Maven—Project Management Tool:

Maven is a build automation tool developed using the Java programming language. It is primarily used for Java-based projects to manage the build process, including source code compilation, testing, packaging, and more. Maven utilizes the Project Object Model (POM), where the pom.xml file describes the project's configuration and dependency management.

Features of Maven:

Dependency Management: Maven simplifies the process of including third external libraries in our project and It can download automatically required libraries.

Standardized Project Structure: It provides a standard project folder structure like source code in one package, testing code in one package, and others.

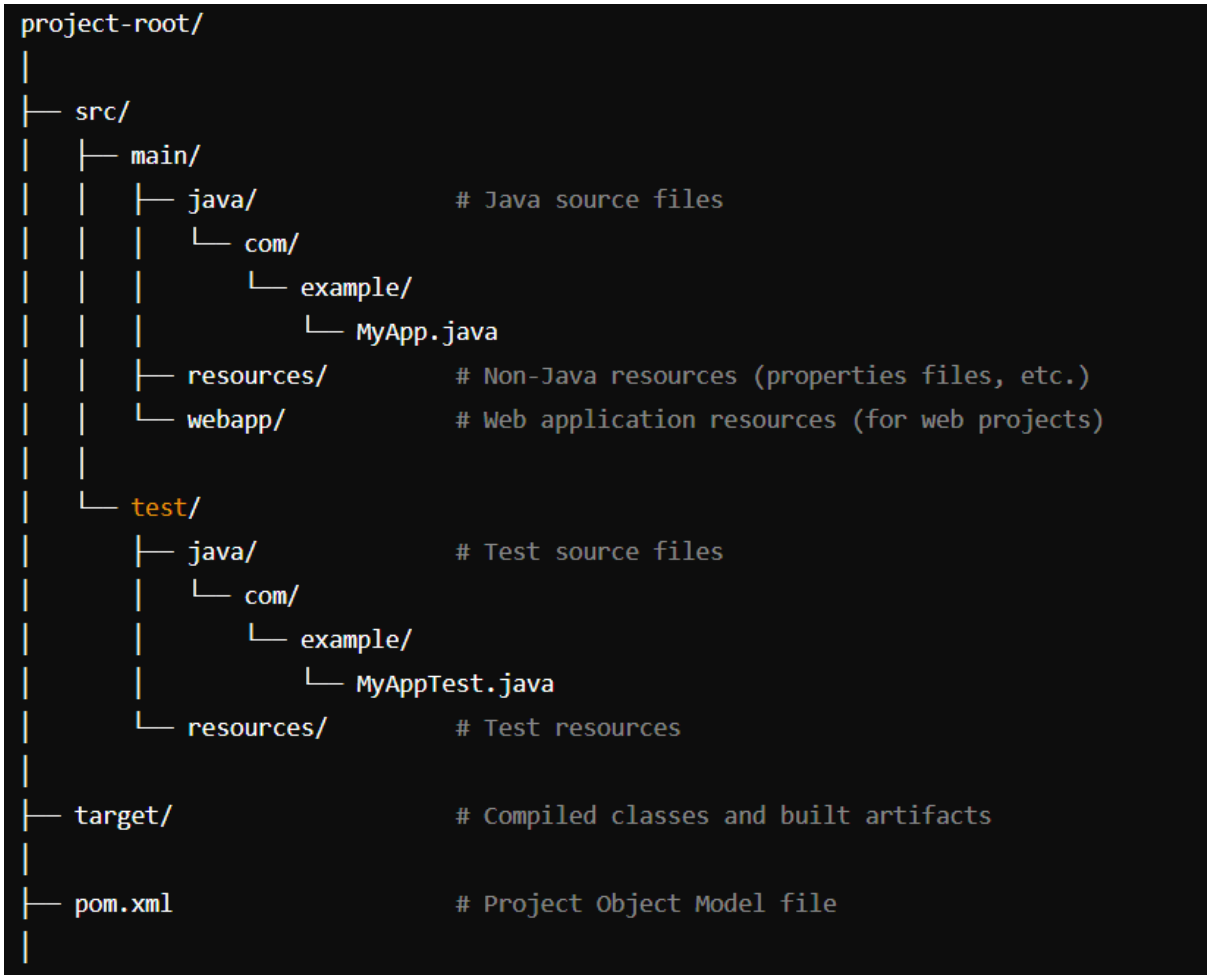
Plugins: It supports different plugins these are perform compilation, testing, packaging, deployment, documentation generation, and others.

