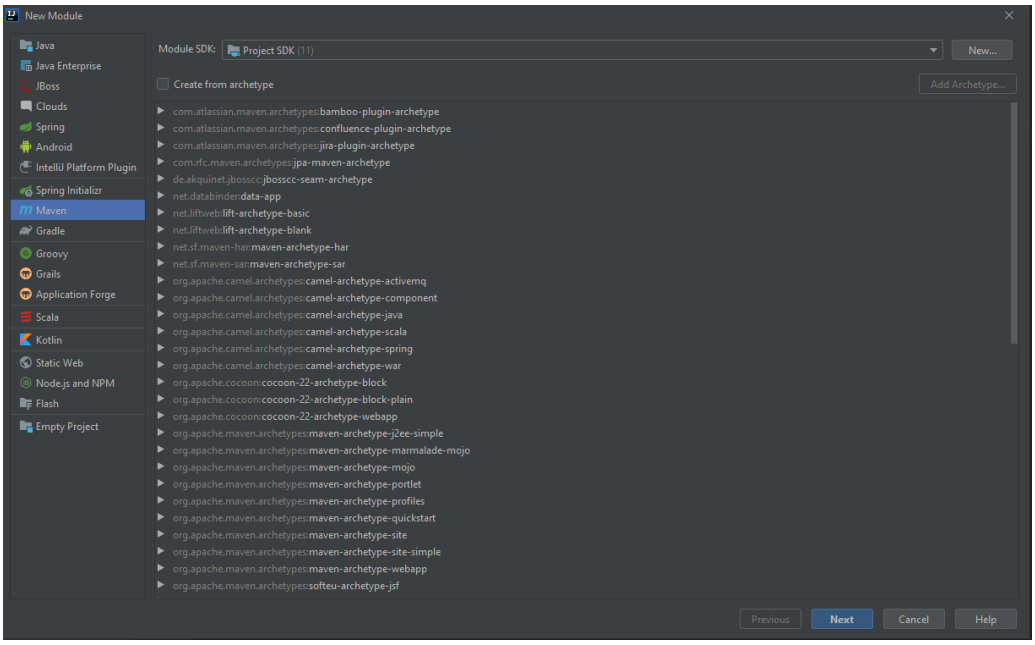
Create Multi-Module Maven Builds

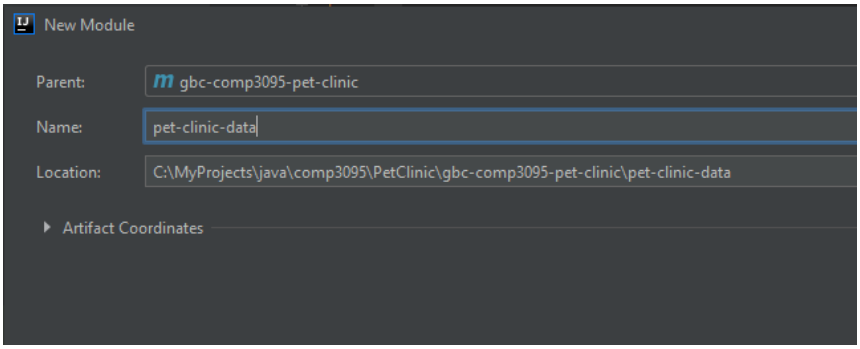
As you begin to work on more complex projects, its to your benefit to partition them into multiple modules. For this portion of the lab, we will look to break our application into two additonal modules, 1 for the **data** (persistence portion) and another for the **web** portion.

