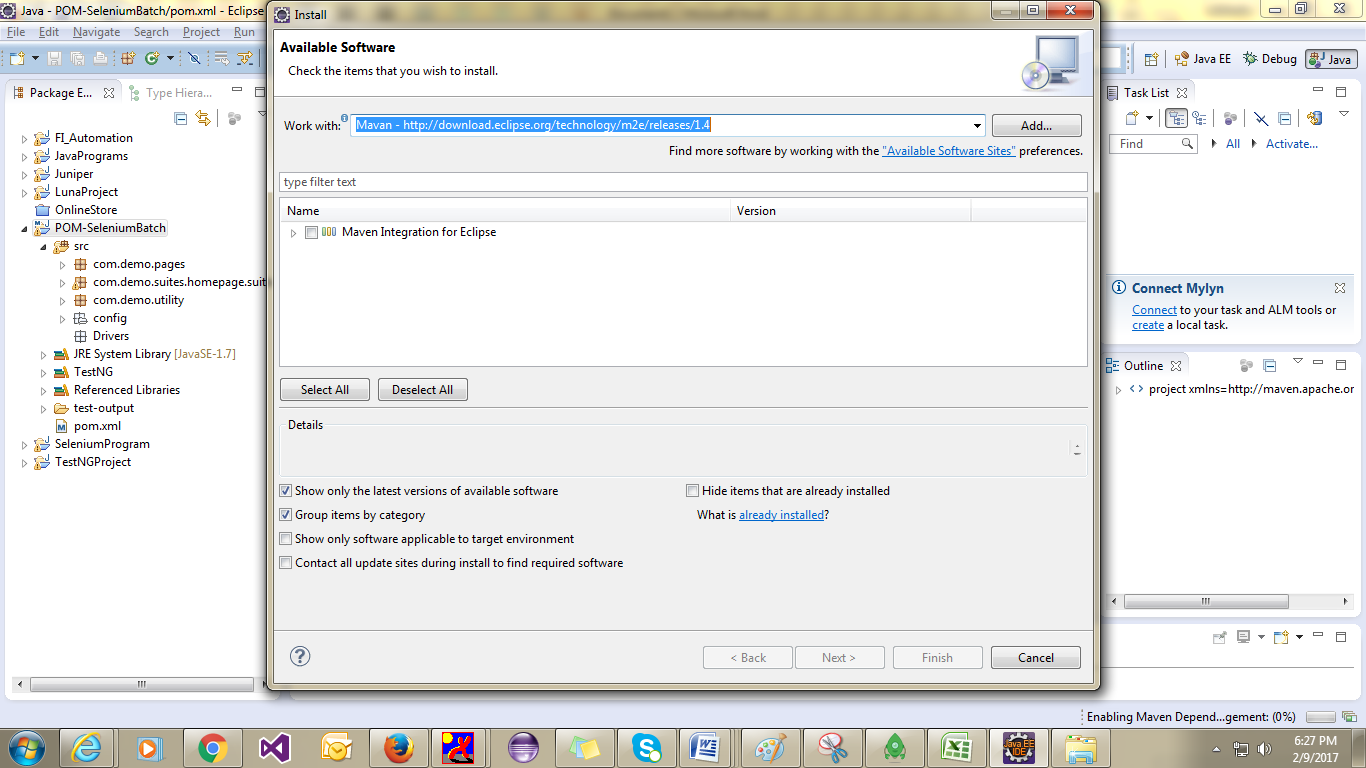
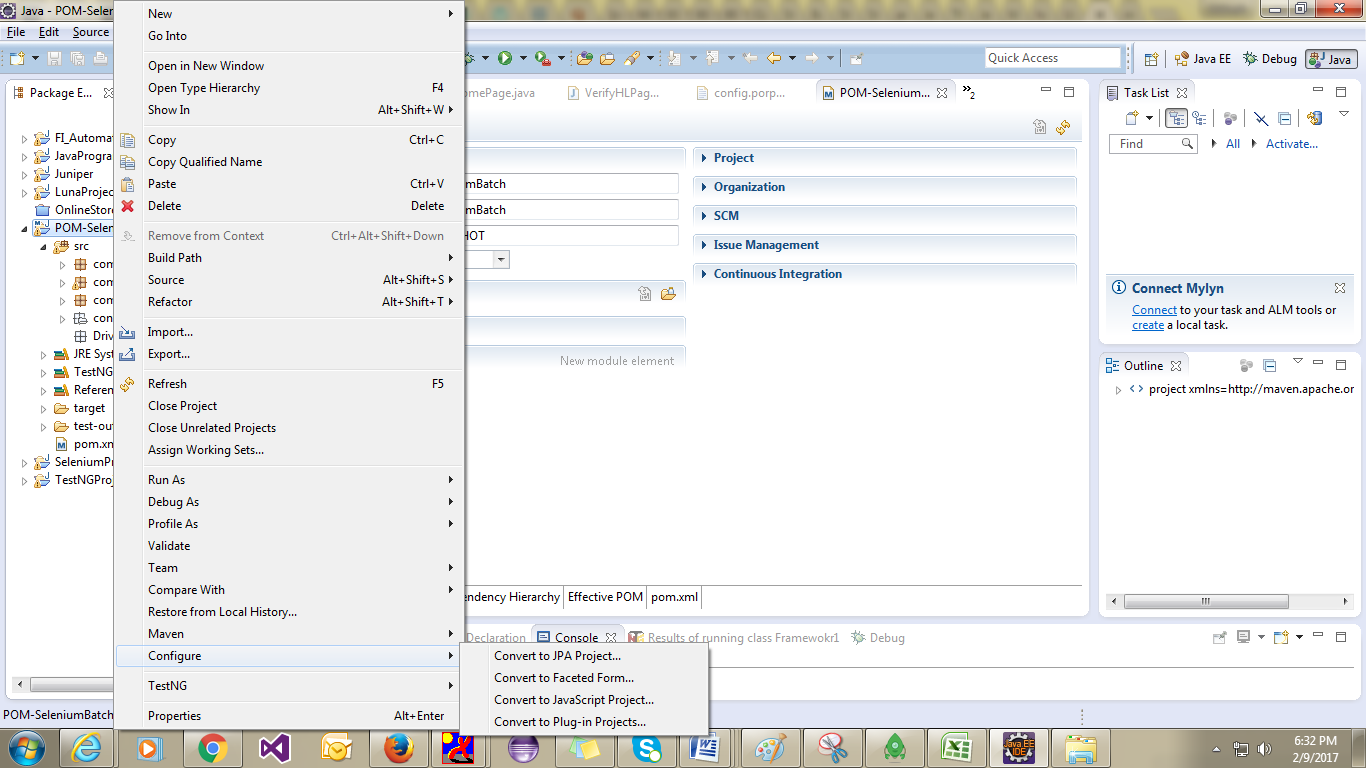
1. Help->Install New Software -> Configure to Maven from

