Maven

What is Maven?

Maven is a project management and comprehension tool.

Maven provides developers a complete build lifecycle framework.

Can automate the project's build

Maven uses a standard directory layout

default build lifecycle.

creating reports, checks, build and testing automation setups.

Maven provides developers ways to manage following:

* Builds
* Documentation
* Reporting
* Dependencies
* SCMs
* Releases
* Distribution
* mailing list

Maven project structure and contents are declared in an xml file, pom.xml referred as Project Object Model (POM), which is the fundamental unit of the entire Maven system.

Maven uses *Convention* over *Configuration* which means developers are not required to create build process themselves.

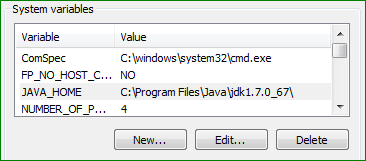
When a Maven project is created, Maven creates default project structure. Developer is only required to place files accordingly and he/she need not to define any configuration in pom.xml.

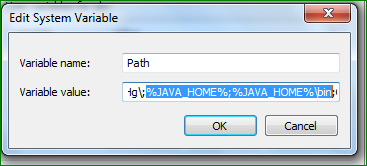
As an example, following table shows the default values for project source code files, resource files and other configurations. Assuming, **${basedir}** denotes the project location:

|  |  |
| --- | --- |
| **Item** | **Default** |
| source code | ${basedir}/src/main/java |
| resources | ${basedir}/src/main/resources |
| Tests | ${basedir}/src/test |
| distributable JAR | ${basedir}/target |
| Complied byte code | ${basedir}/target/classes |

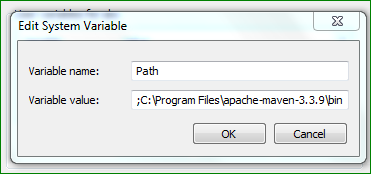
My Computer >> Right Click >> Properties >> Advance System Settings >> Advanced >> Environmental Variables -

How to set JAVA\_HOME Path:





How to set Maven Path:



Running eclipse and also running Maven will require you to store two path variables, one in your jdk1.7\_x\_x\_x location and also in your jdk1.7\_x\_x\_\bin. If you are using Windows, when you are in your environment variables, do the following:

1) create a USER variable called JAVA\_HOME. Point this to the location of your JAVA file. For example: "C:\Program Files\Java\jdk1.7.0\_51" (remove the quotes)

2) under the PATH, append %JAVA\_HOME% to the PATH. This will add the file location from step 1 to your PATH. This is good for MAVEN

3) if you are using eclipse you need to have the path point to "C:\Program Files\Java\jdk1.7.0\_51\bin". Now append %JAVA\_HOME%\bin to the end of your path.

4) your path should look something like this: C:\Program Files (x86)\Google\google\_appengine\;C:\Users\username\AppData\Roaming\npm;%M2%;%JAVA\_HOME%;%JAVA\_HOME%\bin

Notes: the items that are enclosed in %'s like %M2% are assigned variables. It looks redundant but necessary. You can confirm that everything works by typing in:

java -version   
javac -version   
mvn -version

**POM**

POM stands for *Project Object Model*.

It is fundamental Unit of Work in Maven

It is an XML file.

It always resides in the base directory of the project as pom.xml.

POM contains information about the project and various configuration detail used by Maven to build the project(s).

POM also contains the goals and plugins.

While executing a task or goal, Maven looks for the POM in the current directory.

It reads the POM, gets the needed configuration information, then executes the goal.

Some of the configuration that can be specified in the POM are following:

* project dependencies
* plugins
* goals
* build profiles
* project version
* developers
* mailing list

First decide the project **group** (groupId), its**name**(artifactId) and its **version.**

These attributes help in uniquely identifying the project in repository.

It should be noted that there should be a single POM file for each project.

* All POM files require the **project** element and three mandatory fields:**groupId, artifactId,version.**
* Projects notation in repository is **groupId:artifactId:version.**
* Root element of POM.xml is **project** and it has three major sub-nodes :

|  |  |
| --- | --- |
| **Node** | **Description** |
| groupId | This is an Id of project's group. This is generally unique amongst an organization or a project. For example, a banking group com.company.bank has all bank related projects. |
| artifactId | This is an Id of the project.This is generally name of the project. For example, consumer-banking. Along with the groupId, the artifactId defines the artifact's location within the repository. |
| version | This is the version of the project.Along with the groupId, It is used within an artifact's repository to separate versions from each other. For example:   *com.company.bank:consumer-banking:1.0*  *com.company.bank:consumer-banking:1.1.* |

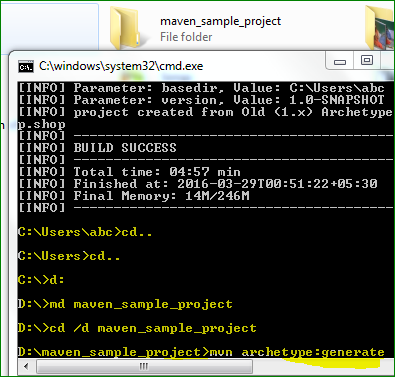
**Commands**

**mvn archetype:generate > d:\archetypes.txt**

**This will create text file detailing about the archetype available.**

An easy way to look at the default configurations of the super POM is by running the following command: **mvn help:effective-pom**

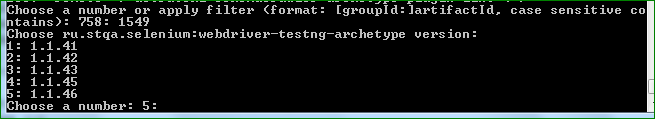
**TO create maven project and its directory:**

****

**Now run below command in created directory to generate goal**

**D:\maven\_sample\_project>mvn archetype:generate**

**After this apply archetype number we want to set maven project, as shown below**

****

**Give version number:**

****

**Now give group Id: example I gave group id as com.selcomp.seltest**

****

**Now to give artifacts: this is name or id of project. Example – VisiCheck**

****

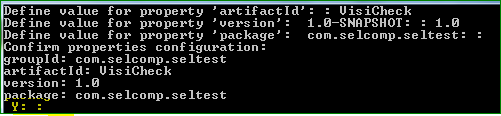
**Now to give version of the project: Example I gave version number as 1.0**

