**Dependency Management using Maven- Exercise**

**Name**: Sharda Kumari

**JVM Trainee**

23/02/2022

**Following parameters are to be considered for exercise submission:**

1. **The complete project has to be submitted.**
2. **.idea, target directories should be excluded.**
3. **A text file containing the relevant URLs which are references and any other documentation.**

**Questions:**

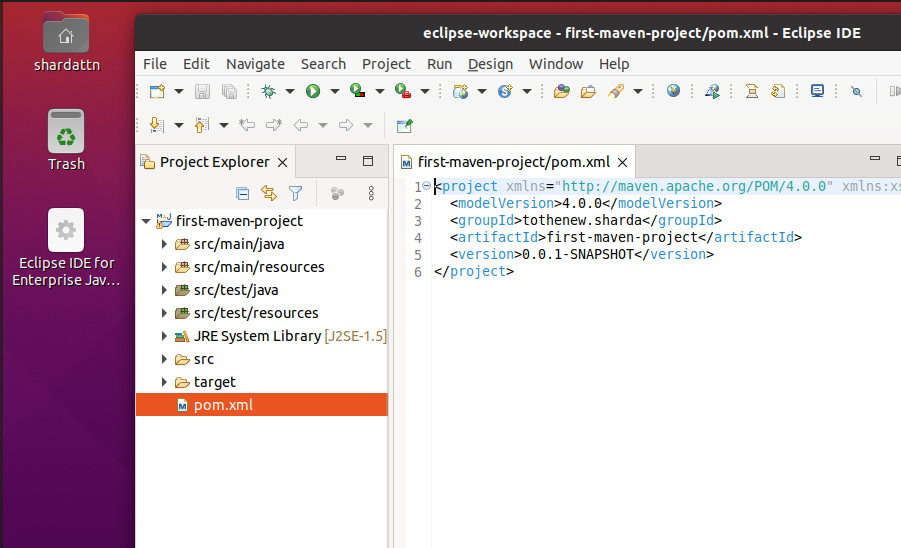
**1. Add a maven dependency and it’s related repository URL.**

First of all open Eclipse EE and create new maven project.

Now skip the archetype as we will be creating simple project.

Provide groupId and artifactId leave everything to default, hit Finish.