<http://toolsqa.com/java/maven/configure-selenium-continuous-integration-maven/>

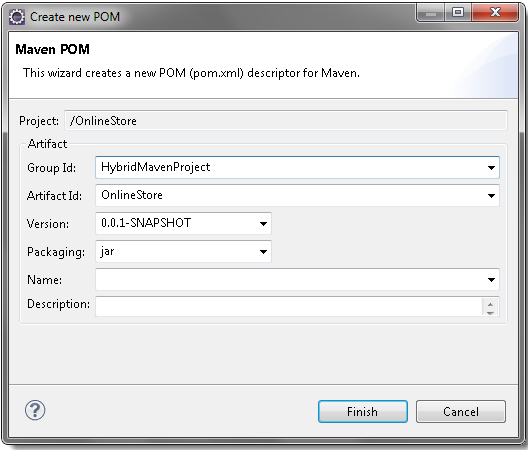
<http://www.softwaretestinghelp.com/maven-project-setup-for-selenium-selenium-tutorial-24/>



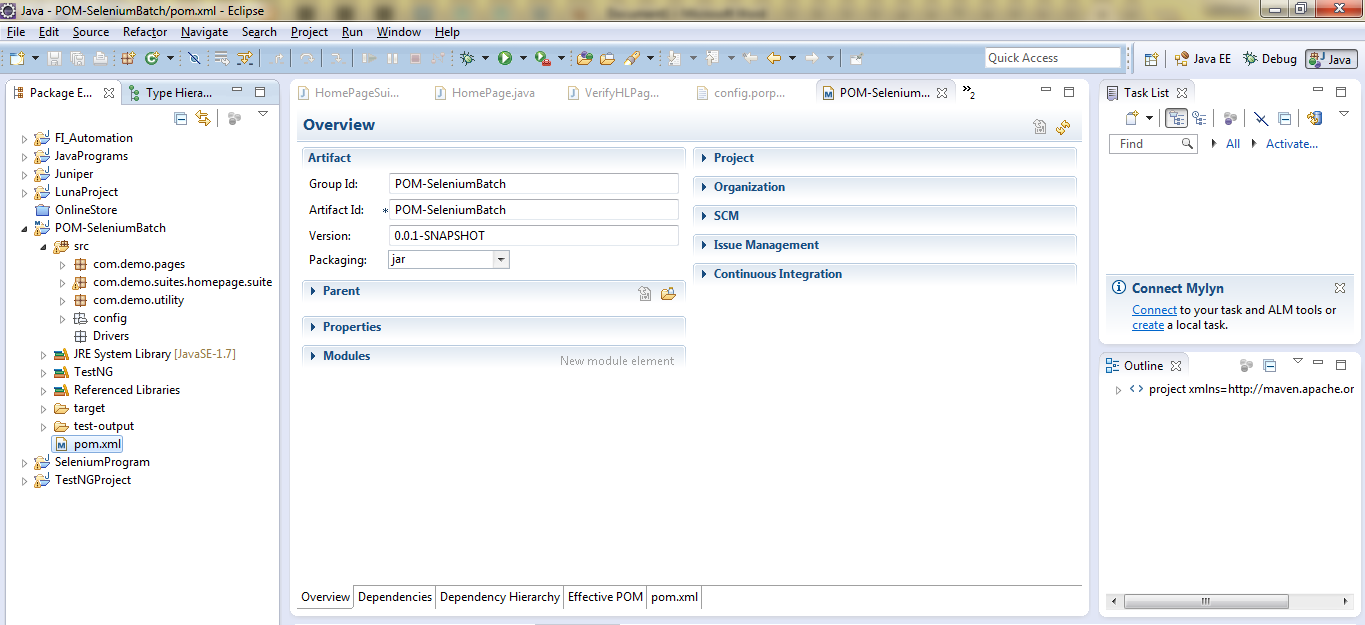
1. Convert your project into Maven



1. Click on Finish so it will create Pom.Xml File in your Project



1. *Eclipse* will take few seconds to convert the project and once it is completed with the process, the *project explorer* window will look like this. There will be few errors in the few packages of the project and a new **pom.xml** file is created at the bottom.