****

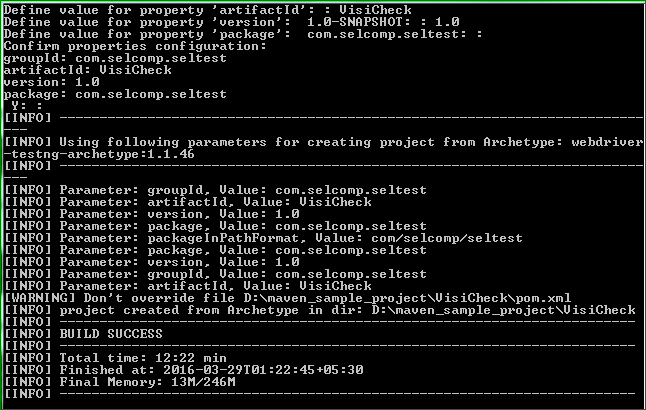
**Next it will ask about the package confirmation by suggesting default package name. press enter as default**

****

**Now it will ask for confirmation. Hit enter or write Y and enter**

****

**Project created successfully:**

****

[**http://www.tutorialspoint.com/maven/maven\_build\_test\_project.htm**](http://www.tutorialspoint.com/maven/maven_build_test_project.htm)

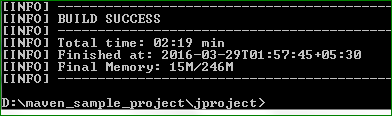
**I have created simple java project “jproject”with archetype id 758 i.e.**

**758: remote -> org.apache.maven.archetypes:maven-archetype-quickstart (An archetype which contains a sample Maven project.)**

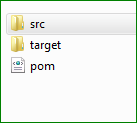
**And then executed command to build the project:**

C:\MVN\consumerBanking>mvn clean package

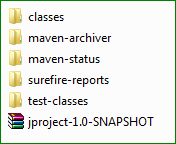
**>> D:\maven\_sample\_project\jproject> mvn clean package**

****

**Check Maven project folder. A target folder is created in the project folder.**

****

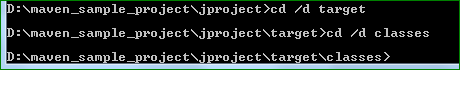
**And inside target folder**

****

**Now move to target folder**

****

**Then move to classes folder**

****

**Now execute below command on app (demo example set by java project app.java; aaptest.java) to execute java demo example**

****

**Read: For adding Java source files**

**http://www.tutorialspoint.com/maven/maven\_build\_test\_project.htm**

**Read: For maven external dependencies**

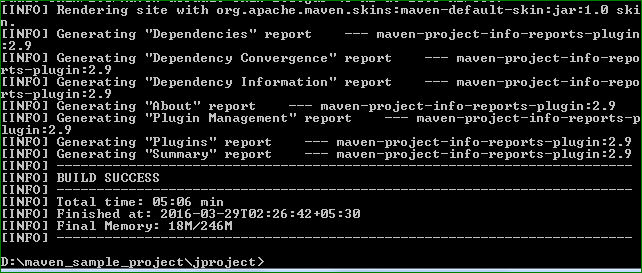
[**http://www.tutorialspoint.com/maven/maven\_external\_dependencies.htm**](http://www.tutorialspoint.com/maven/maven_external_dependencies.htm)

**How to create Project documentation for created project:**

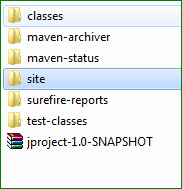
**Go to project folder on command prompt and execute below command**

****

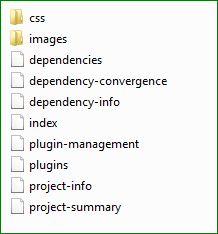
**You will see**

****

**Project Folder will get update**

****

**Inside Site folder**

****

**Click on Index to see the documentation**

****

Maven creates the documentation using a documentation-processing engine called [Doxia](http://maven.apache.org/doxia/index.html) which reads multiple source formats into a common document model.

**http://maven.apache.org/doxia/index.html**

**ECLIPSE (M2Eclipse)**

[**http://www.tutorialspoint.com/maven/maven\_eclispe\_ide.htm**](http://www.tutorialspoint.com/maven/maven_eclispe_ide.htm)

[**http://books.sonatype.com/m2eclipse-book/reference/install-sect-marketplace.html**](http://books.sonatype.com/m2eclipse-book/reference/install-sect-marketplace.html)

**For integrating Maven with Eclipse we need to “m2eclispe” plugin.**

**Features of m2eclipse:-**

* You can run Maven goals from Eclipse.
* You can view the output of Maven commands inside the Eclipse using its own console.
* You can update maven dependencies with IDE.
* You can Launch Maven builds from within Eclipse.
* It does the dependency management for Eclipse build path based on Maven's pom.xml.
* It resolves Maven dependencies from the Eclipse workspace without installing to local Maven repository (requires dependency project be in same workspace).
* It automatic downloads required dependencies and sources from the remote Maven repositories.
* It provides wizards for creating new Maven projects, pom.xml and to enable Maven support on existing projects
* It provides quick search for dependencies in remote Maven repositories

**Download Eclipse from**

[**http://www.eclipse.org/downloads/index-helios.php**](http://www.eclipse.org/downloads/index-helios.php)

[**Eclipse IDE for Java EE Developers**](http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/heliosr), 206 MB OR [**Eclipse IDE for Java Developers**](http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/heliosr), 99 MB

To install eclipse, just unzip folder and click on eclipe.exe.

