**Use of Maven Build Automation Tool and Maven Project Setup for Selenium**

In our last Selenium module we learned a [build tool named as “Apache Ant”](http://www.softwaretestinghelp.com/apache-ant-selenium-tutorial-23/). We also broadly discussed its applicability and importance besides the practical approach.

In this [Selenium Testing module](http://www.softwaretestinghelp.com/selenium-tutorial-1/) we will learn **Maven – a build automation tool** which is distributed under Apache Software Foundation. It is mainly used for java projects. It makes build consistent with other project.

Maven is also used to manage the dependencies. For example if you are using selenium version 2.35 and any later point of time you have to use some other version, then same can be managed easily by Maven. You will find more examples of this later in this chapter. It works very effectively when there is huge number of Jar files with different versions.

### ****What is a build tool?****

Build tool is used to setup everything which is required to run your java code independently. This can be applied to your entire java project. It generates source code, compiling code, packaging code to a jar etc. Maven provides a common platform to perform these activities which makes programmer’s life easier while handling huge project.

Maven provides pom.xml which is the core to any project. This is the configuration file where all required information’s are kept. Many of the IDEs (Integrated Development Environments) are available which makes it easy to use. IDEs are available for tools like Eclipse , NetBeans, IntelliJ etc.

Maven stores all project jars. Library jar are in place called repository which could be central, local or remote repository. Maven downloads the dependency jar from central repository. Most of the commonly used libraries are available in<http://repo1.maven.org/maven2/>.

Downloaded libraries are stored in local repository called m2. Maven uses the libraries available in m2 folder and if any new dependency added then maven downloads from central repository to local repository. If libraries are not available in central repository then maven looks for remote repository. User has to configure the remote repository in pom.xml to download from remote repository.

**Below is the example of configuring a remote repository topom.xml file**. Provide id and url of the repository where libraries are stored.

|  |  |
| --- | --- |
| 1 | <repositories> |
| 2 | <repository> | |

|  |  |
| --- | --- |
| 3 | <id>libraryId</id> |
| 4 | <url>http://comanyrepositryId</url> | |

|  |  |  |
| --- | --- | --- |
| 5 | </repository> | |
| 6 | </repositories> |

**General Phrases used in Maven:**

* **groupId**: Generally groupId refers to domain id. For best practices company name is used as groupId. It identifies the project uniquely.
* **artifactId**: It is basically the name of the Jar without version.
* **version**: This tag is used to create a version of the project.
* **Local repository**: Maven downloads all the required dependencies and stores in local repository called m2. More details regarding the same would be shared in the next topic.

### ****Build Life Cycle:****

Basic maven phases are used as below.

* **clean**: deletes all artifacts and targets which are created already.
* **compile**: used to compile the source code of the project.
* **test**: test the compiled code and these tests do not require to be packaged or deployed.
* **package**: package is used to convert your project into a jar or war etc.
* **install**: install the package into local repository for use of other project.

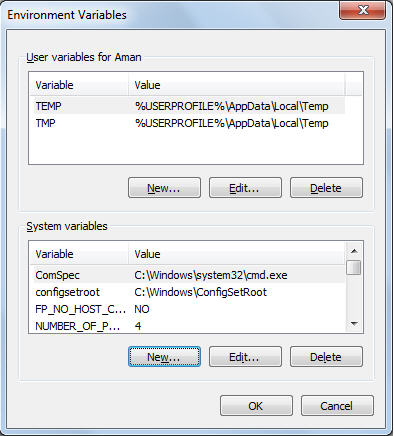
### ****Maven Setup:****

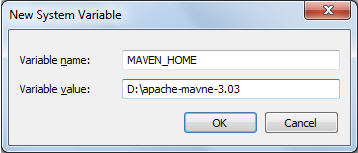
**Step 1**: To setup Maven, download the maven’s latest version form Apache depending upon different OS.