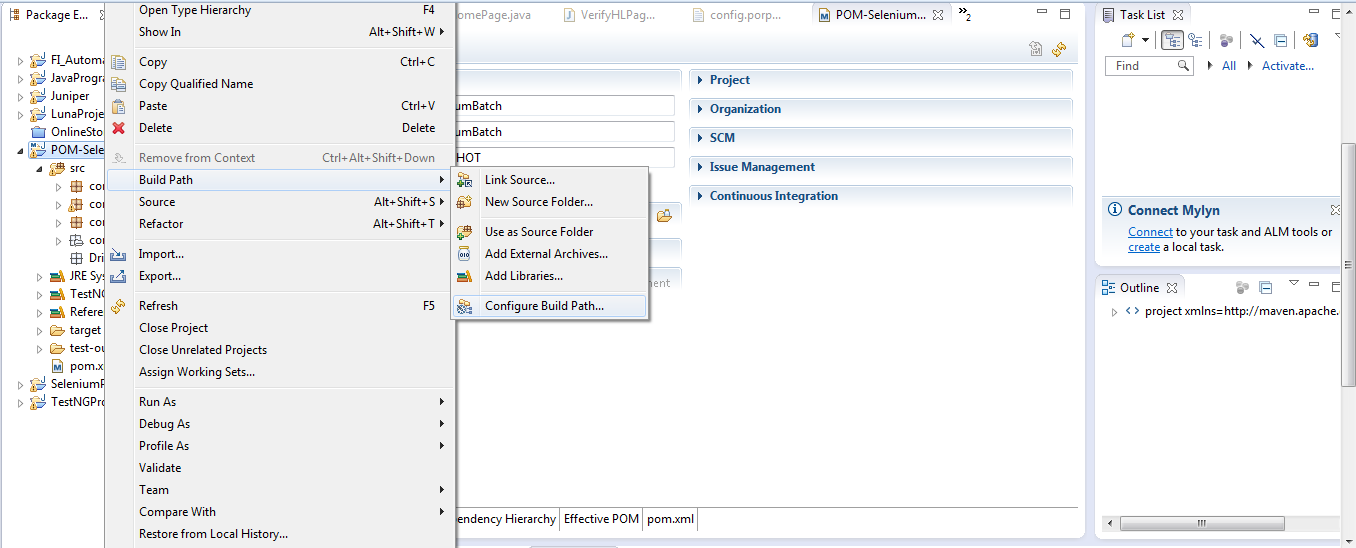
1. Remove Associated Libraries from the Project Build Path

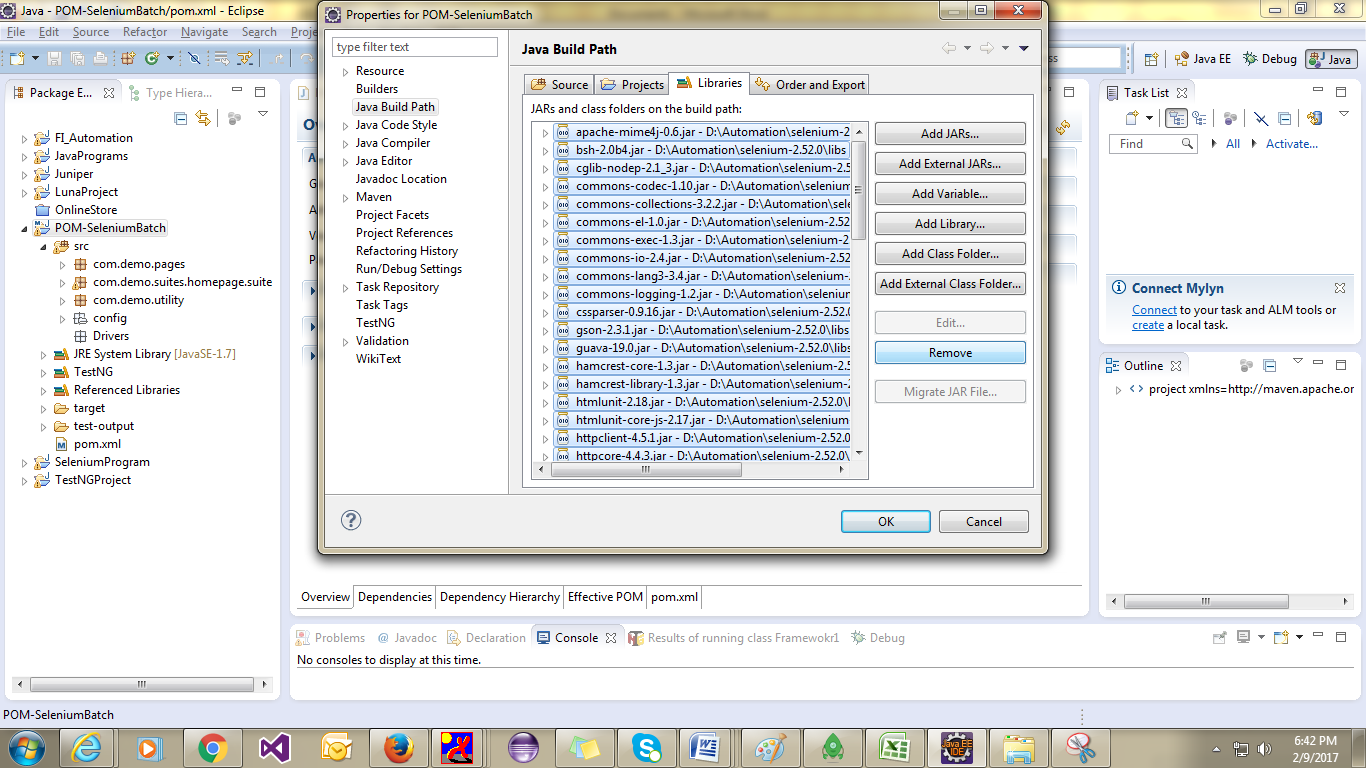
During the build, Maven only considers its own libs/repository and it will not consider the libraries available in the project build path. So the next step is to add all those libraries and dependencies which is required in the project. To figure out what all is required, just remove all the libraries from the build path and the project will end up with many more errors. Then you can browse to every single error and one by one add missing jars files to maven repository.