Now, we have to install m2eclipse to integrate maven with eclipse:-

<http://books.sonatype.com/m2eclipse-book/reference/install-sect-marketplace.html>

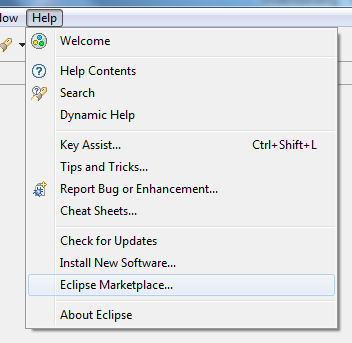
Use this link to know how to install m2eclipse

**Steps:**

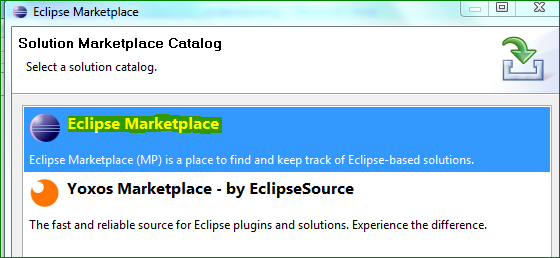
**1. Installing m2eclipse in Eclipse 3.6 (Helios) with the Eclipse Marketplace**

Marketplace makes it easy to install m2eclipse and m2eclipse (Extras) in Eclipse 3.6 (Helios)

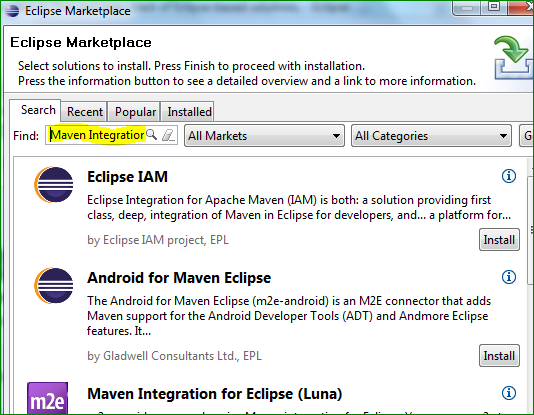
2.  open the Eclipse Marketplace, go to the Eclipse Help menu and select Eclipse Marketplace

****

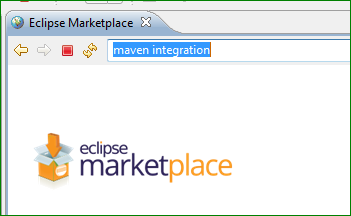
**Select Eclipse Marketplace**

****

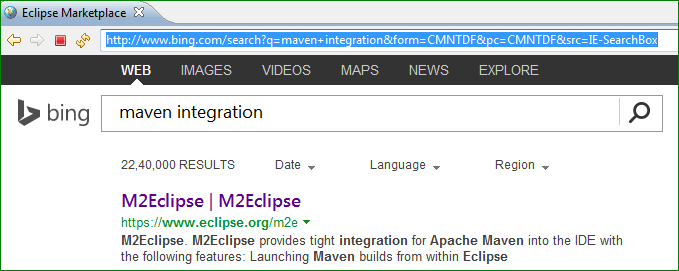
**Search “Maven Integration” as below**

****

**If not found than click on Switch Marketplace > search for maven integration**

****

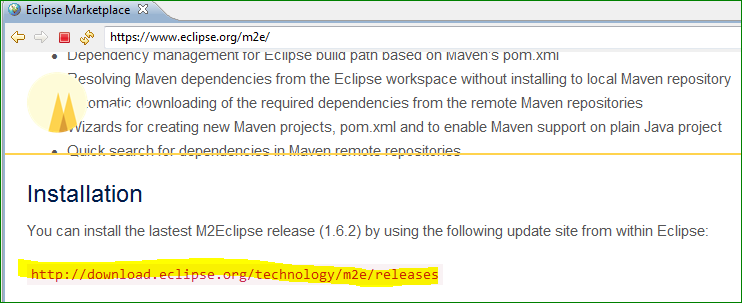
**Click on first link M2Eclipse as below:**

****

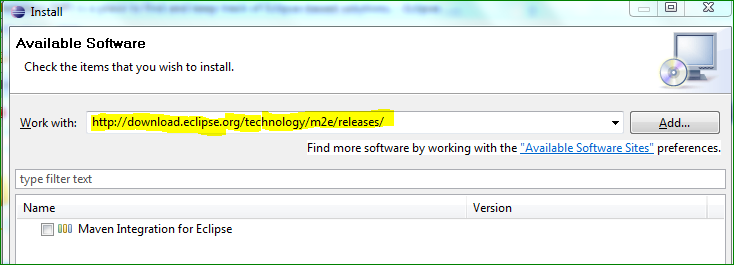
**Web page with M2Eclipes details will be shown. Also, it will mention below** [**URL:-**](URL:-)

[**http://download.eclipse.org/technology/m2e/releases/1.3**](http://download.eclipse.org/technology/m2e/releases/1.3)

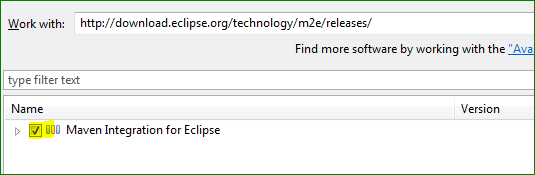
<http://download.eclipse.org/technology/m2e/releases/1.3>

****

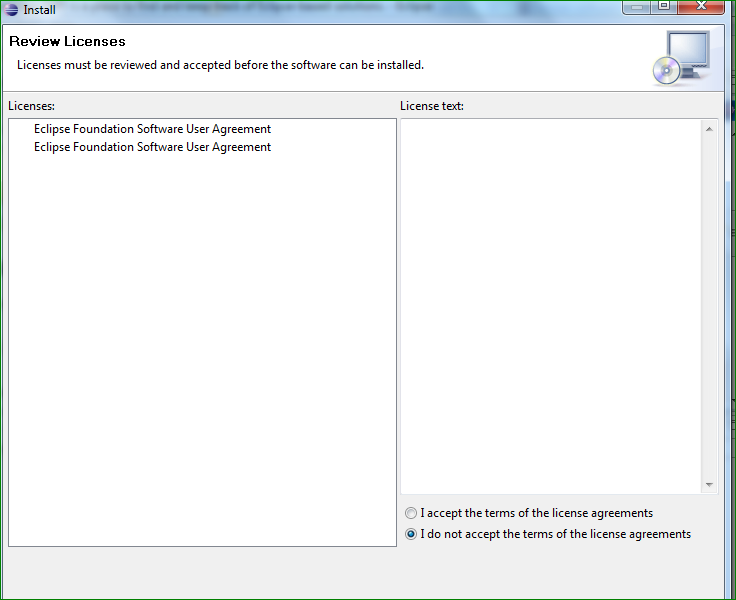
**Copy this URL and open “Install New Software” from Help menu of eclipse and paste this into Work With field and hit enter**