Project is ready,

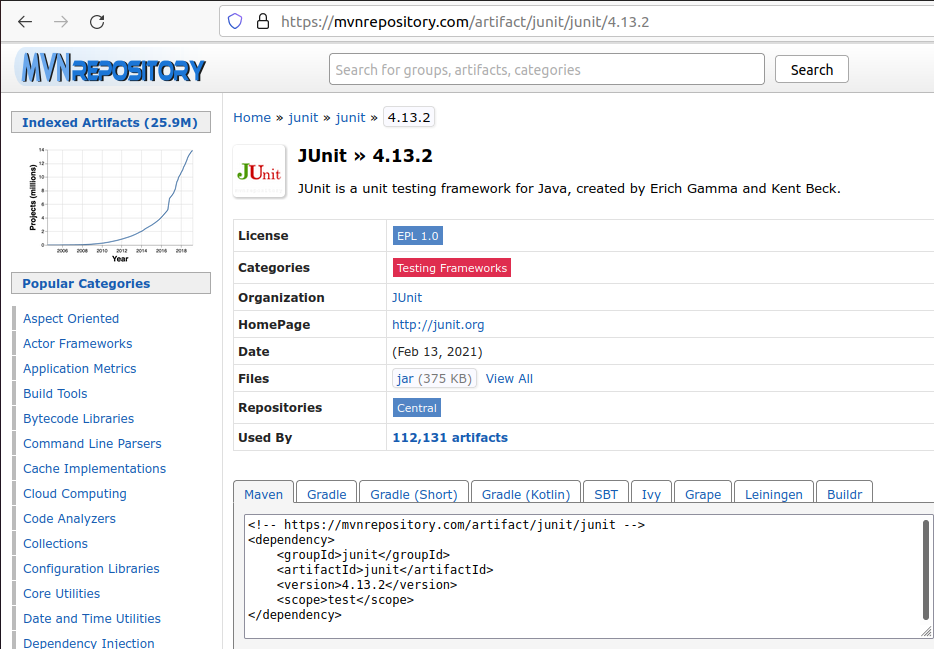


To add a maven dependency, visit <https://mvnrepository.com/>

And search for the dependency you want to add to your project

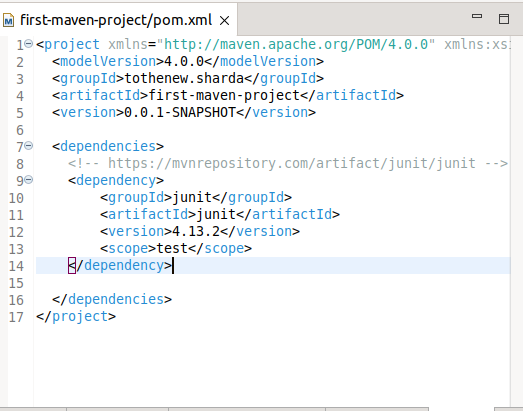
I want to add junit dependency.

So I will search it on maven repository site.



Copy the maven code.

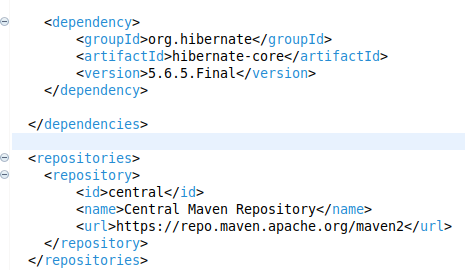
Add it in pom.xml under dependencies tag and our pom.xml looks like this.



Now as you can see we added a maven dependency and it’s related repository URL is already there as a comment in the dependency tag.

**2. Add a new repository in the pom.xml and use it’s dependencies.**

Let’s add central maven repository and add hibernate dependency in our pom.xml.



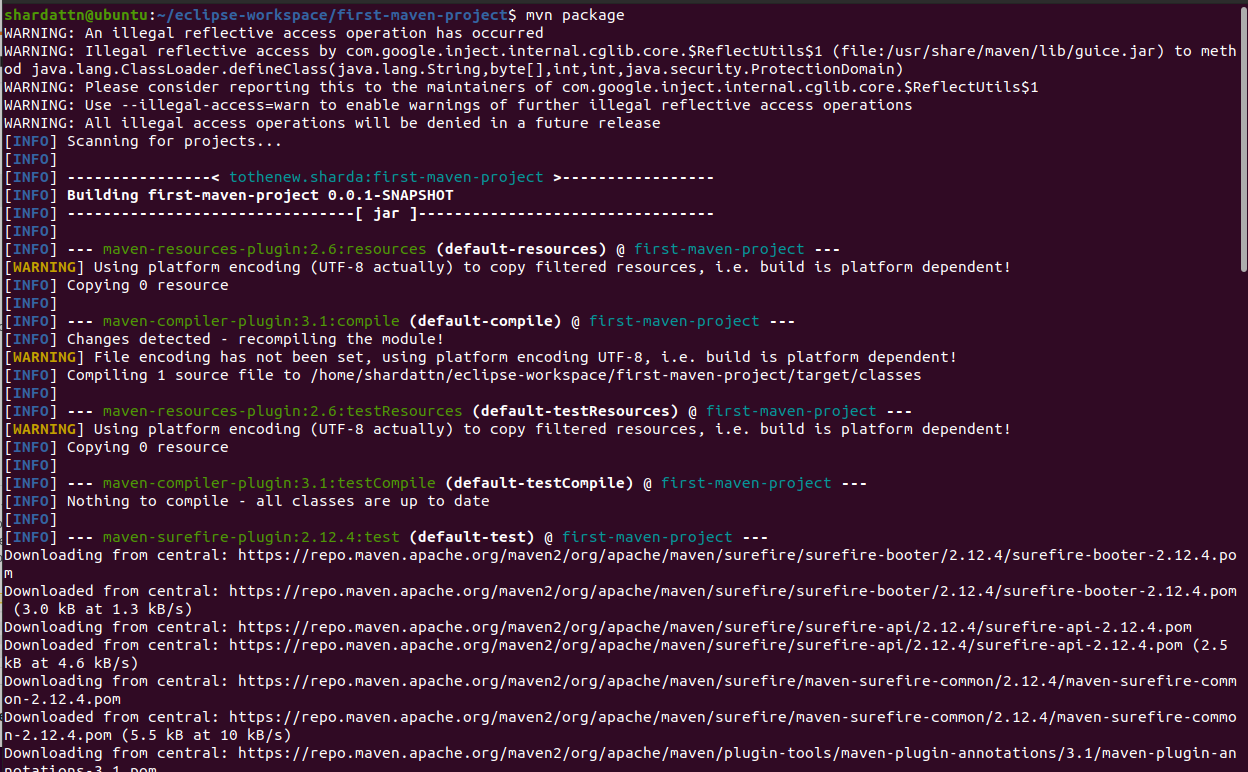
**3. Using JAR plugin, make changes in the pom.xml to make the jar executable. Using java -jar JAR\_NAME, the output should be printed as “Hello World”.**