1) Right Click on the project explorer and select Build Path > Configure Build Path.

2) Select the Libraries tab.

3) Select all the available libraries and Remove.





Lets just open the ***Utils***class under the ***utility***package. Notice that the first error is displayed on the ***WebDriver***. This is because we just removed the Selenium Libraries from the build path. So it is the time to add Selenium Libraries under the ***Maven Dependencies***.

### Dependency Keyword

Dependencies are the libraries, which are required by the project. For example Log4j jars, Apache Poi jars, Selenium Jars etc.

### Maven Local Repository

This is the place where Maven stores all the project jars files or libraries or dependencies. By default the folder name is ‘**.m2**‘and by default the location in windows 7 is ‘**Libraries\Documents\.m2**‘or ‘***C:\Users\yourusername\.m2***‘.

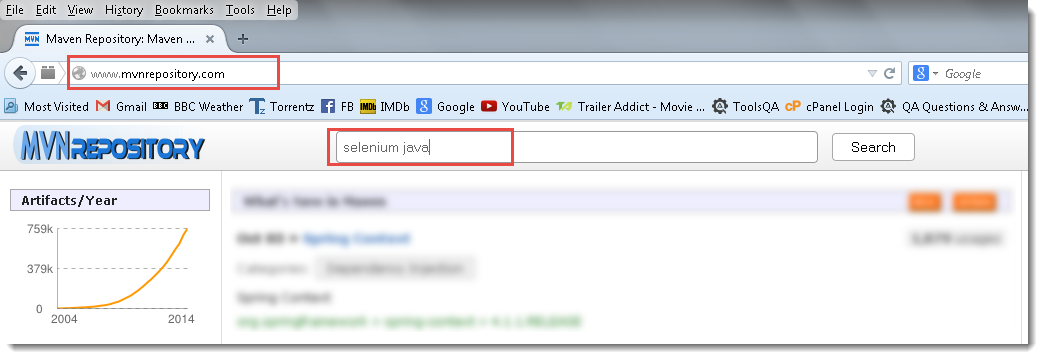
### Maven Central Repository

Maven central repository is the default location ‘**http://mvnrepository.com/**‘for Maven to download all the project dependency libraries. For any library required in the project, Maven first look in to the .m2 folder of Local Repository, if it does not find the required library then it looks in Central Repository and download the library in to local repository.

### Maven POM

***POM***is ***Project Object Model*** XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects. Some of the configuration that can be specified in the POM are the project dependencies, the plug-in or goals that can be executed, the build profiles, and so on.

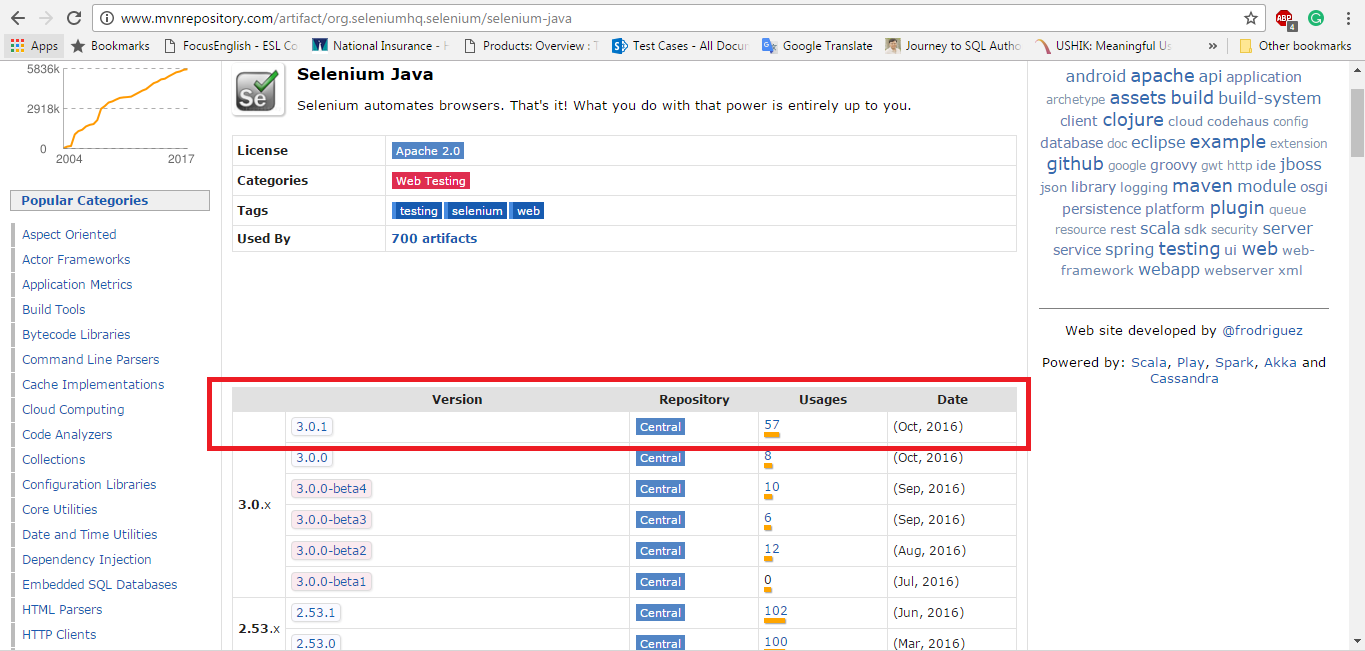
1) Go to <http://www.mvnrepository.com/> and search for Selenium Java.