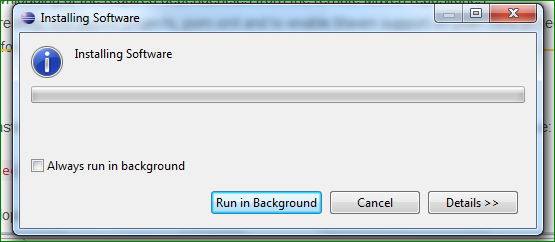
****

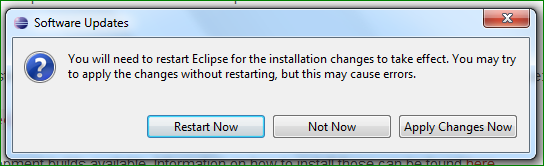
**Wait and select Maven Integration for Eclipse check box and click Next button**

****

**Select Terms and Condition to agree.**

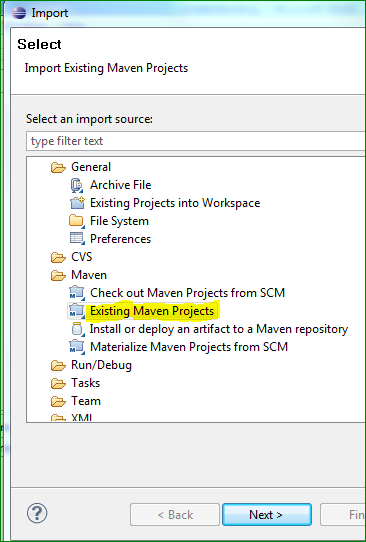
****

****

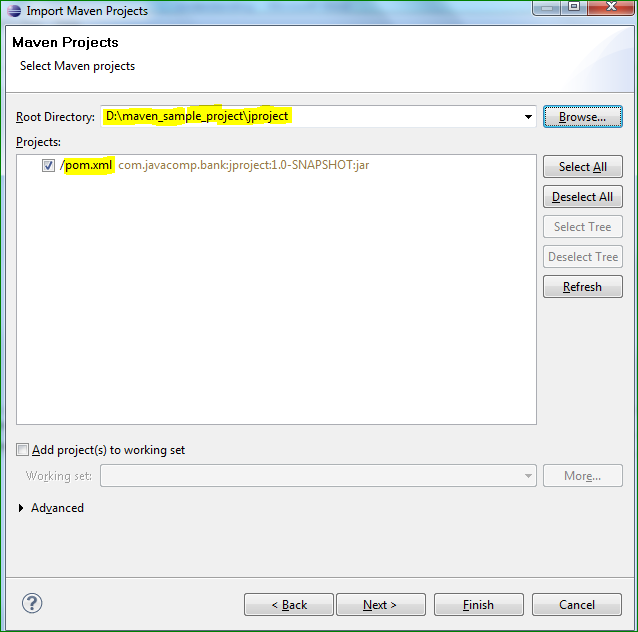
****

**Exporting Maven Project to Eclipse:-**

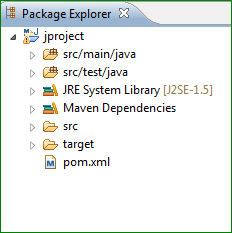
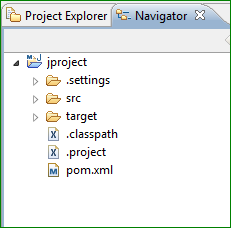
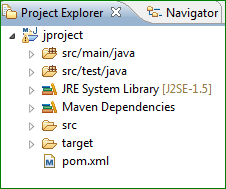
* Open Eclipse.
* Select **File > Import >** option.
* Select Maven Projects Option. Click on Next Button.

****

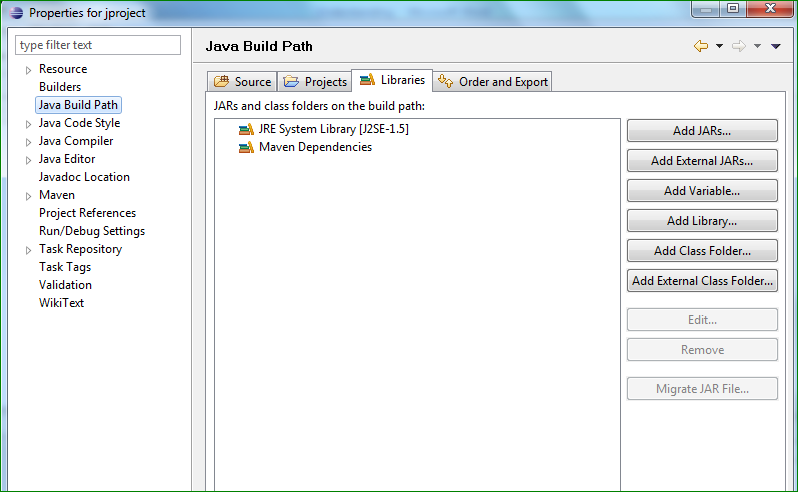
* Click Next
* Browse root directory where maven project is created.
* It will analyze the root directory and POM file there. Click Finish.
* POM file with package name will be shown

****

**Project imported to Eclipse and shown on left side in project explorer.**

**  **

**Right Click on Project Name and select Properties >> Eclipse has added Maven dependencies to java build path.**

****

How to build project using maven capability of eclipse:-

* Right Click on project name to open context menu.
* Select Run as option
* Then maven package option

Maven will start building the project. You can see the output in Eclipse Console

**To Do**

**To Learn what is Maven Package**

**Project >> right click >> Run as >> Maven Build >> In Goal write “Package”**

**Maven package:-**

**http://stackoverflow.com/questions/9422142/missing-the-maven-package-menu-entry-in-eclipse**

**https://www.eclipse.org/forums/index.php/t/224163/**

**https://bugs.eclipse.org/bugs/show\_bug.cgi?id=354289**

**chrome-extension://ninejjcohidippngpapiilnmkgllmakh**

**Configuring Webdriver :**

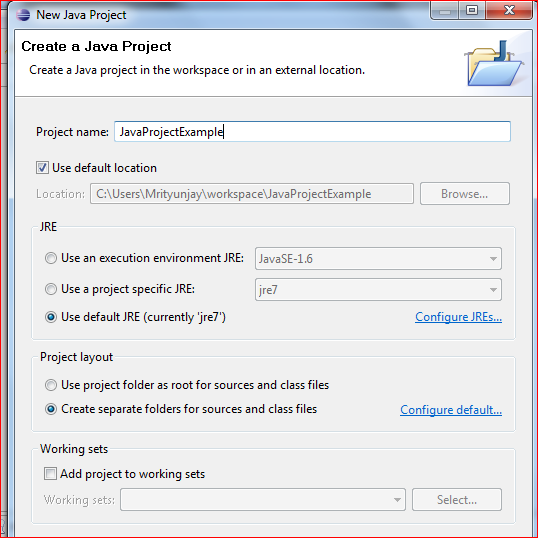
**For automation with selenium and Java language we need to add language specific client drivers – Selenium Java Client Libraries.**