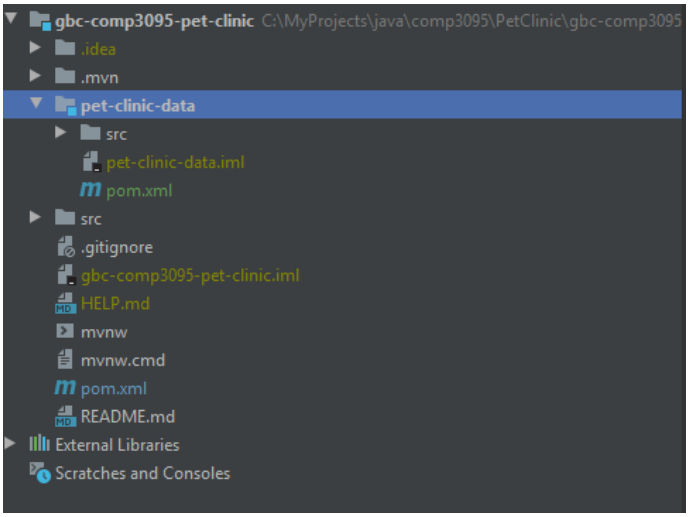


a. Assign yourself issue #1 "**Create a multi-module project for Data Model**"

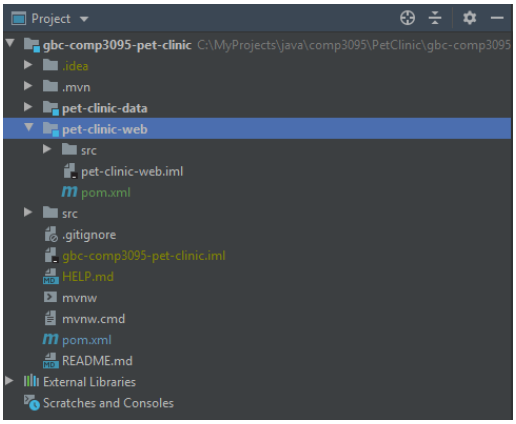
b. Open your project, and create a new module called, **pet-clinic-data**







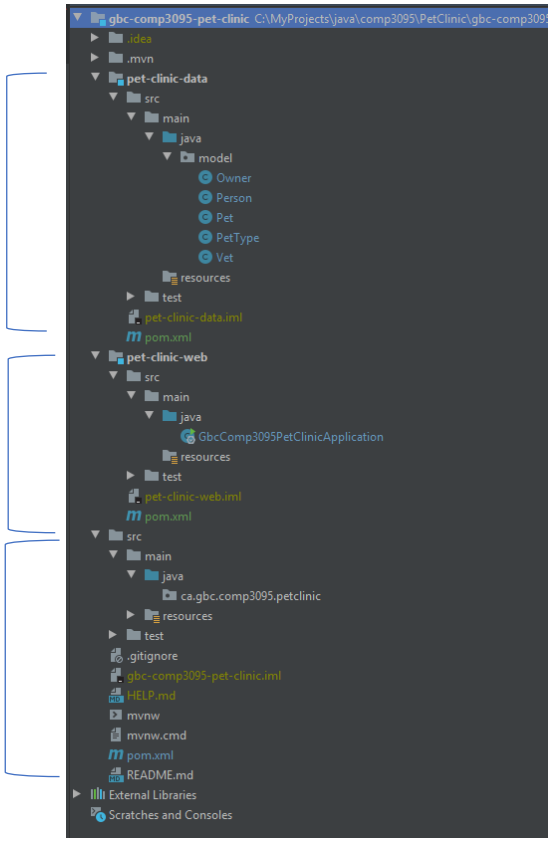
c. Using the same process as above, create another new module called, **pet-clinic-web.**



d. Reviewing your project, you should note that your project now has two modules (1 for data and another for web). Each module capartamentalizes the applications functionality by their respective logic.

e. Refactor the project further, to place to the class files into their respective modules:

i. The **main** class should be moved to the **pet-clinic-web** module
ii. **Owner**, **Person**, **Pet**, **PetType** and **Vet** should be moved to the **pet-clinic-data** module.



f. Open up the pet-clinic (parent) **pom.xml** file and move the following from the **parent** pom.xml to the **web** pom.xml:

i. **spring-boot-actuator**
ii. **spring-boot-starter-thymeleaf**
iii. **sprint-boot-starter-web**
iv. **sprint-boot-devtools**
v. **spring-boot-starter-test**

g. Open up the **parent** project pom.xml file and move the following from the **parent** pom.xml to the **data** module pom.xml.