**Step 2**: Unzip the folder and save it on the local disk.

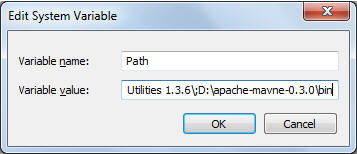
**Step 3**: Create environment variable for MAVEN\_HOME. Follow the below step:

Navigate to System Properties ->Advanced System Setting->Environment Variable ->System Variable ->New ->Add path of Maven folder

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-1.jpg)

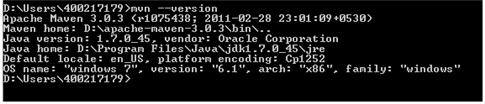
[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-2.jpg)

**Step 4**: Edit path variable and provide the bin folder path.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-3.jpg)

**Step 5**: Now verify the maven installation using command prompt and don’t forget to setup JAVA\_HOME

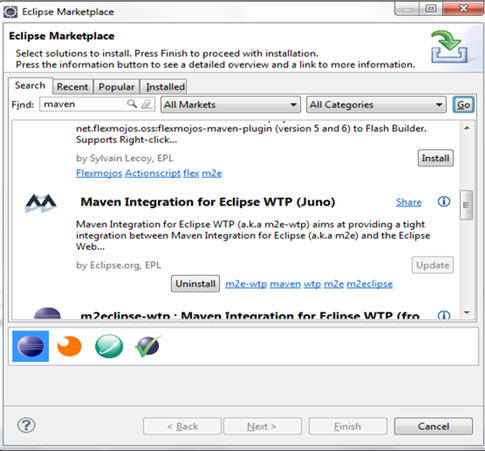
Use mvn –version to verify maven version and output comes like below.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-4.jpg)

### ****Install maven IDE in Eclipse:****

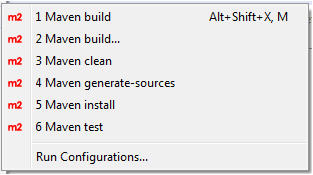
Maven provides IDE to integrate with eclipse. I am using eclipse Juno here.

Navigate to Help->Eclipse Marketplace-> Search maven ->Maven Integration for Eclipse ->INSTALL

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-5.jpg)

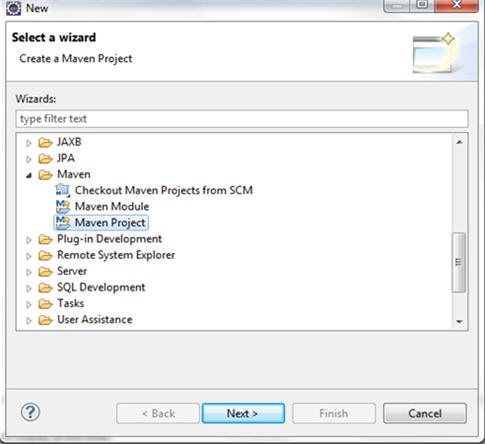
After installation you have to restart eclipse.

Then right click on pom.xml and verify all the options are available like below.

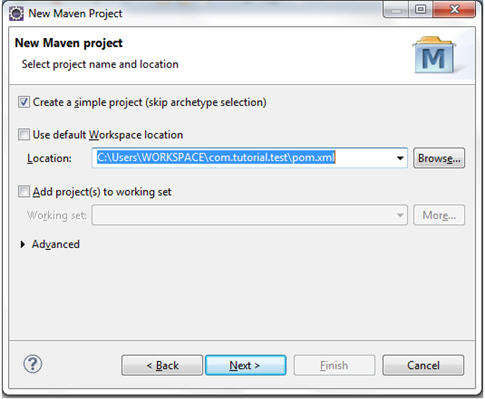
[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-6.jpg)

**Create Maven project:**

**Step 1**: Navigate to File- new-others-Maven-Maven Project-Click Next

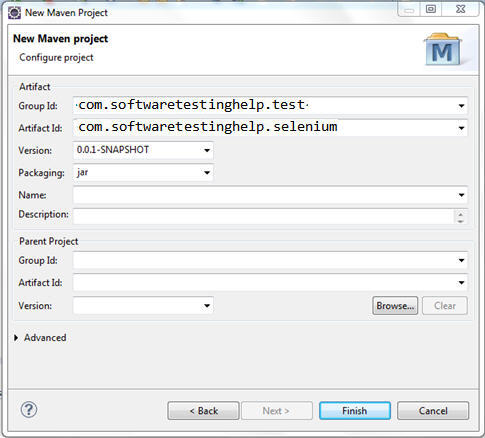
[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-7.jpg)