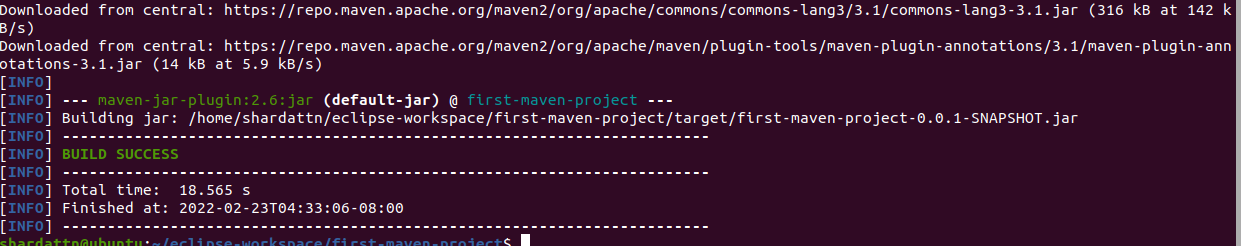
Let’s edit our pom.xml to add JAR plugin and use it.



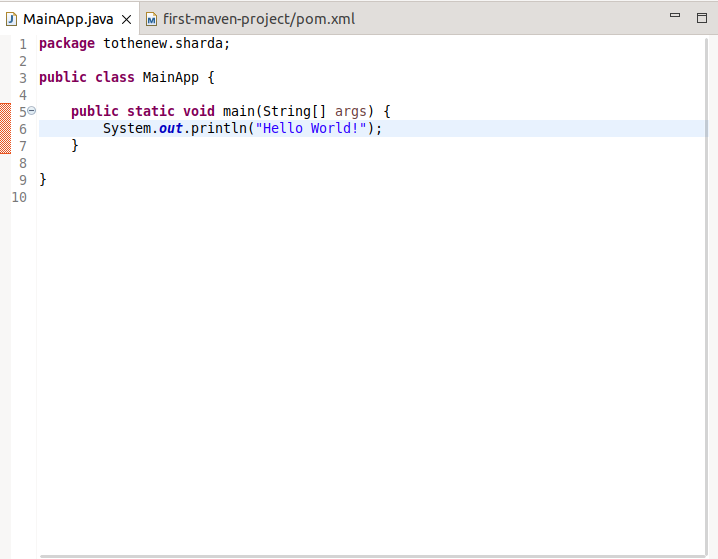
Added the above code.