Click on the ***Selenium Java*** from the search results.



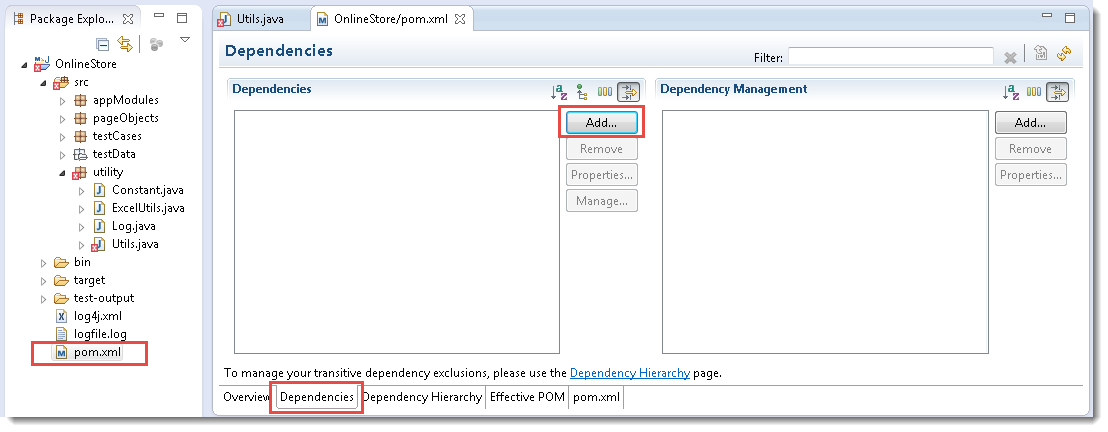
3) Click on the latest version of the Selenium. You are allowed to choose any version here if in case it is specific to your project.



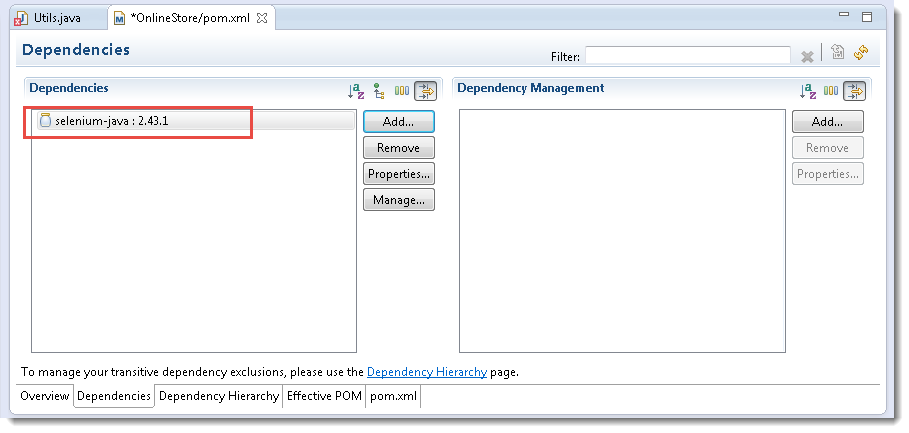
4) Take a look at the highlighted area on the below image. This highlighted details needs to be entered in the Maven **pom.xml**. There are two different ways of adding dependencies to pom file. First we will go through the easy way.

.5) Double click on the **pom.xml** file which is at the bottom of the project explorer window.