**Download this from -** [**http://docs.seleniumhq.org/download/**](http://docs.seleniumhq.org/download/)

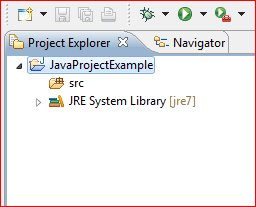
**How To Setup a Selenium WebDriver Project in Eclipse ?**

[**http://www.assertselenium.com/eclipse-2/how-to-setup-a-webdriver-project-in-eclipse/**](http://www.assertselenium.com/eclipse-2/how-to-setup-a-webdriver-project-in-eclipse/)

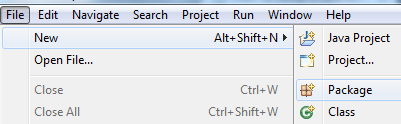
* **Step 1: Download Eclipse**
* **Step 2: Creating a New Java Project > File > Java Project > Enter Project Name > Finish**

****

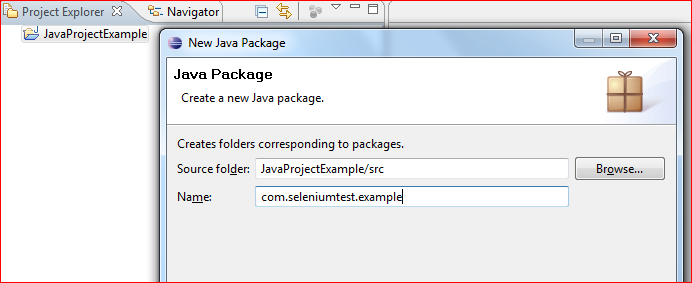
* **It will be shown like this after project is created.**

****

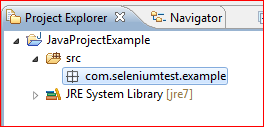
* **Step 3: Creating a New Package  
   Right Click on the New Java Project created > New > Package**

****

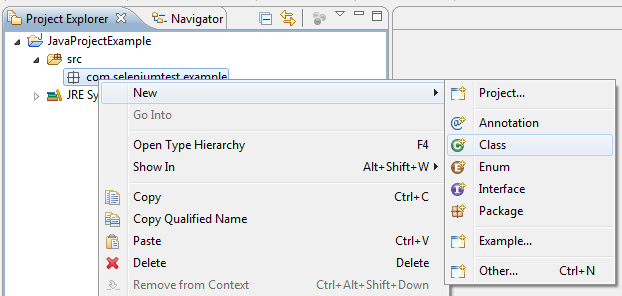
* **Enter package name: example – com.seleniumtest.example > Click finish**

****

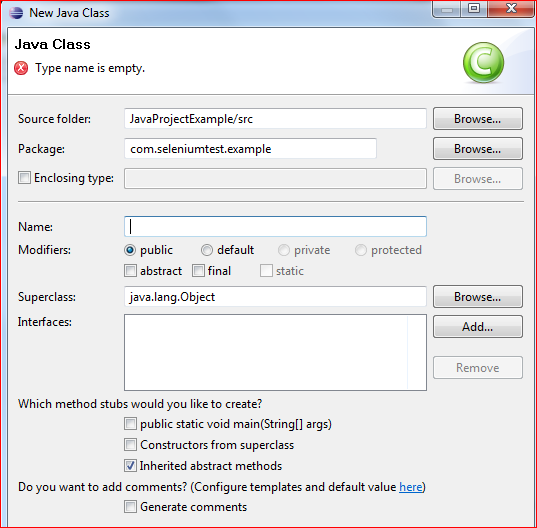
* **Project with package will be shown like this:**

****

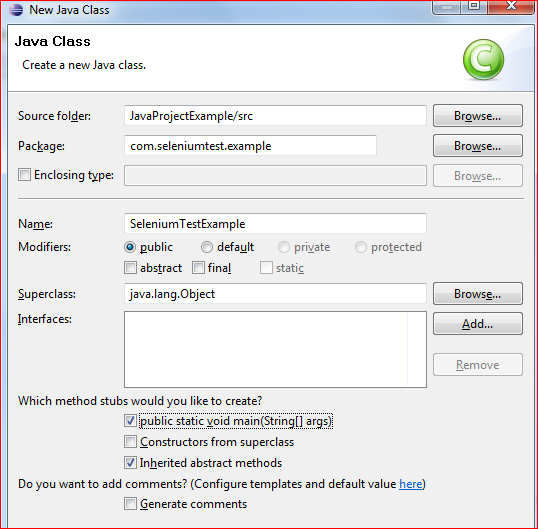
* **Step 4: Creating a New Java Class file.  
  Right Click on the Newly created Package > New > Class**

****

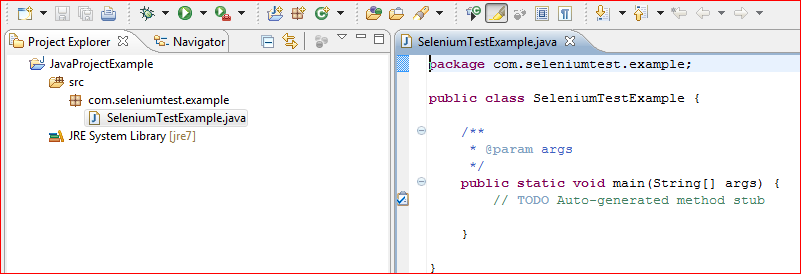
* **Default screen**

****

* **Enter class name – example: SeleniumTestExample (Java Class Name Starts with Uppercase)**
* **Also select – public static void main(String[] args)**
* **Click on Finish**

