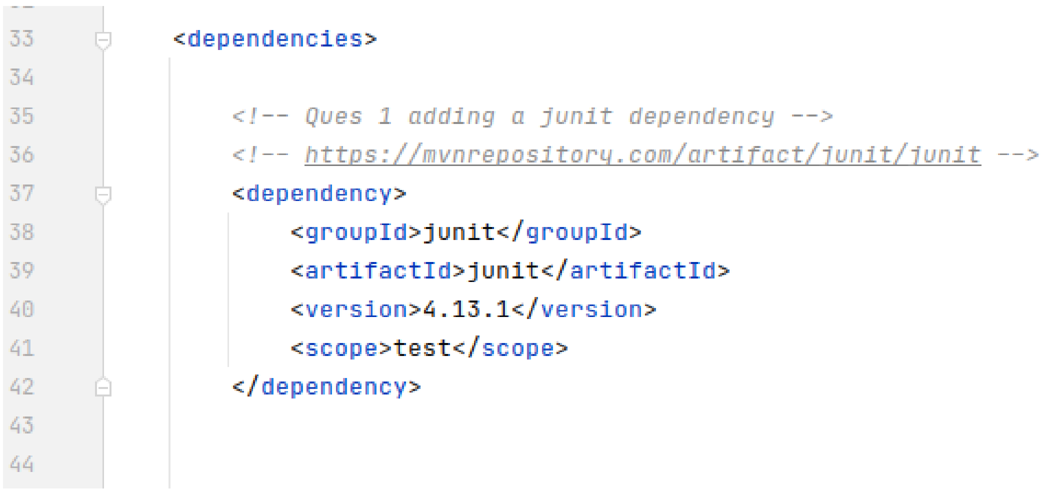
# Session: Dependency Management using Maven

# Assignment

- All the code file and hole project files are uploaded to GitHub
  Link: https://github.com/Parth-C8/Maven-Projects

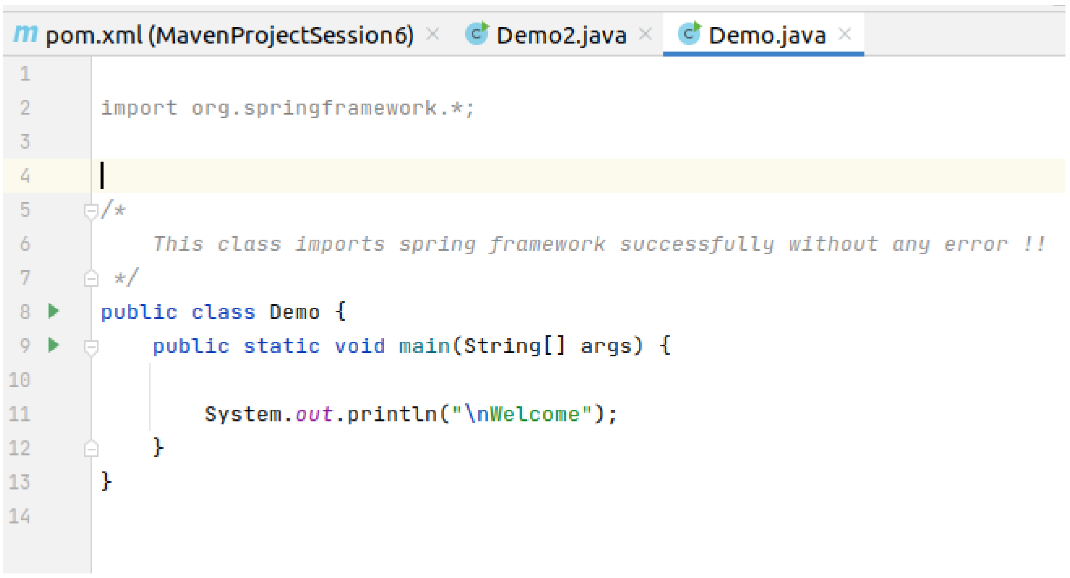
1. Add a maven dependency and its related repository URL.



```xml
    <dependencies>

        <!-- Ques 1 adding a junit dependency -->
        <!-- https://mvnrepository.com/artifact/junit/junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.1</version>
            <scope>test</scope>
        </dependency>