Project Object Model: Every maven Project have parent pom.xml file which can decided the configuration and dependency management of the maven project.

Central Repository: Maven uses a central repository for downloading project dependencies, which simplifies dependency management.

IDE Integration: It support different IDEs like STS, Eclipse, IntelliJ IDEA, and NetBeans.

Maven Project Structure:



src/main/java: It contains the main Java source code.

src/main/resources: It contains non-Java resources used by the application.

src/main/webapp: It contains resources for web applications.

src/test/java: It contains test source code.

src/test/resources: It contains resources used for testing.

target: It contains compiled classes, packaged JARs/WARs, and other built artifacts.

pom.xml: The Project Object Model file that defines the project configuration, dependencies, and build settings.

Jar-Java Archive Files

Rather than importing all the libraries we can use the Maven and provide the dependencies int the pom.xml then the Jar files will be downloaded and let's the Project run.

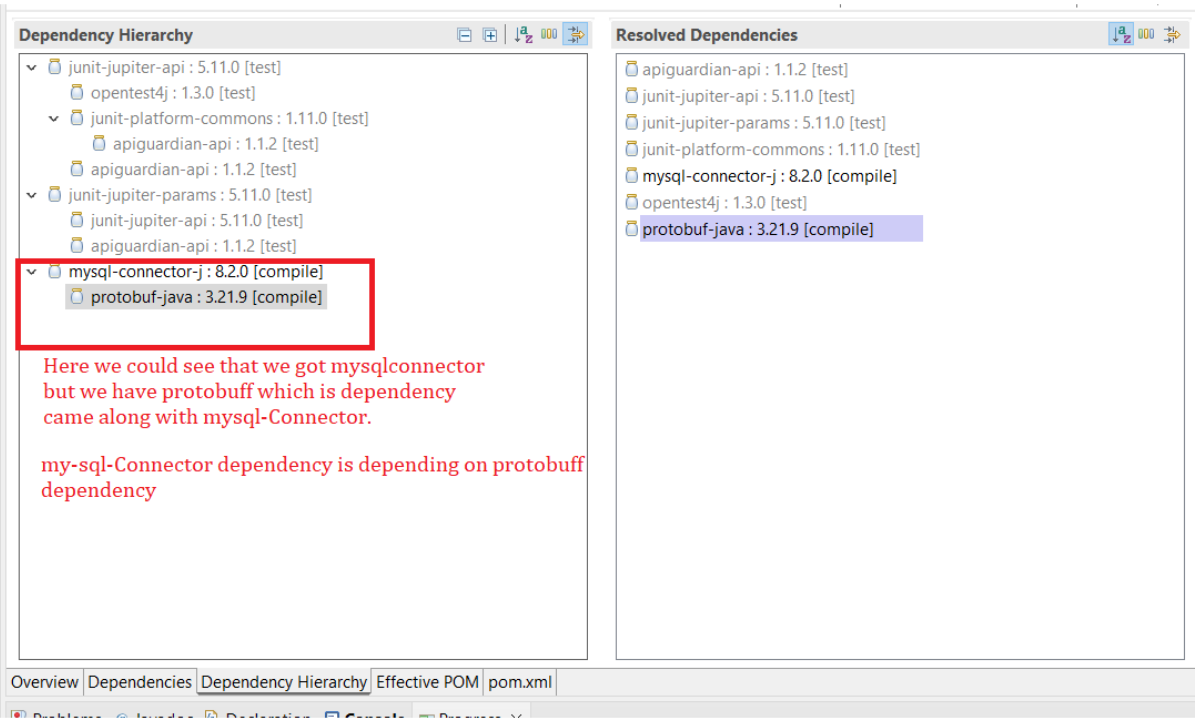
Normally once we are write these steps has to be followed.

Some times we need to use External Libraries(JAR).We can download them and use it.

Ex: when we wanted to connect with Database corresponding libraries are needed.