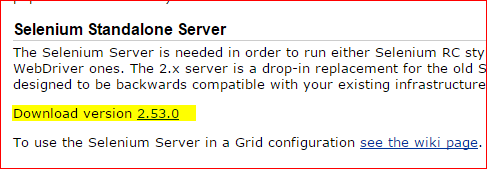
****

* **Project will get setup and will look like**

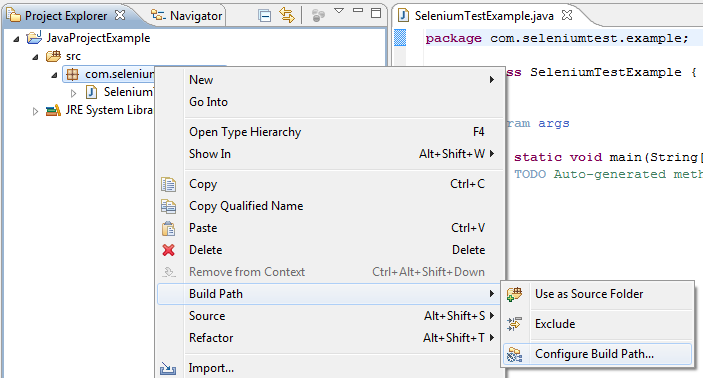
****

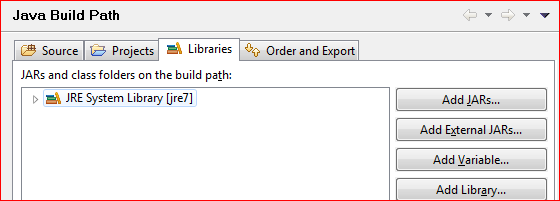
* **To add new Class repeat same steps > Right click on Package > Class > Enter class name > select public string and > click finish**
* **Now, we need to add the Selenium Library files to our project  
  Step 5: Download the selenium server from the seleniumhq.org website.**

[**http://docs.seleniumhq.org/download/**](http://docs.seleniumhq.org/download/)

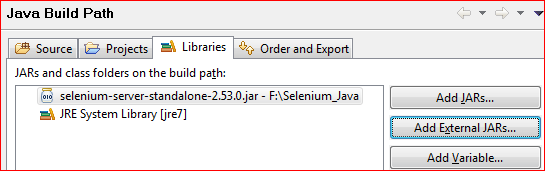
****

* **Save jar file to any drive you want**
* **Now we need to add this jar file to eclipse for this:-**
* **Step 6: Configuring Build Path.  
  Right Click on the package → Build Path → Configure Build Path**

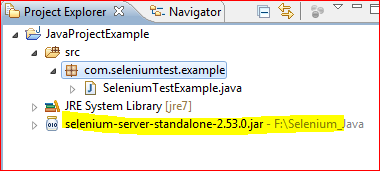
****

****

* **Click on Library tab**
* **Click on “Add External Jars” > Browse “selenium-server-standalone-2.53.0”**

****

* **Click Finish**

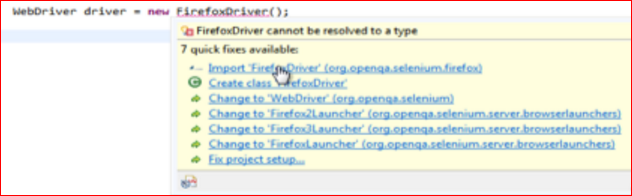
****

**Now Project is ready for Selenium automation with Java.**

* **There may be error if import statements is not used written in selenium code**
* **To resolve this either write import statements like –**

**import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.firefox.FirefoxDriver; Or**

* **Hover mouse over the line where error is coming. Auto suggestion will be shown to import. Select particular driver. This will add import statement in the code.**

****

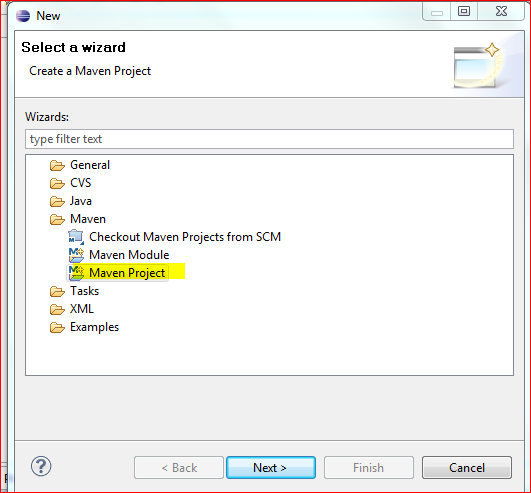
**How to Create Maven Project directly from Eclipse for Selenium Webdriver?**

[**http://www.guru99.com/maven-jenkins-with-selenium-complete-tutorial.html**](http://www.guru99.com/maven-jenkins-with-selenium-complete-tutorial.html)

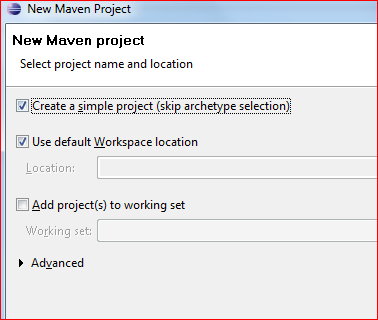
[**http://toolsqa.com/java/maven/create-new-maven-project-eclipse/**](http://toolsqa.com/java/maven/create-new-maven-project-eclipse/)

**Precondition:- Eclipse install; maven Setup; M2e Integration**

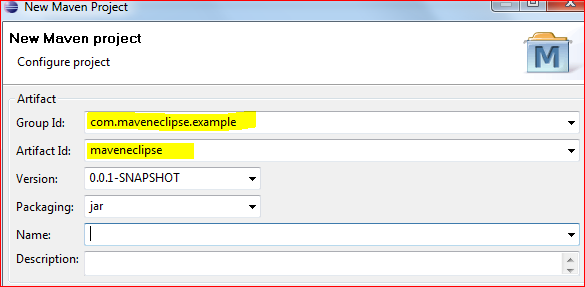
* In Eclipse IDE, create a new project by selecting File > New > Other > Maven Project > Click Next

