**What is Maven?**

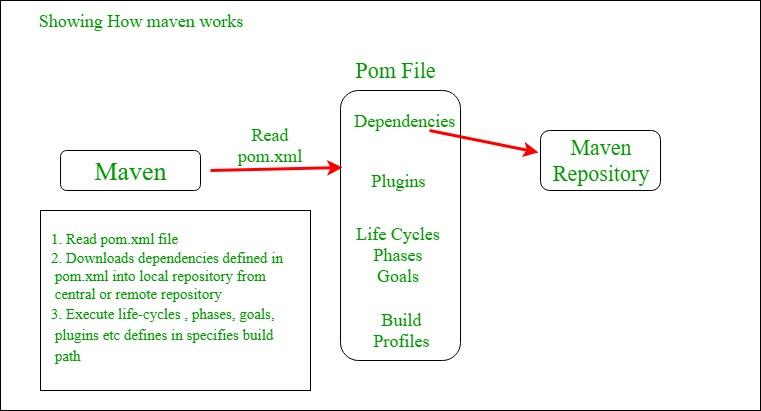
Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation. It simplifies the build process like ANT. But it is too much advanced than ANT.  
In short terms we can tell maven is a tool that can be used for building and managing any Java-based project. maven make the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.

**What maven does?**

Maven does a lot of helpful task like

1. We can easily build a project using maven.
2. We can add jars and other dependencies of the project easily using the help of maven.
3. Maven provides project information (log document, dependency list, unit test reports etc.)
4. Maven is very helpful for a project while updating central repository of JARs and other dependencies.
5. With the help of Maven we can build any number of projects into output types like the JAR, WAR etc without doing any scripting.
6. Using maven we can easily integrate our project with source control system (such as Subversion or Git).

**How maven works?**



The sample of pom.xml

| <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>  <http://maven.apache.org/xsd/maven-4.0.0.xsd>">             <modelVersion>4.0.0</modelVersion>           <groupId> com.project.loggerapi </groupId>           <artifactId>LoggerApi</artifactId>           <version>0.0.1-SNAPSHOT</version>           <!-- Add typical dependencies for a web application -->         <dependencies>                 <dependency>                         <groupId>org.apache.logging.log4j</groupId>                         <artifactId>log4j-api</artifactId>                         <version>2.11.0</version>                   </dependency>         </dependencies>       </project> |
| --- |

Elements used for Creating pom.xml file

1. **project-**It is the root element of the pom.xml file.
2. **modelVersion-**modelversion means what version of the POM model you are using. Use version 4.0.0 for maven 2 and maven 3.
3. **groupId-**groupId means the id for the project group. It is unique and Most often you will use a group ID which is similar to the root Java package name of the project like we used the groupId com.project.loggerapi.
4. **artifactId-**artifactId used to give name of the project you are building.in our example name of our project is LoggerApi.
5. **version-**version element contains the version number of the project. If your project has been released in different versions then it is useful to give version of your project.

Other Elements of Pom.xml file

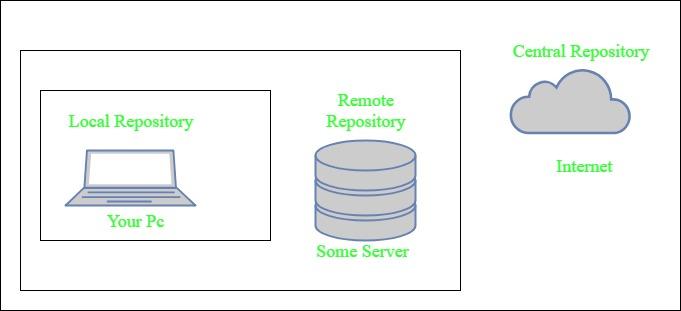
1. **dependencies-**dependencies element is used to defines a list of dependency of project.
2. **dependency-**dependency defines a dependency and used inside dependencies tag. Each dependency is described by its groupId, artifactId and version.
3. **name-**this element is used to give name to our maven project.
4. **scope-**this element used to define scope for this maven project that can be compile, runtime, test, provided system etc.
5. **packaging-**packaging element is used to packaging our project to output types like JAR, WAR etc.

**Maven Repository**

Maven repositories are directories of packaged JAR files with some metadata. The metadata are POM files related to the projects each packaged JAR file belongs to, including what external dependencies each packaged JAR has. This metadata enables Maven to download dependencies of your dependencies recursively until all dependencies are download and put into your local machine.

Maven has three types of repository :

1. **Local repository**
2. **Central repository**
3. **Remote repository**

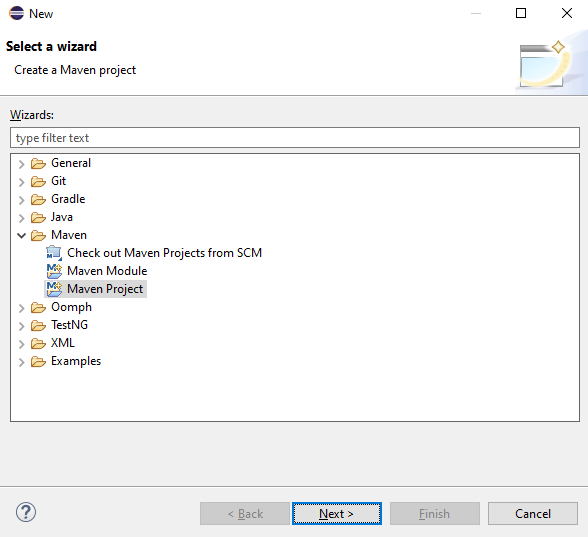
Maven searches for dependencies in this repositories. First maven searches in Local repository then Central repository then Remote repository if Remote repository specified in the POM.  