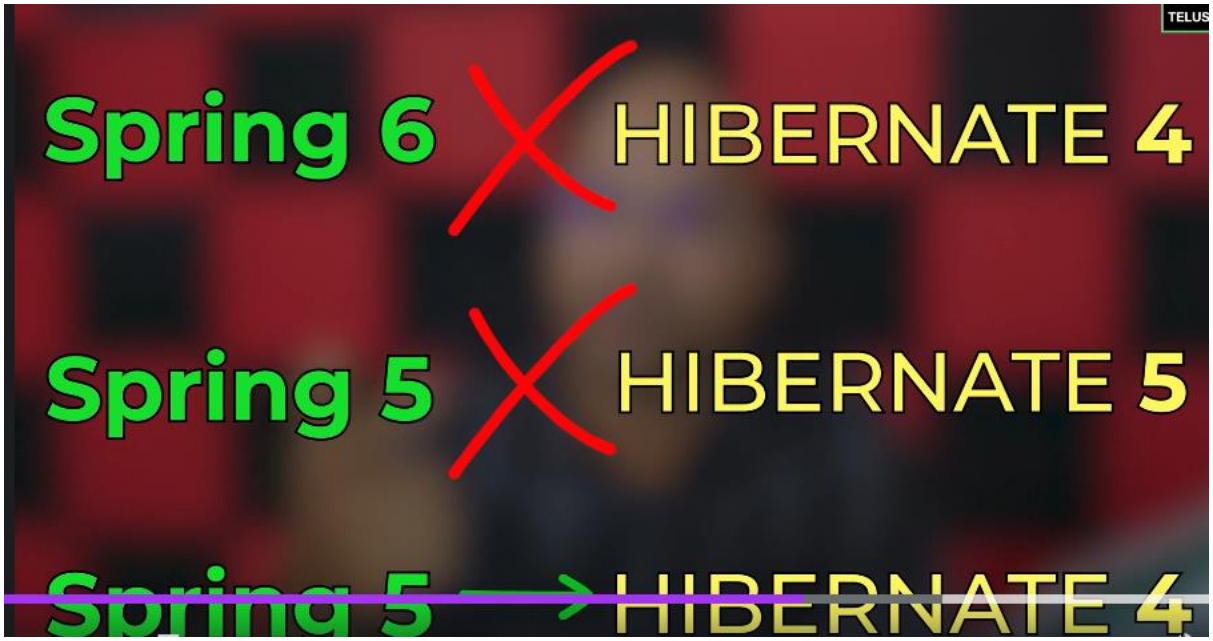
Transitive Dependencies:

One dependency depending on some other dependencies.



This is Example of Transitive Dependencies.

Ex: Hibernate is dependent on Some other dependencies.

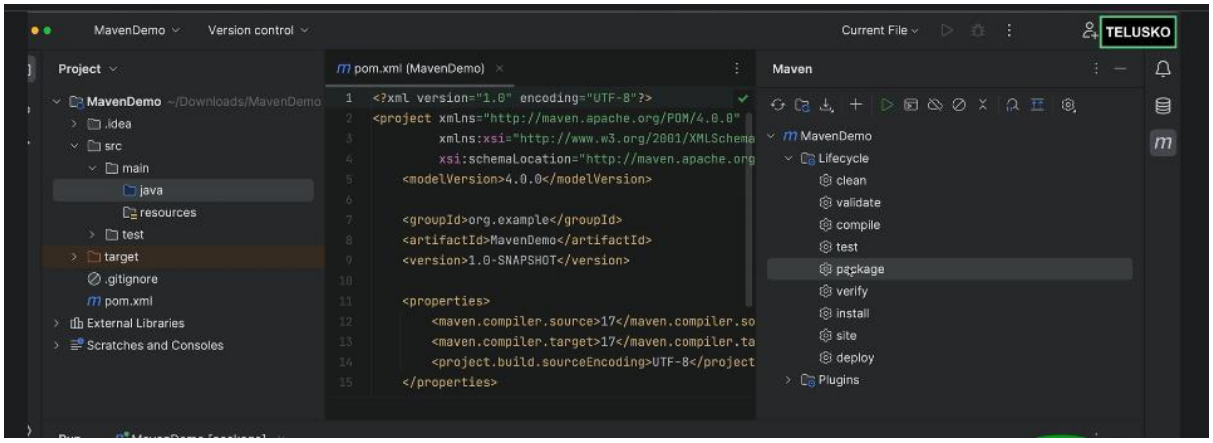


Let suppose we need to share our Project then if we are not using dependencies then we other team also have download and import those jars.

Maven helps us to get the Dependencies.

Once we ask Maven it will provide the dependencies.

Maven also provides the plugins for life Cycle(Compile—Run---Test—Packaging—Deploying)



Plugins provided by Maven for Life cycle. Once we click on the corresponding action will be performed.

Maven provides something called ArcheType—Template tool. We can use those ArcheType .