Executed: mvn package on terminal

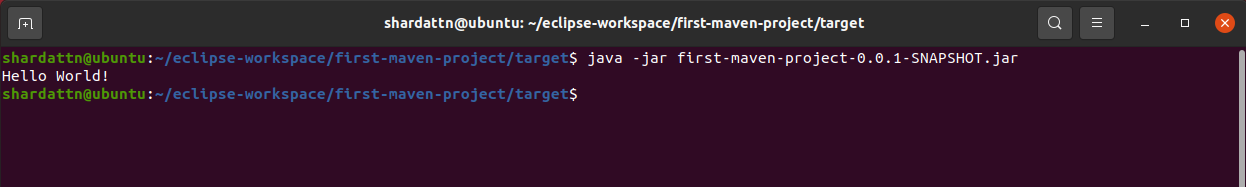




Also, I added some code in source folder.



Let’s try to run



**4. Differentiate between the different dependency scopes: compile, runtime, test., provide using different dependencies being defined in your pom.xml.**

I added few dependencies with different scopes.



Now,

The default scope for every dependency is ***compile***. It makes the dependency to use for everything.

***test*** scope makes the dependency available at only compiling and running the test units.

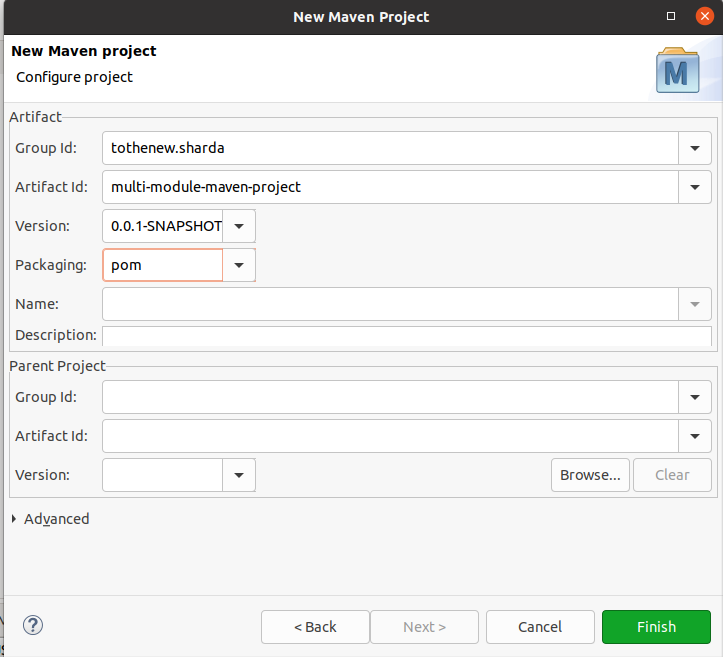
***runtime*** when a dependency is marked as with runtime scope, it will not be available at compilation. It is useful when using APIs.

**5. Create a multi-module project. Run package command at the top level to make jar of every module.**

Let’s make a multi-module maven project in eclipse.

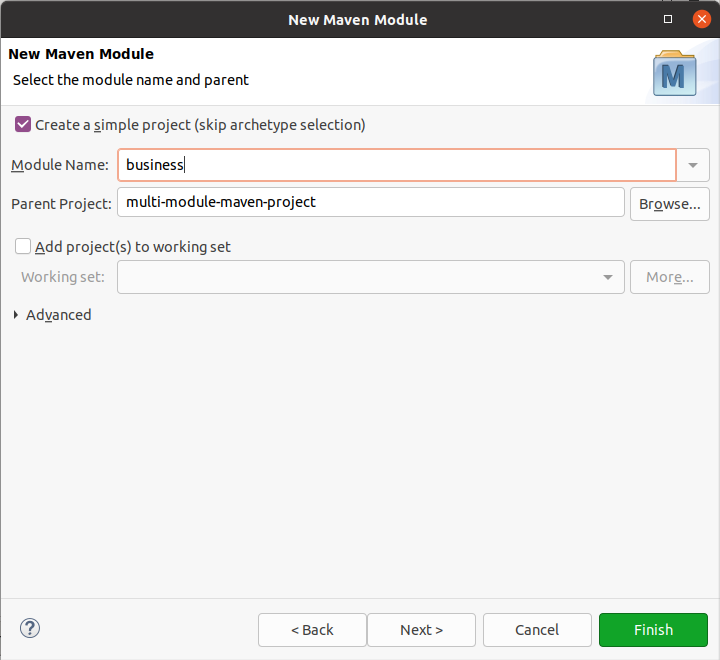
First of all, create a maven project in eclipse-