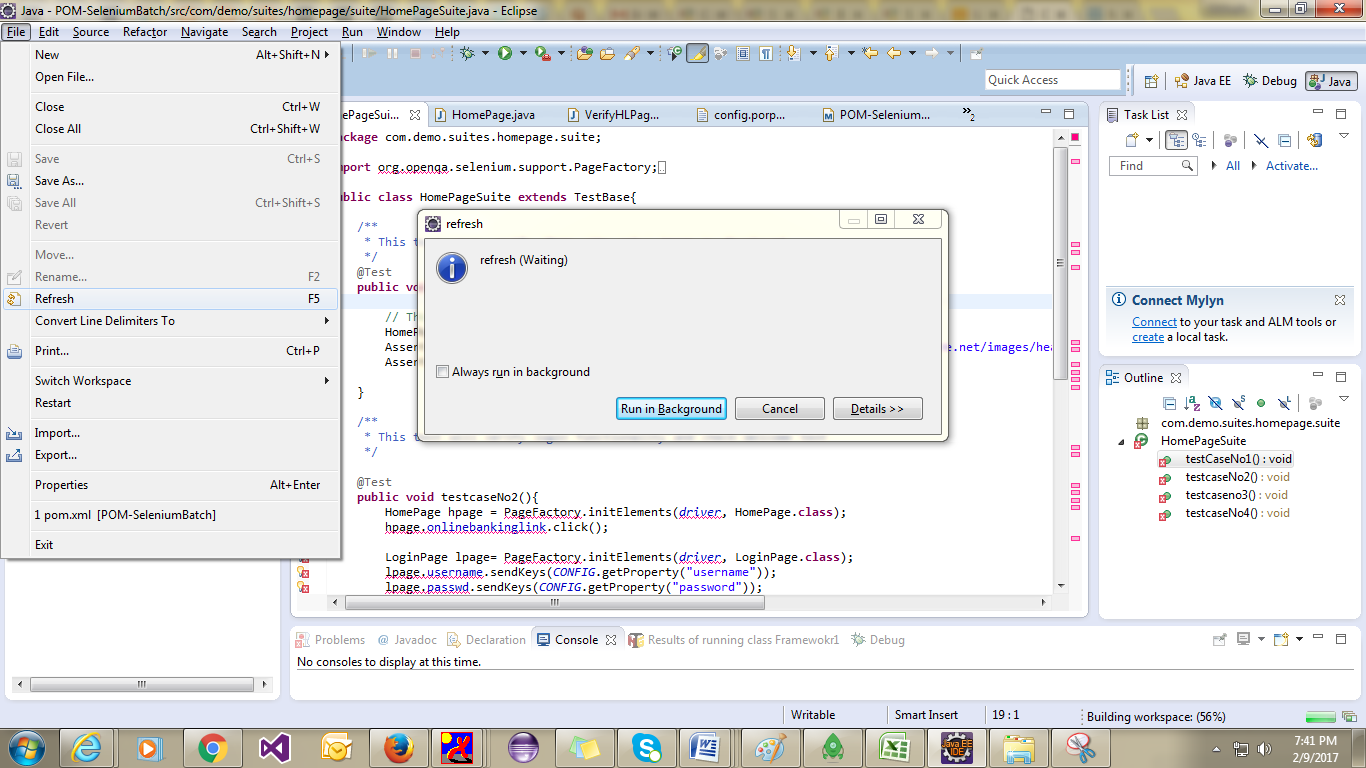
1. From Eclipse Select the ***Dependencies***tab and then click on ***Add*** button.

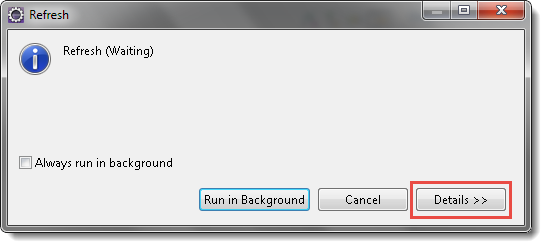


1. A pop window will display. Enter the highlighted detail here which we copied from maven repository website, as per the below screen shot. Make sure that ***Group ID, Artifact ID & Version*** are exactly the same. It is better tocopy paste it, as I am not sure if it is case sensitive.
2. Now the Dependencies tab will look like this with Selenium Java added on it. It is the time to Save the changes, that can be done by pressing ***Ctrl + S***.