**Step 2**: Check the Create a simple project and click Next

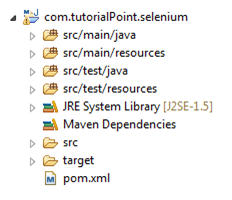
[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-8.jpg)

------------

**Step 3**: Provide Group Id and Artifact Id .You can change the version of Jar as per your wish. Here I am using default name. Click Finish.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-91.jpg)

**Step 4**: After finish you will find the project structure is created like below. pom.xml is created which is used to download all dependencies.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-10.jpg)  
**pom.xml file looks like below:**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | <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"> | | |
| 2 | | <modelVersion>4.0.0</modelVersion> |

|  |  |
| --- | --- |
| 3 | <groupId>com.softwaretestinghelp.test</groupId> |
| 4 | <artifactId>com.softwaretestinghelp.selenium</artifactId> | |

|  |  |  |
| --- | --- | --- |
| 5 | <version>0.0.1-SNAPSHOT</version> | |
| 6 | </project> |

**Step 5**: Add dependencies for Selenium.

All selenium Maven artifacts are available in below central repository

<http://repo1.maven.org/maven2/org/seleniumhq/selenium/>

Add following dependencies in pom.xml for selenium

|  |  |
| --- | --- |
| 1 | <dependency> |
| 2 | <groupId>org.seleniumhq.selenium</groupId> | |

|  |  |  |
| --- | --- | --- |
| 3 | <artifactId>selenium-java</artifactId> | |
| 4 | <version>2.41.0</version> |

|  |  |
| --- | --- |
| 5 | </dependency> |

Similarly, following is the dependency for Junit :

|  |  |
| --- | --- |
| 1 | <dependency> |
| 2 | <groupId>junit</groupId> | |

|  |  |  |
| --- | --- | --- |
| 3 | <artifactId>junit</artifactId> | |
| 4 | <version>4.4</version> |

|  |  |
| --- | --- |
| 5 | </dependency> |

If you want to add other third party jar then add those dependencies in pom.xml

**Step 6**: Final pom.xml will be like below:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | <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"> | | |
| 2 | | <modelVersion>4.0.0</modelVersion> |

|  |  |  |
| --- | --- | --- |
| 3 | <groupId>com.softwaretestinghelp.test</groupId> <artifactId>com.softwaretestinghelp.selenium</artifactId> | |
| 4 | <version>0.0.1-SNAPSHOT</version> |

|  |  |  |
| --- | --- | --- |
| 5 | <dependencies> | |
| 6 | <dependency> |

|  |  |  |
| --- | --- | --- |
| 7 | <groupId>org.seleniumhq.selenium</groupId> | |
| 8 | <artifactId>selenium-java</artifactId> |

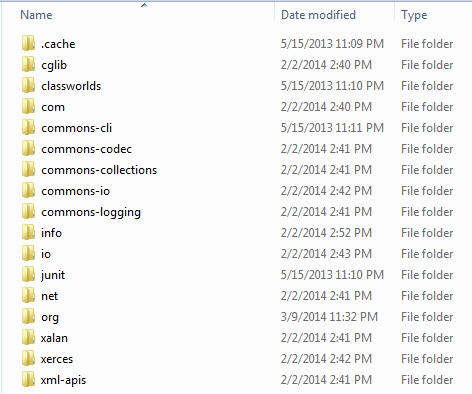
|  |  |  |  |
| --- | --- | --- | --- |
| 9 | <version>2.41.0</version> | | |
| 10 | | </dependency> |

|  |  |  |
| --- | --- | --- |
| 11 | </dependencies> | |
| 12 | </project> |

**Step 7**: Maven will download all the dependency jars in to local repository called .m2.

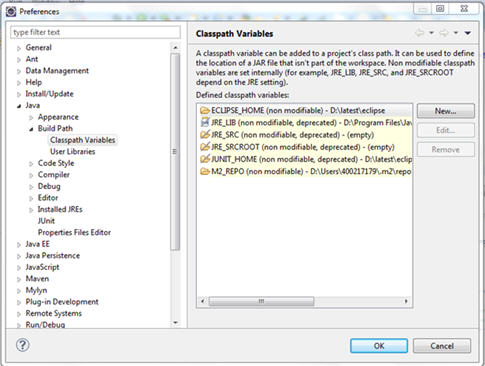
M2 folder is basically inside Users->username->m2