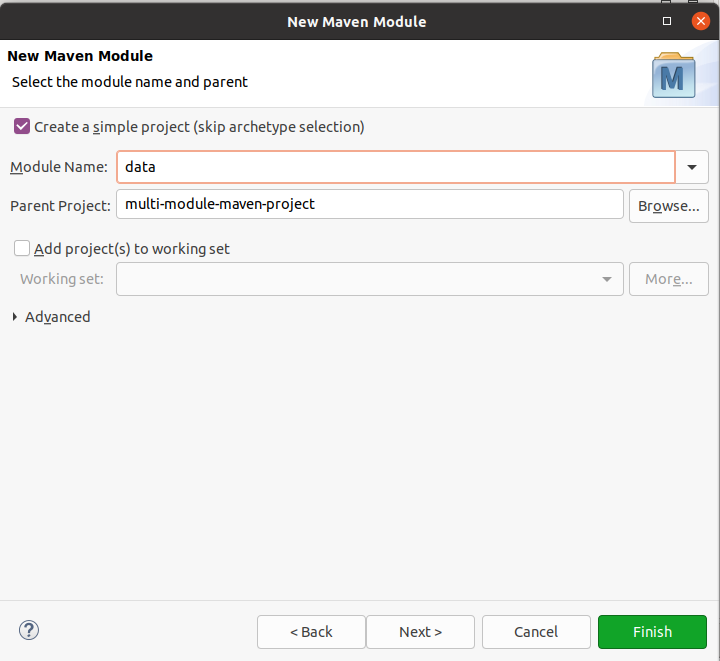
As one should be the parent project we will be using pom for it’s packaging.



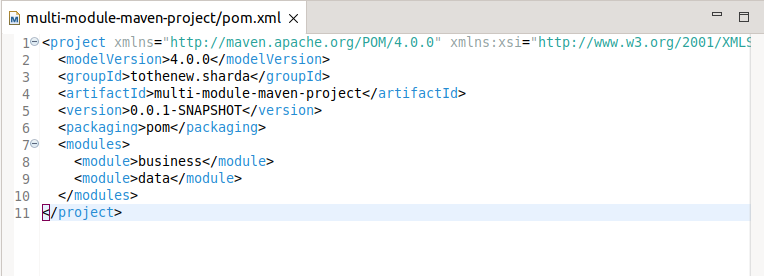
Now let’s create two modules for this project-