Else we can create our own ArcheTypes .(If Project needs Specific ArcheType we can create that ArcheType and other can also use same for developing the Other projects).

Pom—Project Object Model—Where we do everything to handle Maven

Every Library will have three Things(GAV)

Group id: It has to be Unique

Artifact Id : Project Name

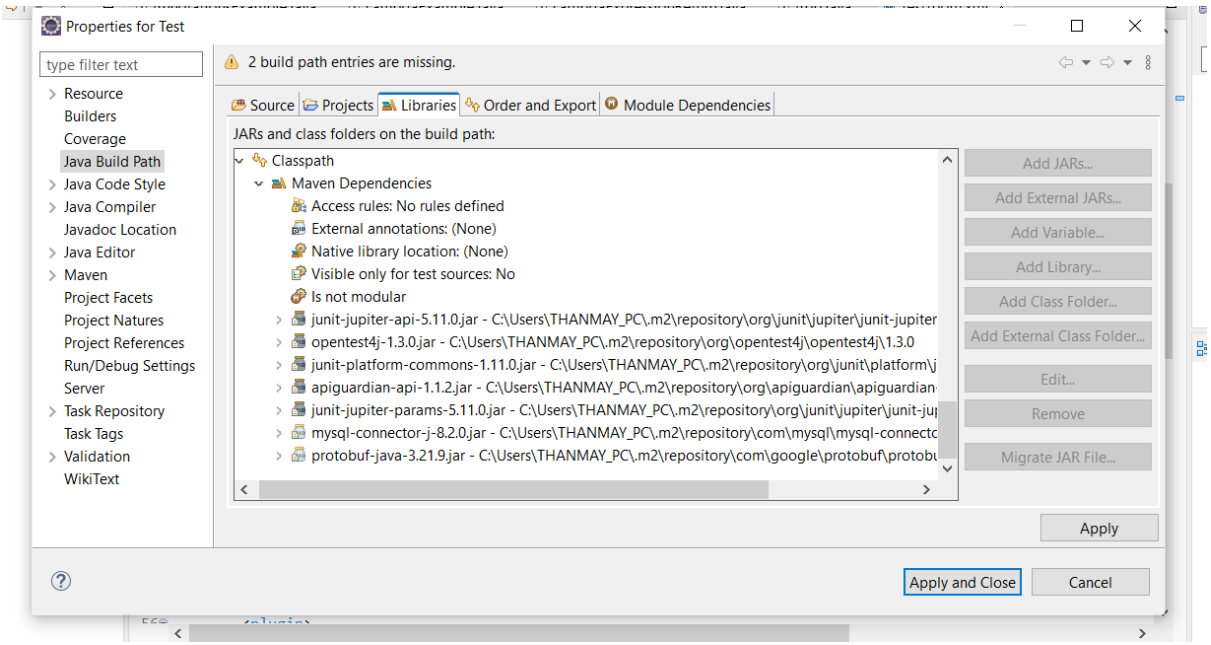
Version:

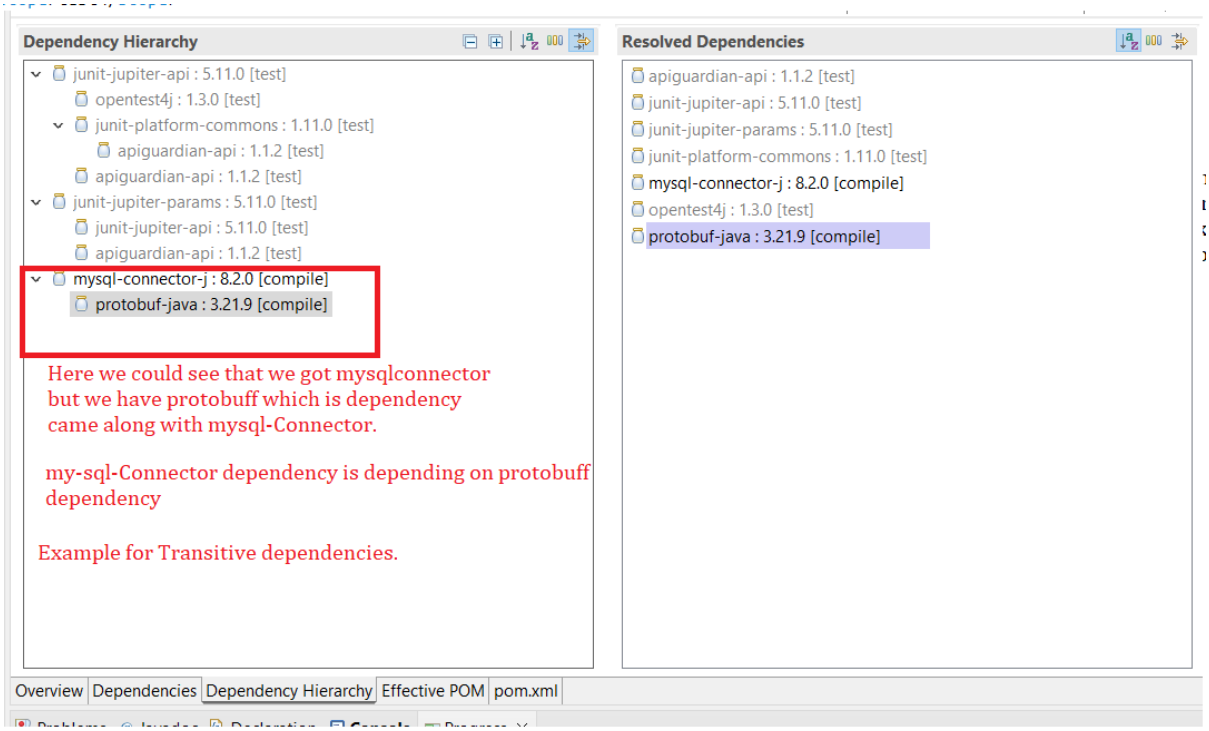
We might have multiple dependencies we can keep those dependencies in the Dependencies tag.

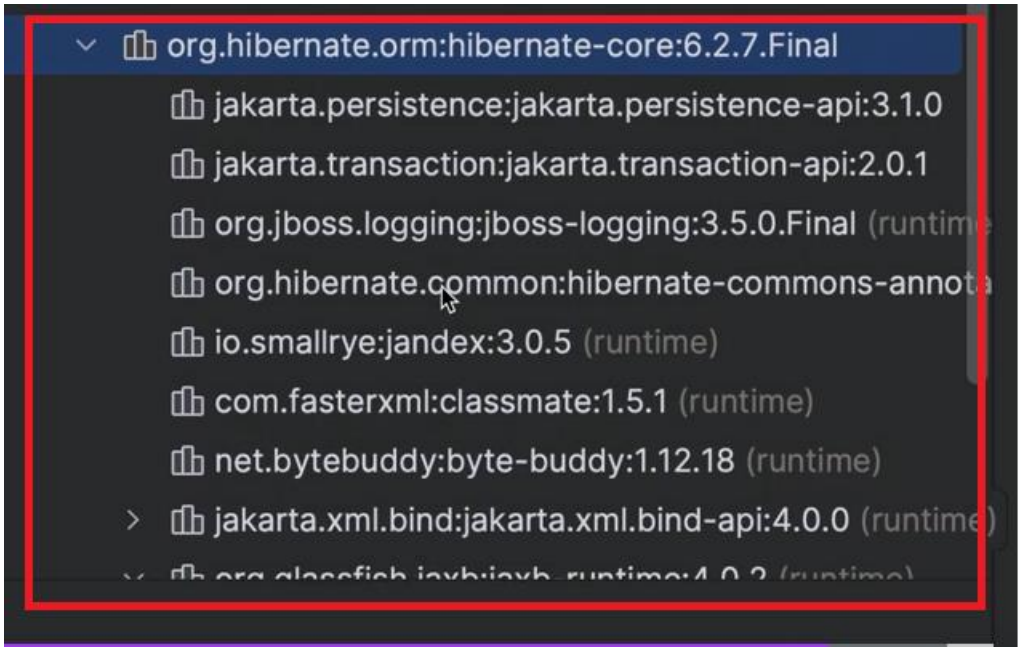
```
<dependencies>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <scope>test</scope>
  </dependency>
  <!-- Optionally: parameterized tests support -->
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-params</artifactId>
    <scope>test</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/com.mysql/mysql-connector-j -->
 <dependency>
  <groupId>com.mysql</groupId>
  <artifactId>mysql-connector-j</artifactId>
  <version>8.2.0</version>
 </dependency>
</dependencies>
```

This is the dependency which we have taken and maven repository and kept so due to this once we save mysql connector will be downloaded from the internet and will be kept in the Project.Attached the image for reference after importing