All the jars will be placed in a folder called repository which is inside .m2 folder. Maven will create separate folders for different version and different group id.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-11.jpg)

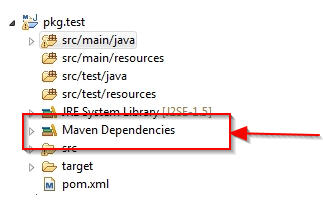
**Step 8**: If m2 folder does not populate in Maven dependencies, then you can populate those jars manually.

– Eclipse Windows ->Preference  
– Navigate Java->Build Path->Classpath Variables

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-12.jpg)

– Click New Button ->Define M2\_REPO and provide the path of m2 folder.

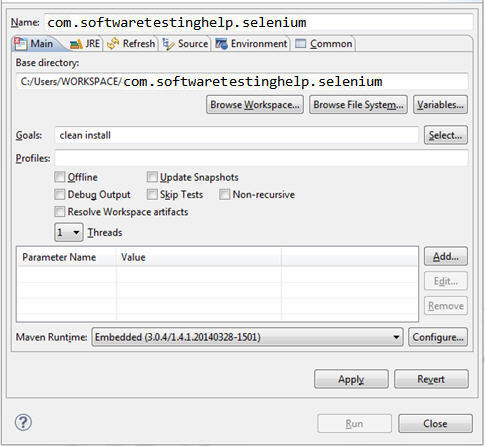
**Step 9**: Upon successful setup you will find Maven Dependencies folder like below which will have the required dependency jar for the project

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-13.jpg)

### ****Build the Project:****

Project can be built by both using IDE and command prompt.

Using IDE you have to right click on POM-Run As-Maven Build

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/Maven-Tutorial-141.jpg)

Enter goals like clean install etc. and click Run.  
Same can be done using command prompt. Navigate to project folder where pom.xml lies.  
And use below commands to clean, compile and install

**For clean**: mvn clean  
**For compile**: mvn compile  
**For Install**: mvn install

Below is the info which is displayed when you clean any project and shows “BUILD SUCCESS”.

|  |  |  |
| --- | --- | --- |
| 1 | [INFO] Scanning for projects... | |
| 2 | [INFO] |

|  |  |  |
| --- | --- | --- |
| 3 | [INFO] ------------------------------------------------------------------------ | |
| 4 | [INFO] Building com.softwaretestinghelp.0.0.1-SNAPSHOT |

|  |  |  |
| --- | --- | --- |
| 5 | [INFO] ------------------------------------------------------------------------ | |
| 6 | [INFO] |

|  |  |
| --- | --- |
| 7 | [INFO] --- maven-clean-plugin:2.4.1:clean (default-clean) @ com.softwaretestinghelp ---[INFO] Deleting C:\Users\rshwus\WORKSPACE\com.softwaretestinghelp\target |
| 8 | [INFO] ------------------------------------------------------------------------ |

|  |  |  |
| --- | --- | --- |
| 9 | [INFO] BUILD SUCCESS | |
| 10 | | [INFO] ------------------------------------------------------------------------ | |

|  |  |
| --- | --- |
| 11 | [INFO] Total time: 0.702s |
| 12 | [INFO] Finished at: Sat May 24 18:58:22 IST 2014 | |

|  |  |
| --- | --- |
| 13 | [INFO] Final Memory: 2M/15M |
| 14 | [INFO] ------------------------------------------------------------------------ | |

### ****Conclusion:****

**Maven simplifies the code handling and process of building the project**. Most of the projects follow maven structure.

Download all dependencies provided the dependencies are available in maven central repository. If any of the dependency is not available in maven central repository then you have to add repository path in pom.xml explicitly.

There are many other build tools available in like ant. But it is better to use maven while dealing with different versions and different dependencies. Maven even can manage the dependencies of dependencies. Other tools may not provide such flexibility like maven. Please post your queries anything related to maven here.