I am adding two modules ***business & data.***



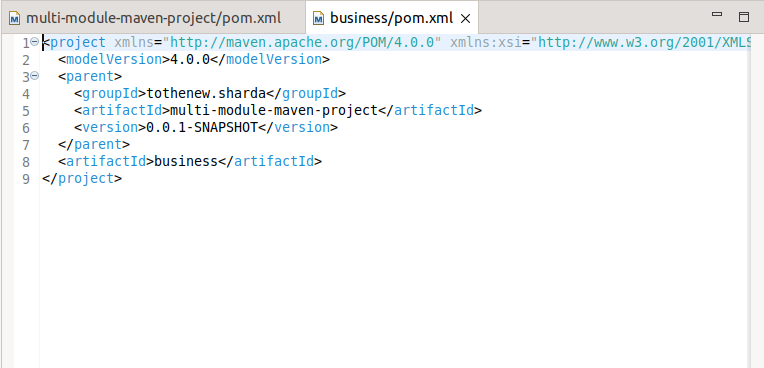


Now let’s take a look at the main project’s pom.xml-

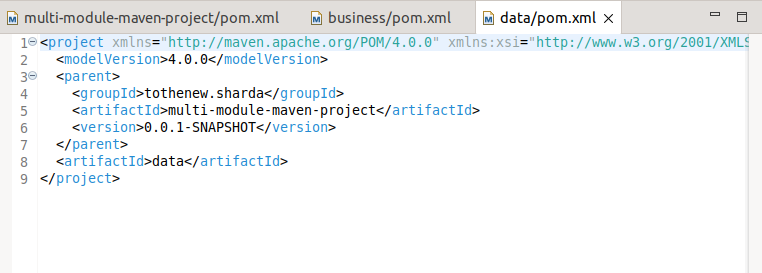


Let’s take a look at both module’s pom.xml files-

***business*** pom.xml-

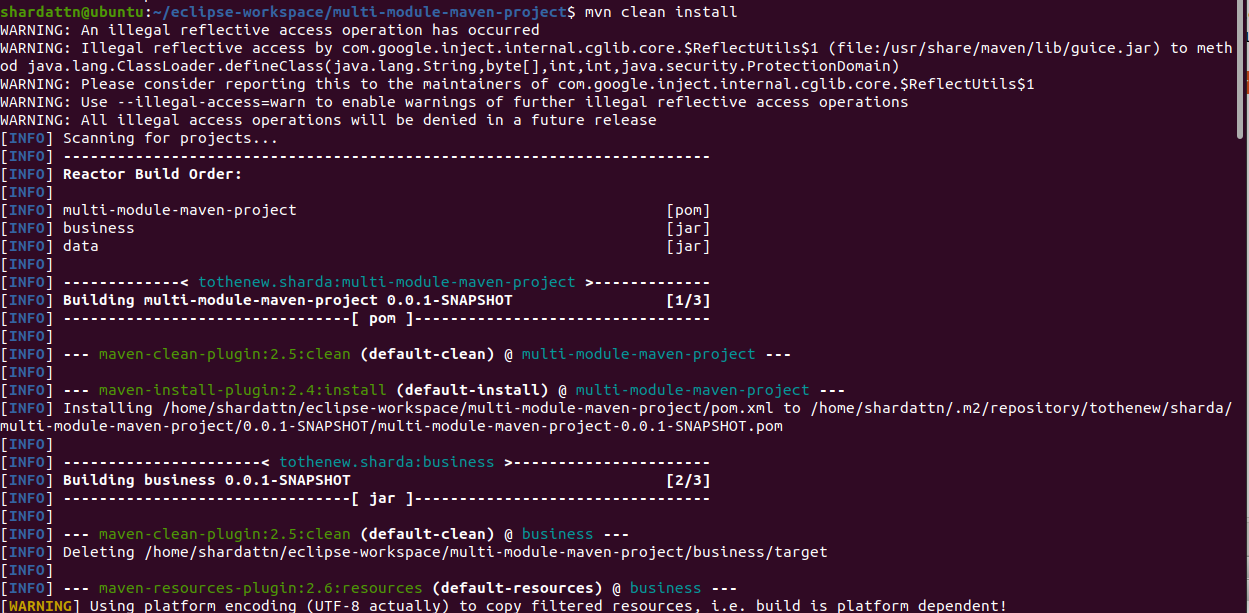


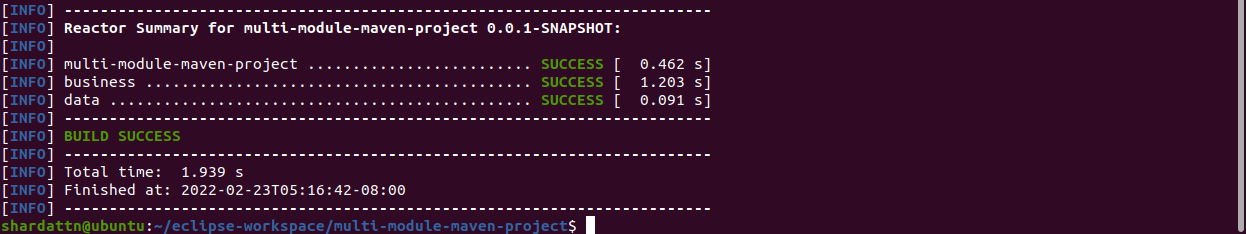
***data*** pom.xml-



Let’s test if this multi-modularity is even working, by typing below command in terminal-