1. **Local repository-** A local repository is a directory on the machine of developer. This repository contains all the dependencies Maven downloads / tester. Maven only needs to download the dependencies once, even if multiple projects depends on them (e.g. ODBC).  
   By default, maven local repository is user\_home/m2 directory.  
   example – **C:\Users\ashok\.m2**
2. **Central repository-** The central Maven repository is created Maven community. Maven looks in this central repository for any dependencies needed but not found in your local repository. Maven then downloads these dependencies into your local repository.
3. **Remote repository-** remote repository is a repository on a web server from which Maven can download dependencies.it often used for hosting projects internal to organization. Maven then downloads these dependencies into your local repository.

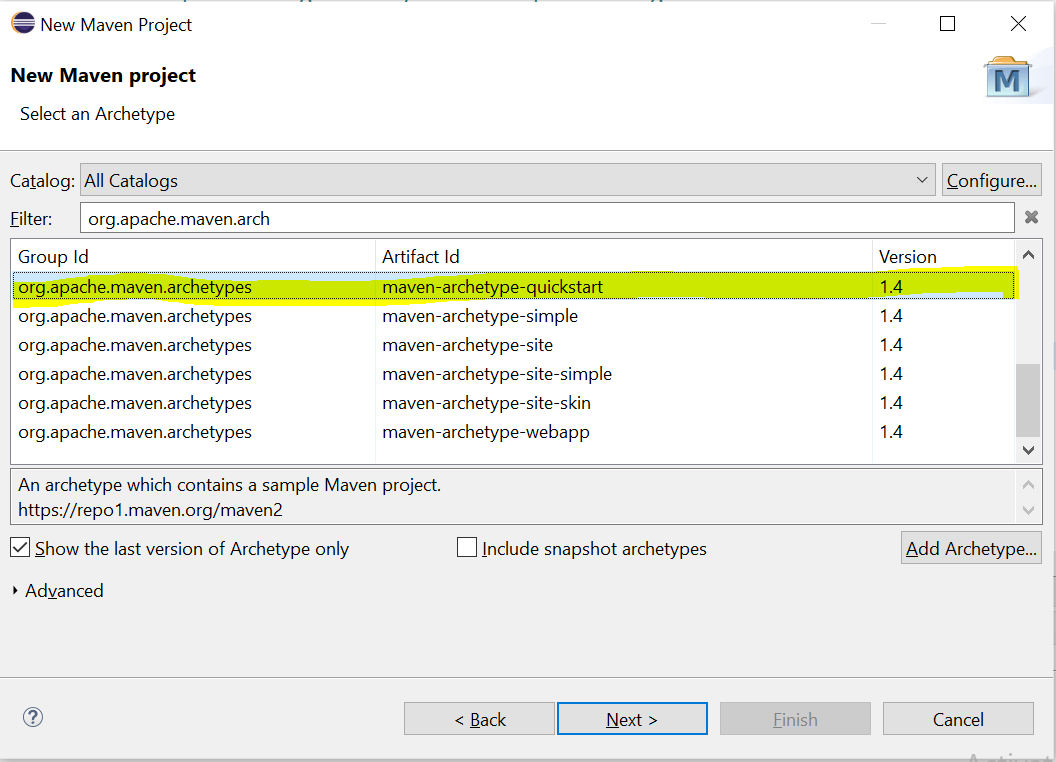
**Steps to create Maven Project:**

**Step 1)** In Eclipse IDE, create a new project by selecting **File** | **New**

**Step 2)** On the **New** dialog, select **Maven** | **Maven Project** and click Next

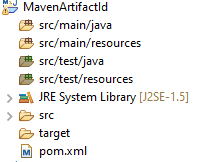


**Step 3)**On the **New Maven Project** dialog click Next

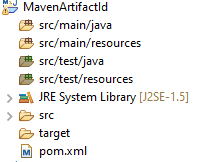
**Step 4) Click on Add archtype** 

**Step 5)**Enter in **Group Id**: and **Artifact Id**: and click finish

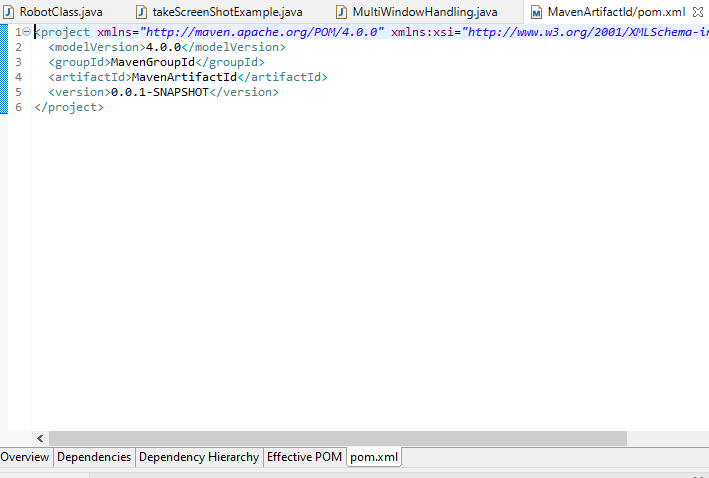