i. **spring-boot-starter-data-jpa**
ii. **h2**
iii. **mysql**
iv. **projectlombok**
v. **spring-boot-starter-test**

h. Open up the **web** module pom.xml and create a dependency in the **web** module pom.xml, for the **pet-clinic-data** module.

```
<!-- pet-clinic-data module dependency-->
<dependency>
    <artifactId>pet-clinic-data</artifactId>
    <groupId>ca.gbc.comp3095</groupId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
```

i. Edit the **pet-clinic-data** pom.xml to package the module as a **jar** file. This will create a lean jar file (and not a bloated one):

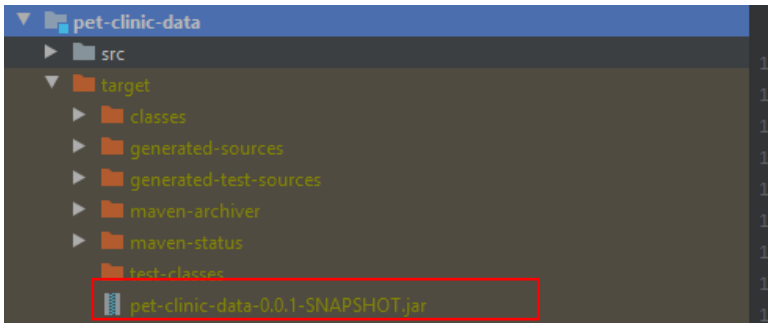
```
<properties>
    <spring-boot.repackage.skip>true</spring-boot.repackage.skip>
</properties>
```

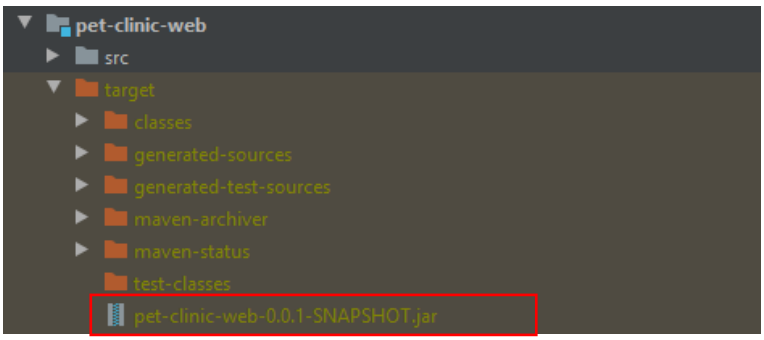
```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <executions>
                <execution>
                    <goals>
                        <goal>repackage</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

j. Run the maven clean lifecycle of the pet-clinic (**parent**) project.

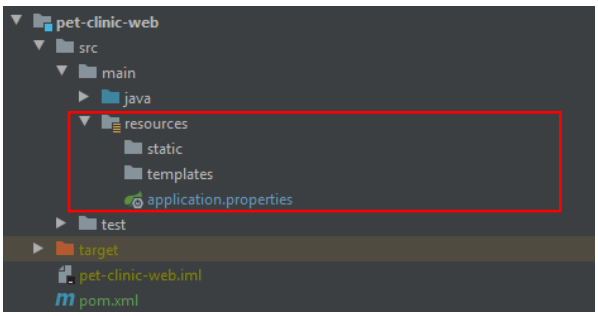
```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for gbc-comp3095-pet-clinic 0.0.1-SNAPSHOT:
[INFO]
[INFO] gbc-comp3095-pet-clinic ........................... SUCCESS [  0.836 s]
[INFO] pet-clinic-data ................................... SUCCESS [  0.020 s]
[INFO] pet-clinic-web .................................... SUCCESS [  0.028 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

k. Run the pet-clinic **parent** project maven **package** lifecycle. This will compile eveything in the **parent** pet-clinic project, then the **pet-clinic-data**, then **pet-clinic-web**. You will note the **pet-clinic-data** and **pet-clinic-web** jar files have now been created. Specifically, the **pet-clinic-web** jar, will include the jar for the **pet-clinic-data** module as a dependency.