***mvn clean install***



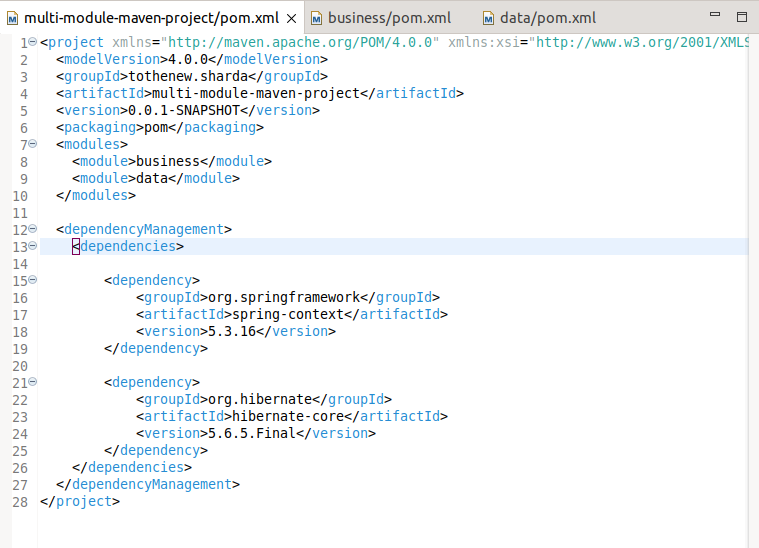


It is working 😊.

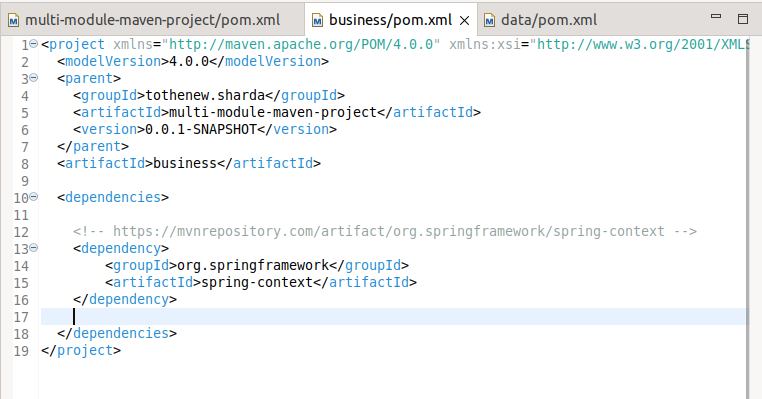
Let’s add some dependencies and do dependency management-