```

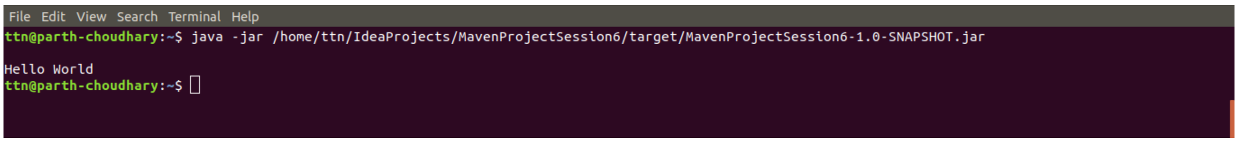
2. Add a new repository in the pom.xml and use its dependencies.

   - Dependency added and imported to a Demo.java class successfully



```java

import org.springframework.*;


/*
    This class imports spring framework successfully without any error !!
 */
public class Demo {
    public static void main(String[] args) {

        System.out.println("\nWelcome");
    }
}

```

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"



```
ttn@parth-choudhary:~$ java -jar /home/ttn/IdeaProjects/MavenProjectSession6/target/MavenProjectSession6-1.0-SNAPSHOT.jar

Hello World
ttn@parth-choudhary:~$ 
```

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

   **Dependency scopes** can help to limit transitivity of the dependencies and they modify classpath for different built tasks.

   - **Compile:** This is the default scope when no other scope is provided. Dependencies with this scope are available on the classpath of the project in all build tasks and they're propagated to the dependent projects.

     ```
     <dependency>
        <groupId>commons-lang</groupId>
        <artifactId>commons-lang</artifactId>
        <version>2.6</version>
     </dependency>
     ```

   - **Runtime:** The dependencies with this scope are required at runtime, but they're not needed for compilation of the project code.

     ```
     <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>6.0.6</version>
        <scope>runtime</scope>
     </dependency>
     ```

   - **Test:** This scope is used to indicate that dependency isn't required at standard runtime of the application, but is used only for test purposes.

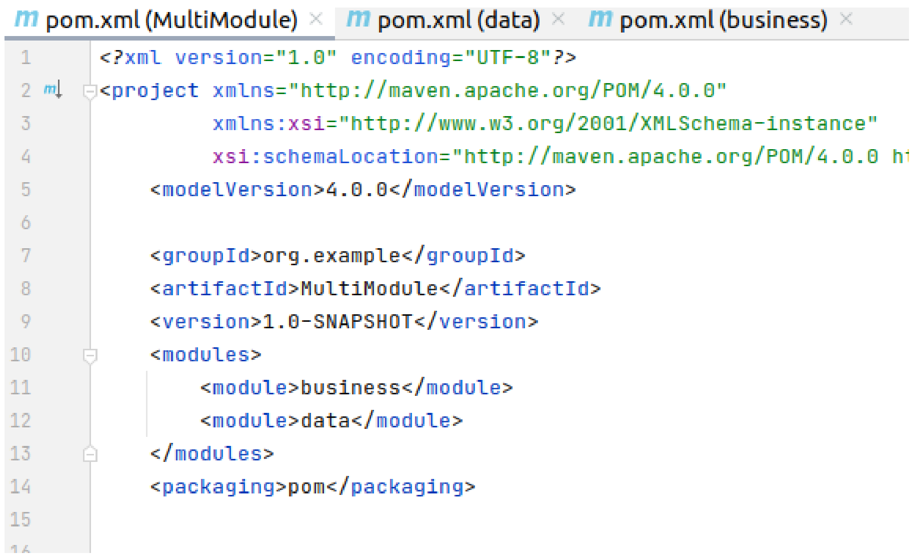
     ```
     <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
     </dependency>
     ```