****

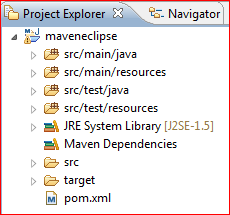
* On the **New Maven Project** dialog select the **Create a simple project (Skip Archetype Selection)** and click Next

****

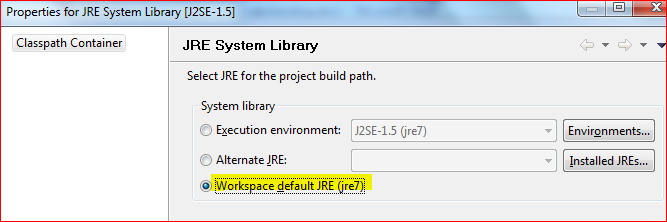
* **Enter Group id and Artificat Id**
* **Example – com.maveneclipse.example; maveneclipse**

****

* **Click Finish > Basic maven project with POM.xml will be created as shown below:**

****

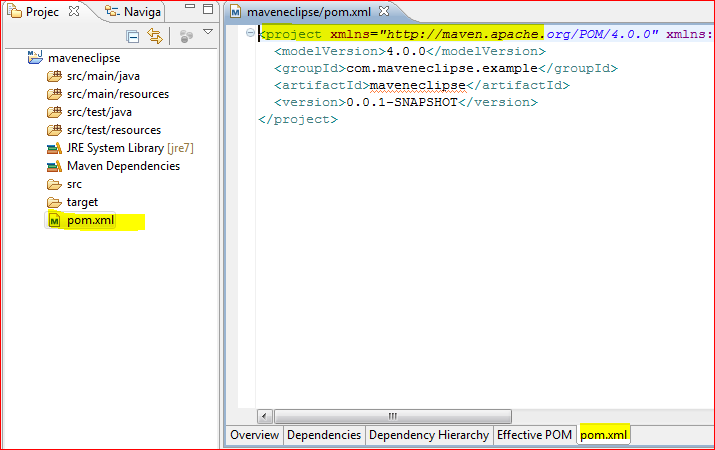
* **Right click on JRE system library > Select Properties > Select Workspace default JRE (jre7)**

****

* **Click Ok**

**Editing and adding dependencies to POM.xml**

* **Select pom.xml from Project Explorer and select pom.xml from tab**
* **Default pom will be like below**

****

* **Default pom will only contain group id and artifact.**
* **Now we have to add dependencies for Selenium and TestNG**
* **Before adding TestNg lets add tesNg Plugin**

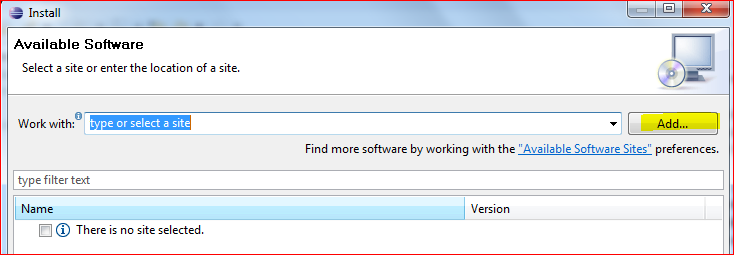
**How to Add TestNG plugin:-**

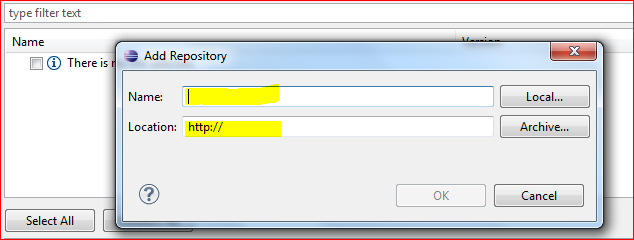
[**http://testng.org/doc/eclipse.html**](http://testng.org/doc/eclipse.html)

[**http://howtodoinjava.com/testng/testng-tutorial-with-eclipse/**](http://howtodoinjava.com/testng/testng-tutorial-with-eclipse/)

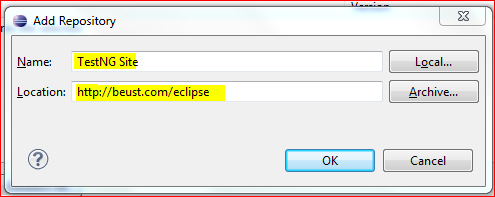
[**https://maven.apache.org/general.html#tools-jar-dependency**](https://maven.apache.org/general.html#tools-jar-dependency)

* **Eclipse > Help > Install New Software**
* **Click Add Button**

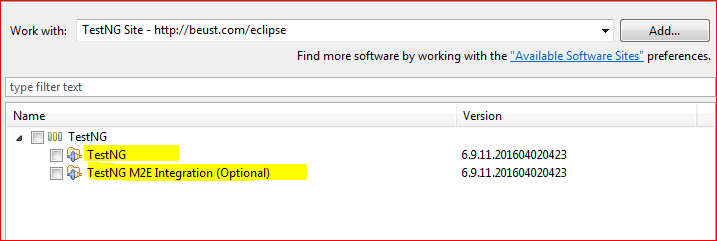
****

****

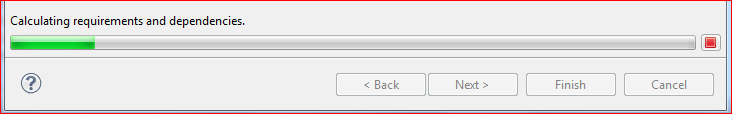
* **Write TestNG Site and** <http://beust.com/eclipse> **in name and location**
* **Click OK**



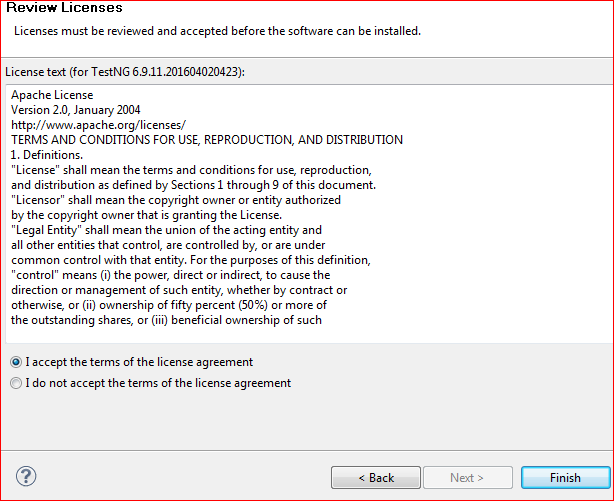
* **On clicking OK, TestNG update site will get added to Eclipse. The available software window will show the tools available to download under the TestNG site.**