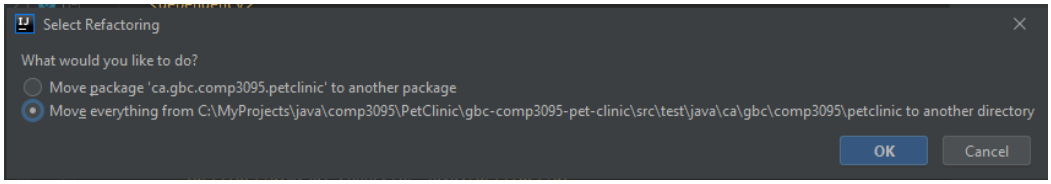


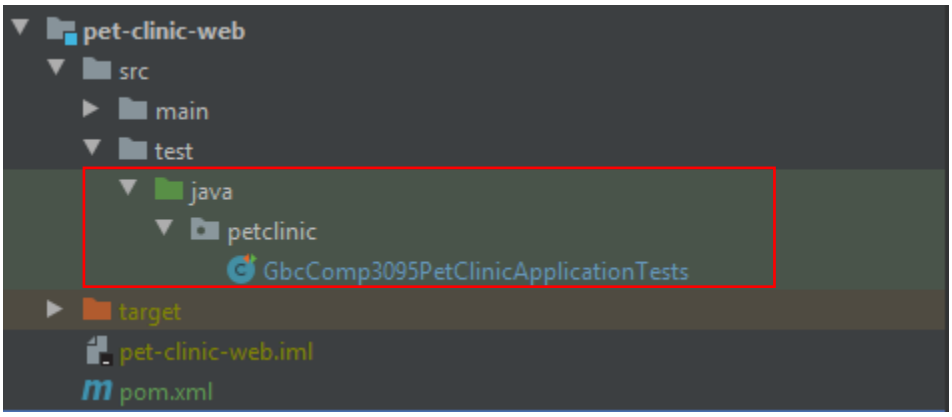


l. Move the **pet-clinic** (parent) resources (**static, templates and appplication properties**) to **the pet-clinic-web** resources location.



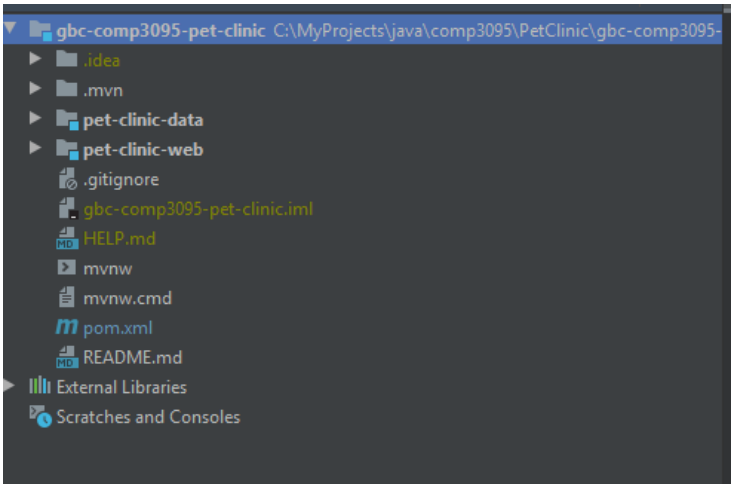
m. Move the **pet-clinic** (parent) **test/java/*** folder contents to **the pet-clinic-web test/java/*** location.