Let’s take a look at updated pom.xml of the whole project and both modules-

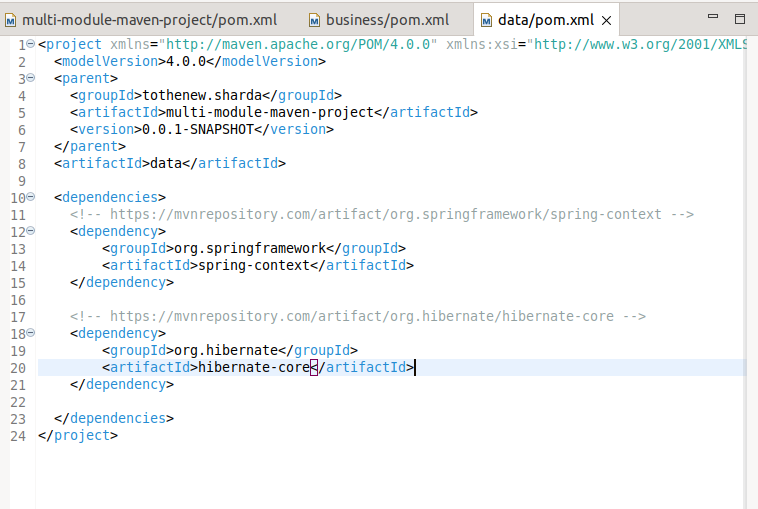
***parent*** pom.xml-



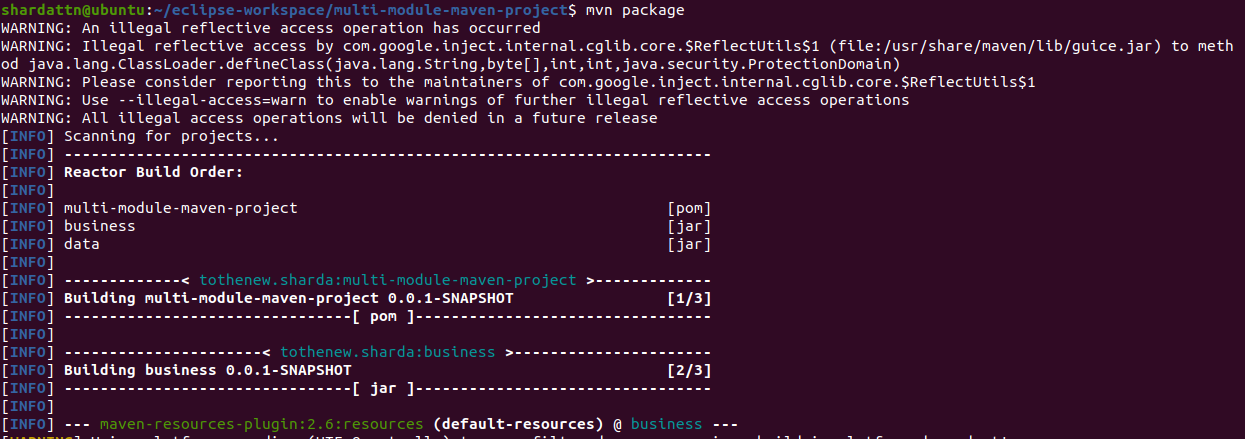
***business*** pom.xml-

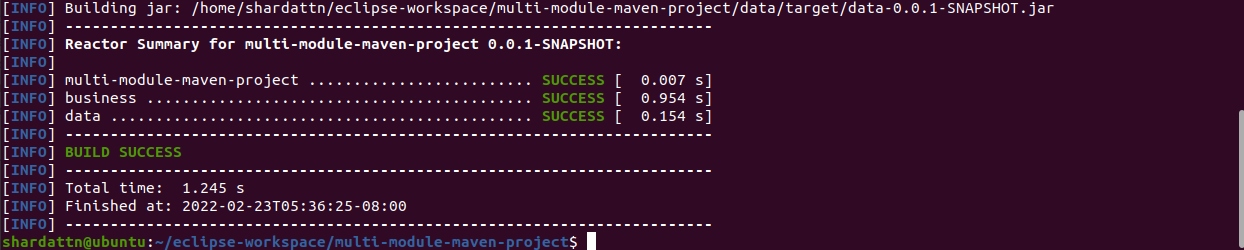


***data*** pom.xml-



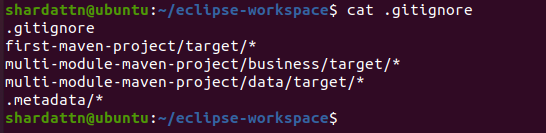
Let’s try to package it-





And it’s done.

Let’s create a .gitignore file and enter the below data-



Also I am uploading this file in the repository too just so you can get a good view of all the work process.

Thanks 😊.