- **Provided:** This scope is used to mark dependencies that should be provided at runtime by JDK or a container. For example, a web server that already provides the Servlet API at runtime, thus those dependencies can be defined with the provided scope.

```
<dependency>
   <groupId>javax.servlet</groupId>
   <artifactId>servlet-api</artifactId>
   <version>2.5</version>
   <scope>provided</scope>
</dependency>
```

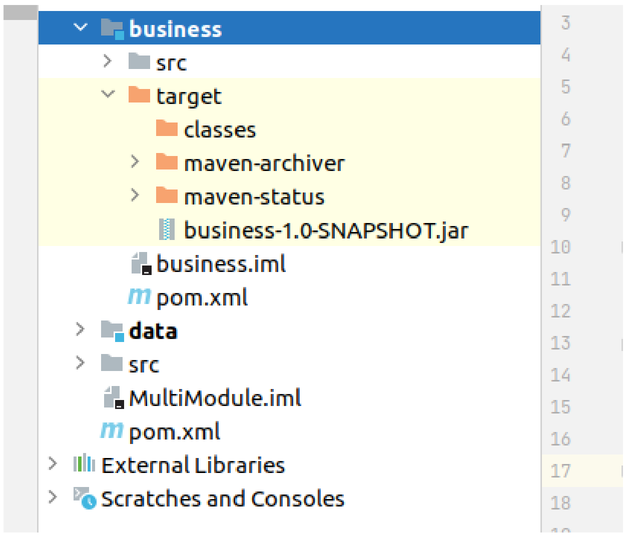
5. Create a multi-module project. Run package command at the top level to make jar of every module.

   - Two modules with the name as data and business are added in the parent module and jar file for both of them is created using the package command.



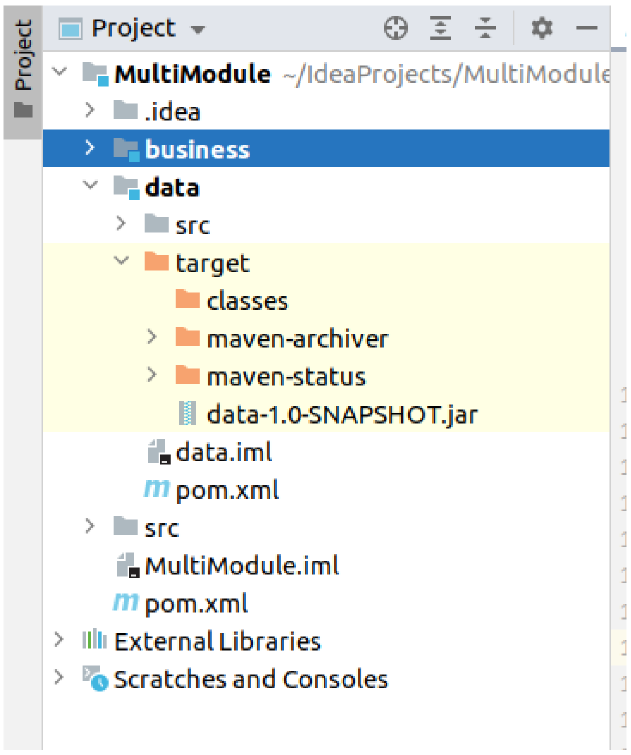
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 ht
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>MultiModule</artifactId>
    <version>1.0-SNAPSHOT</version>
    <modules>
        <module>business</module>
        <module>data</module>
    </modules>
    <packaging>pom</packaging>

```

- Business jar



```
business
  src
  target
    classes
    maven-archiver
    maven-status
    business-1.0-SNAPSHOT.jar
  business.iml
  pom.xml
data
src
MultiModule.iml
pom.xml
External Libraries
Scratches and Consoles
```

- Data jar



```
Project
MultiModule ~/IdeaProjects/MultiModule
  .idea
  business
  data
    src
    target
      classes
      maven-archiver
      maven-status
      data-1.0-SNAPSHOT.jar
    data.iml
    pom.xml
  src
  MultiModule.iml
  pom.xml
External Libraries
Scratches and Consoles
```