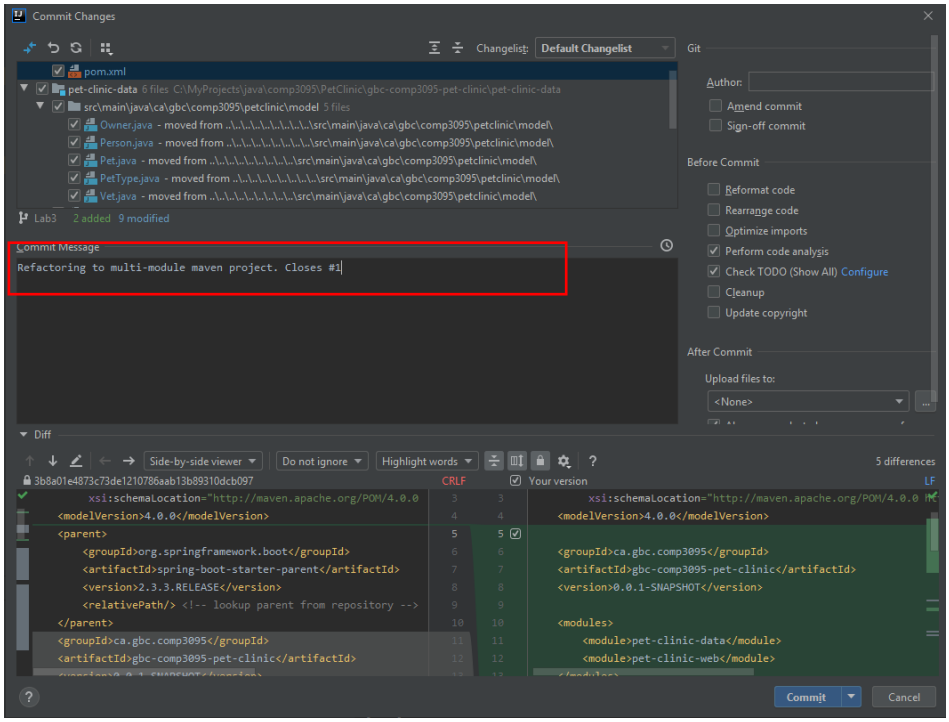
n. Delete the **src** folder from the **pet-clinic** (parent) project, as it is no longer needed.

What you should be left with is a **pet-clinic** (parent) project, that has distinct modules for both **web** and **data** respectively. Each compartmentalizing the functionlity into their respective modules.



This concludes setting up a multi-module project.

o. If you are commiting your code in yout own code repo, now would be a good time to commit your code ☺


Commit Changes
Changelist: Default Changelist
Git
Author:
Amend commit
Sign-off commit
pom.xml
pet-clinic-data 6 files C:\MyProjects\java\comp3095\PetClinic\gbc-comp3095-pet-clinic\pet-clinic-data
src\main\java\ca\gbc\comp3095\petclinic\model 5 files
Owner.java - moved from ..\..\..\..\..\..\..\..\..\src\main\java\ca\gbc\comp3095\petclinic\model\
Person.java - moved from ..\..\..\..\..\..\..\..\..\src\main\java\ca\gbc\comp3095\petclinic\model\
Pet.java - moved from ..\..\..\..\..\..\..\..\..\src\main\java\ca\gbc\comp3095\petclinic\model\
PetType.java - moved from ..\..\..\..\..\..\..\..\..\src\main\java\ca\gbc\comp3095\petclinic\model\
Vet.java - moved from ..\..\..\..\..\..\..\..\..\src\main\java\ca\gbc\comp3095\petclinic\model\
Lab3 2 added 9 modified
Before Commit
Reformat code
Rearrange code
Optimize imports
Perform code analysis
Check TODO (Show All) Configure
Cleanup
Update copyright
Commit Message
Refactoring to multi-module maven project. Closes #1
After Commit
Upload files to:
<None>
Diff
Side-by-side viewer
Do not ignore
Highlight words
5 differences
3b8a01e4873c73de1210786aab13b89310dcb097
CRLF
Your version
LF
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
<modelVersion>4.0.0</modelVersion>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.3.3.RELEASE</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>ca.gbc.comp3095</groupId>
<artifactId>gbc-comp3095-pet-clinic</artifactId>
<version>0.0.1-SNAPSHOT</version>
<modules>
<module>pet-clinic-data</module>
<module>pet-clinic-web</module>
Commit
Cancel