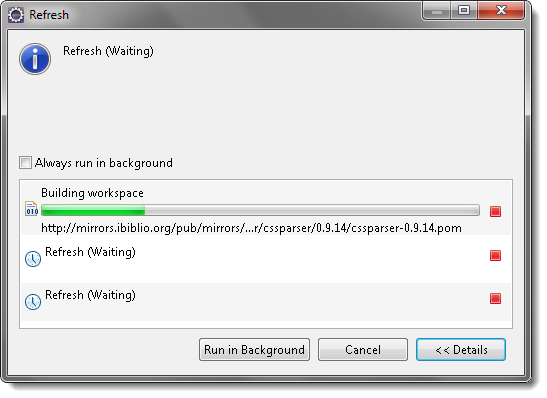


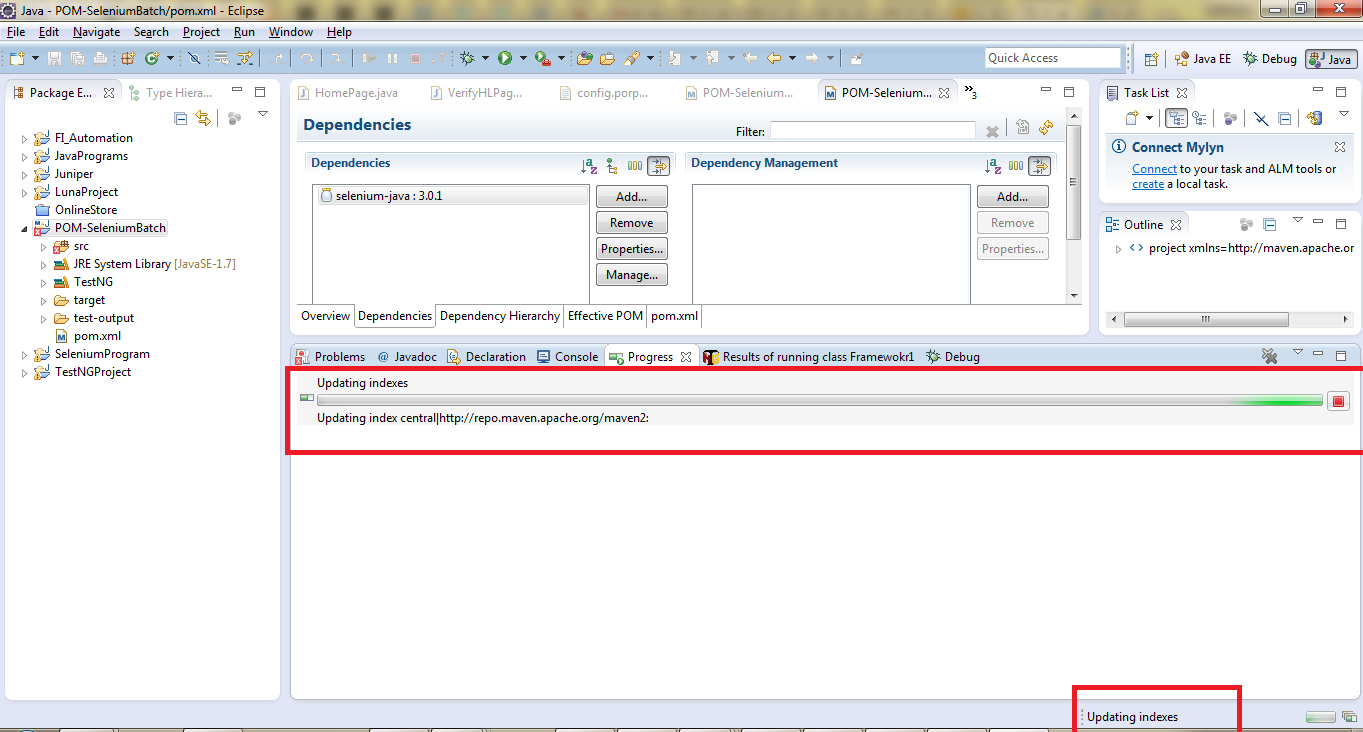
1. Now once the changes have been saved, right click at the empty space of the **project explorer** and select***Refresh***.





If you choose ‘***Run in Background***‘, it will download the Selenium Java jar files in to the Maven dependencies and the downloading process will take place at the background. If you choose ‘***Details>>***‘, it will display the downloading status bar and the process.

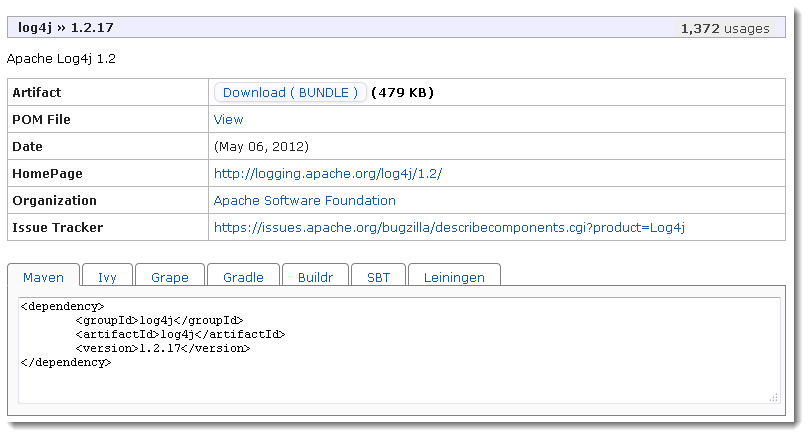


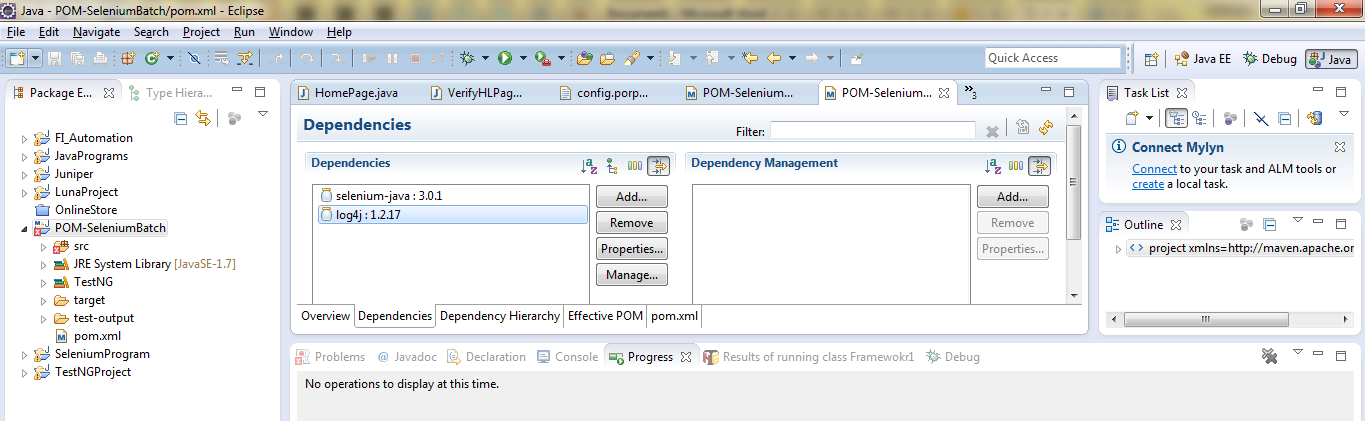


***Note:***Maven try to connect with Maven repository website and download the added dependency to the maven dependency folder. If in case you are in office or behind the firewall, you might need to perform some proxy changes to download it.