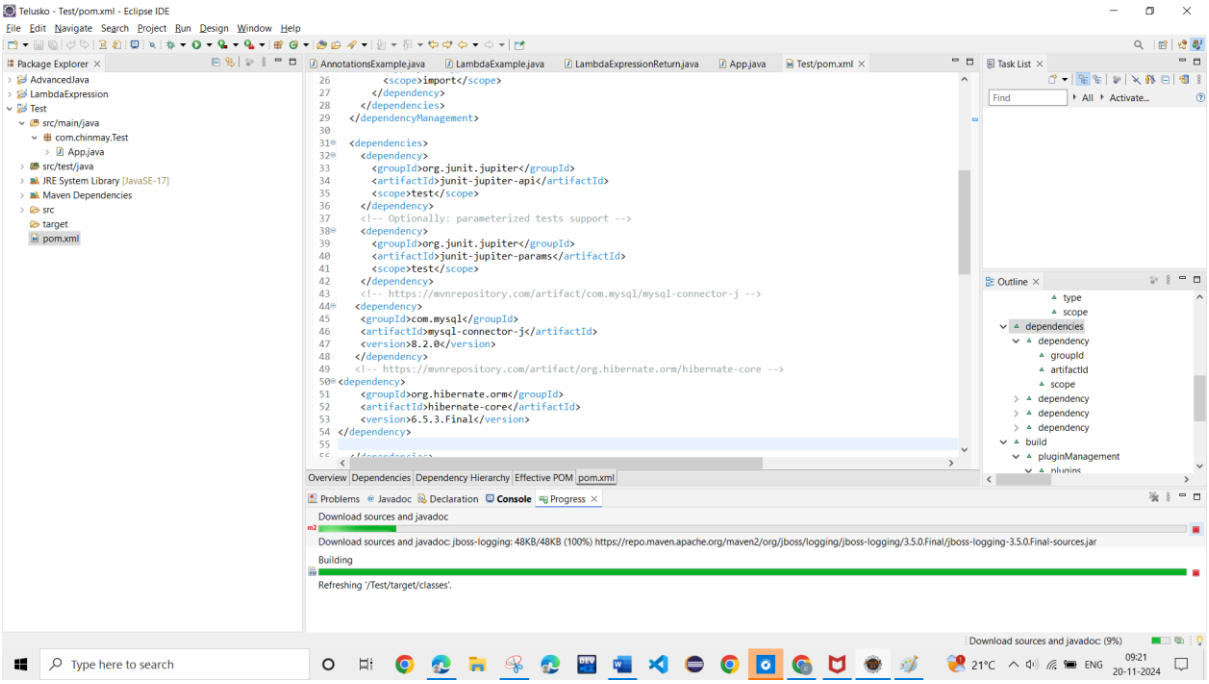
Corresponding library or Jar file will be kept in the folder.







We have added hibernate dependency but we could see from the above image many mode dependencies because hibernate is dependent on those dependencies.



As we could see from the above image after adding dependency to the pom.xml and save it is downloading the related jar files

If we share the project to some other developer, we doesn't have to share the jar files along with the project. As once they import and load the Project jar files will get downloaded automatically.

POM File :

https://www.baeldung.com/maven-super-simplest-effective-pom

Super POM :

To understand super POM more easily, we can make an analogy with the Object class from Java: Every class from Java extends, by default, the Object class. Similarly, in the case of POM, every POM extends the super POM.

The super POM file defines all the default configurations. Hence, even the simplest form of a POM file will inherit all the configurations defined in the super POM file.

Simplest POM:

The simplest POM is the POM that you declare in your Maven project. In order to declare a POM, you will need to specify at least these four elements:

modelVersion, groupId, artifactId, and version. The simplest POM will inherit all the configurations from the super POM.

Effective POM:

Effective POM combines all the default settings from the super POM file and the configuration defined in our application POM. Maven uses default values for configuration elements when they are not overridden in the application pom.xml. Hence, if we take the same sample POM file from the simplest POM section, we'll see that the effective POM file will be the merge between simplest and super POM.

Maven ArcheType:

Templates are basically called ArcheTypes.

We can basically choose the ArcheTypes from the available one's.

We can create a new ArcheType as well.

How Maven Works:

Once we add the new dependencies Maven will search for those dependencies in the local system like C:\Users\THANMAY_PC\.m2 (Users/.m2/repositories) folder where we will get the details of the downloaded library files if present.

If the file is not present then it will take from the Maven central repository and updated into the Project.

If Version is vulnerable we need to update the version.