**Step 6)** Eclipse created project **MavenGroupId** with following structure:



**Step 7)**. Select **pom.xml** from **Project Explorer**..



pom.xml file will Open in Editor section



Step 8). Add dependicy to POM.xml file like below

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

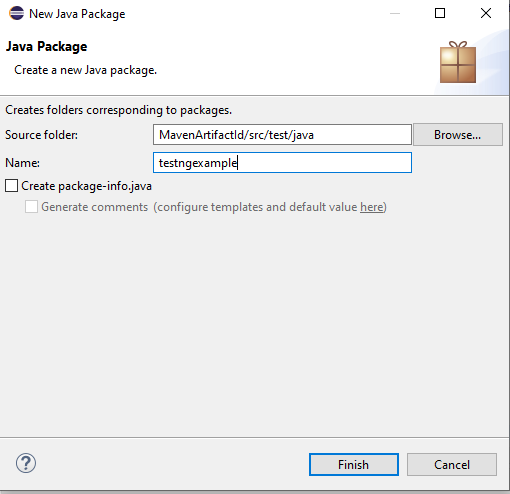
<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

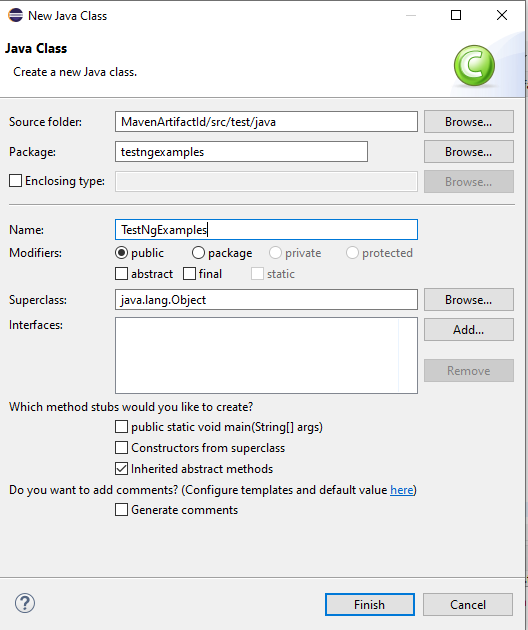
<version>3.141.5</version>

</dependency>

**Step 9)**. Now create a package with as shown below under src/test/java



**Step 10)**. Create new class as shown below under above package



**Step 11)**. Add below code

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebDriverException;

**import** org.openqa.selenium.chrome.ChromeDriver;

**public** **class** TestNgExample {

**private** WebDriver driver;

@Test

**public** **void** testEasy() {

driver.get("https://google.com/");

String title = driver.getTitle();

Assert.assertTrue(title.contains("google"));

}

@BeforeTest

**public** **void** beforeTest() {

driver = **new** ChromeBrowser();

}

@AfterTest

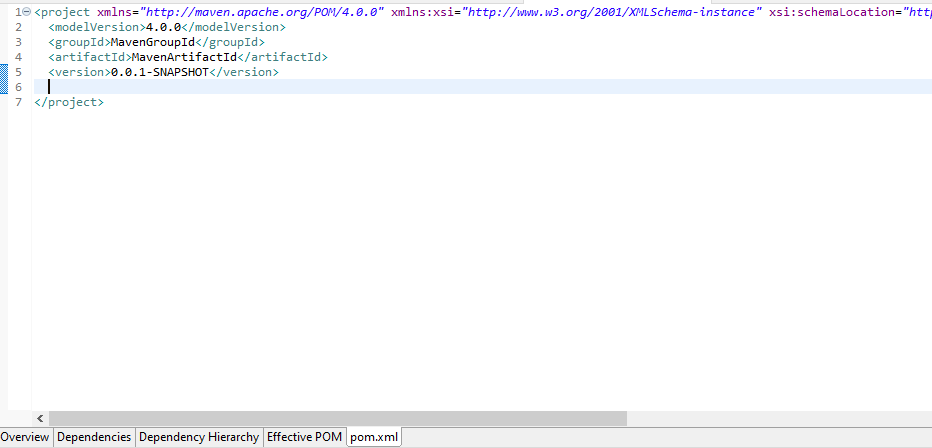
**public** **void** afterTest() {

driver.quit();

}

}

**Step 12)**. Open POM.xml file and see that no Dependencies added in pom.xml file



**Step 15)**. Observe no Dependencies are added