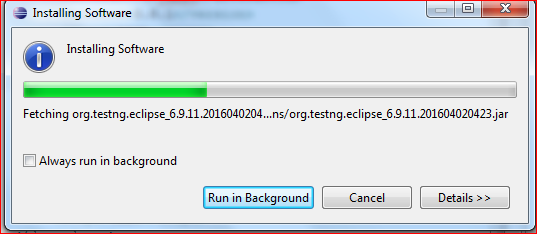


* **NOTE: since TestNG Eclipse Plugin 6.9.10, there is a new optional plug-in for M2E (Maven Eclipse Plugin) integration. It's recommended to install it if your Java project(s) are managed by Maven.**
* **Select both checkbox and click Next**

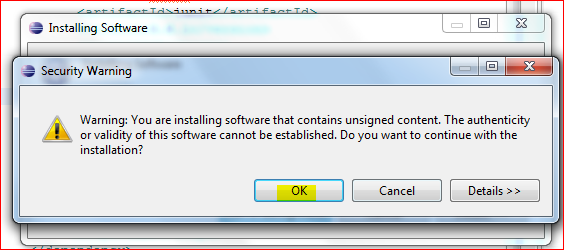


* **For now uncheck TestNG M2E Integration (optional).**
* **Some error came while installing above integration so again select onl TesTNG and click next**
* **Select I agree and click Finish. This will install software.**

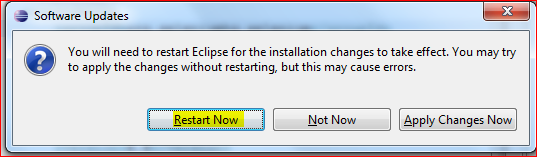
****

****

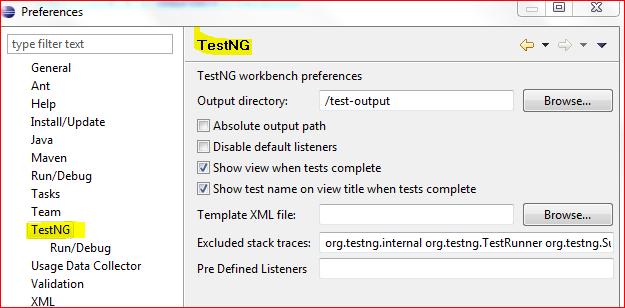
* **Click Ok for warning**

****

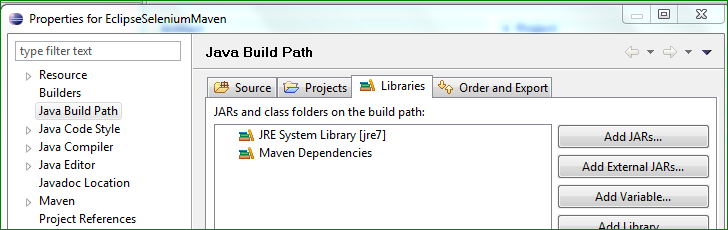
* **Restart Eclipse on asking**

****

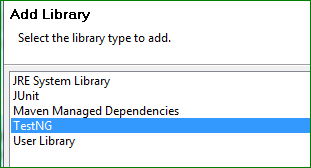
* **After restart > Eclipse > Windows > Preference > TestNG will be shown.**

****

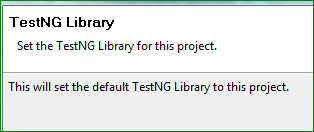
* **This completes installing TestNG successfully.**
* **Now either create Java project or Maven Project**
* **After creating project we have to add TestNG to build path.**
* **Right click on Project > Properties > Java Build Path > Select Libraries**

****

* **Click on Add Library and Select TestNG**

****

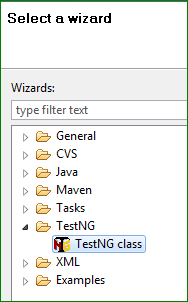
* **Click Next**

****

* **Click Finish and Ok**

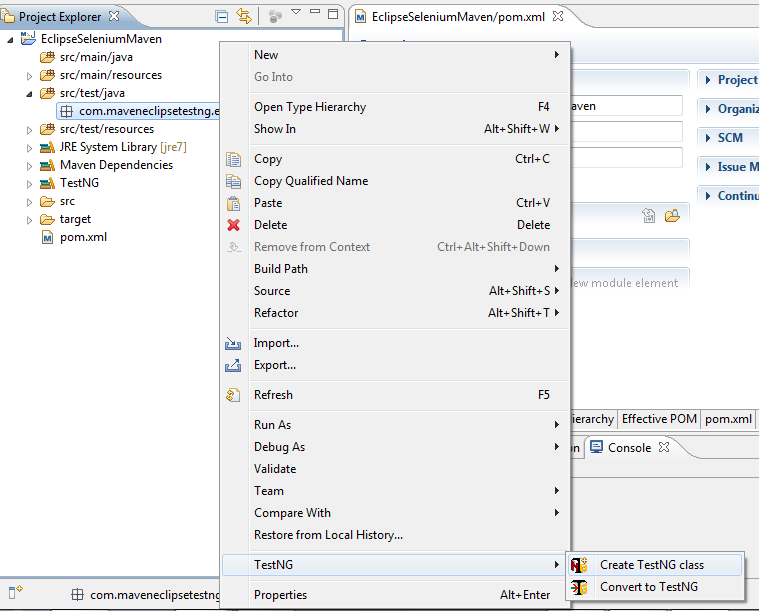
**Now we need to create TestNG class to proceed.**

* **To add TestNG Class > File > New > Other > Select TestNG Class > Click Next**

****

**NOTE: We cannot add TestNG class without adding any Package to project. So while setting up the project, we need to add Package. Now to this package we can later add TestNG class. See image:**

****

****

* **Right click on package added under src/test/java**