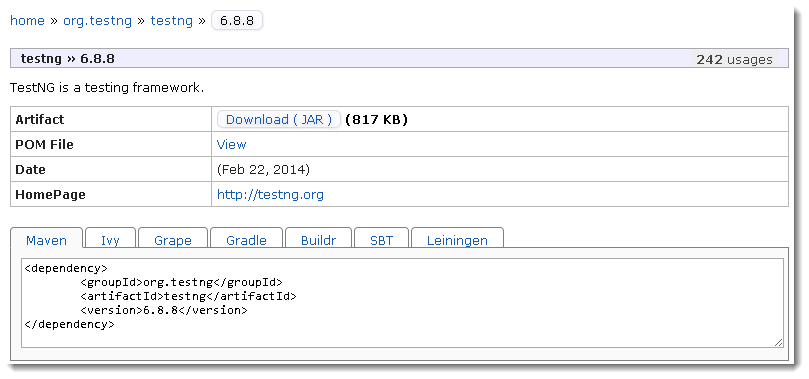
***Note:*** By default the folder name is ‘**.m2**‘ and by default the location in windows 7 is ‘**Libraries\Documents\.m2**‘ or ‘***C:\Users\yourusername\.m2***‘.

11) The same way it is required to add all the dependencies which are required in the project. Now add the ***Log4j***dependency in the POM file.

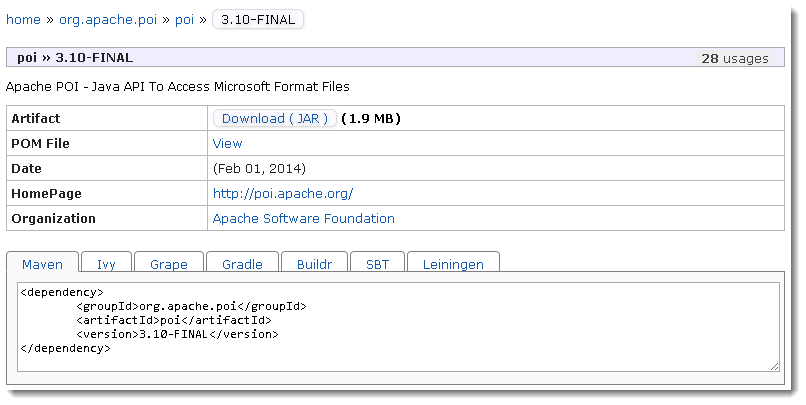




12) Notice the other errors in the project, you will notice that the TestNG is also required for the dummy project which we have used. Add ***TestNG***dependency to the POM file.

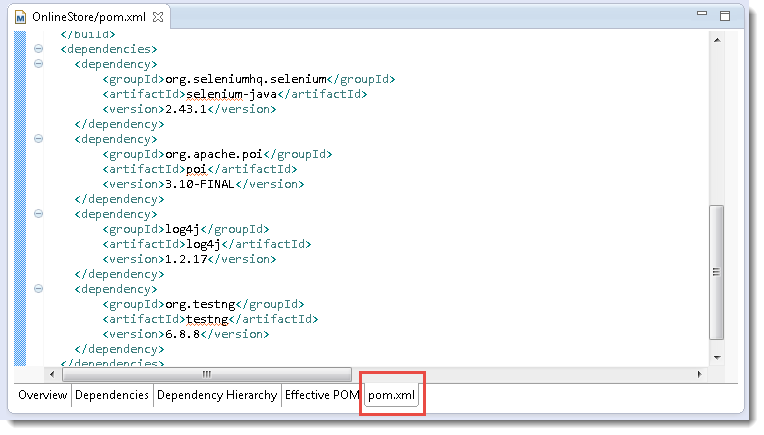


13) Go to **ExcelUtils**class under the ***utility***package. You will notice so may error on the Excel keywords. It is also required to add ***Apache POI*** jars to maven dependencies.

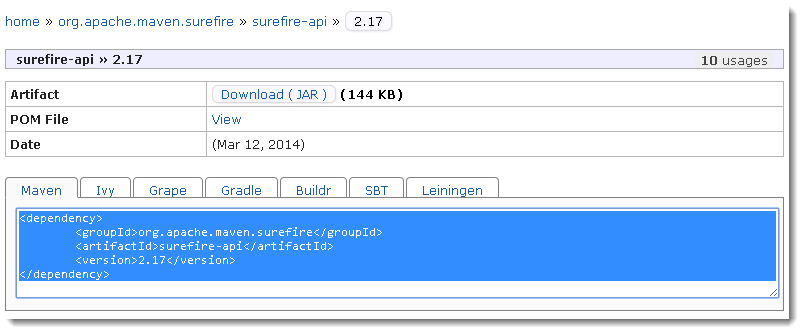


13) One more dependency is left which is called ***Surefire***. The Surefire Plugin is used during the test phase of the build life cycle to execute the unit tests of an application. It generates reports in 2 different file formats like plain text file, xml files and html files as well. Even if you are using TestNG or Junits framework for reporting, this plugin is must to use, as it helps Maven to identify tests.

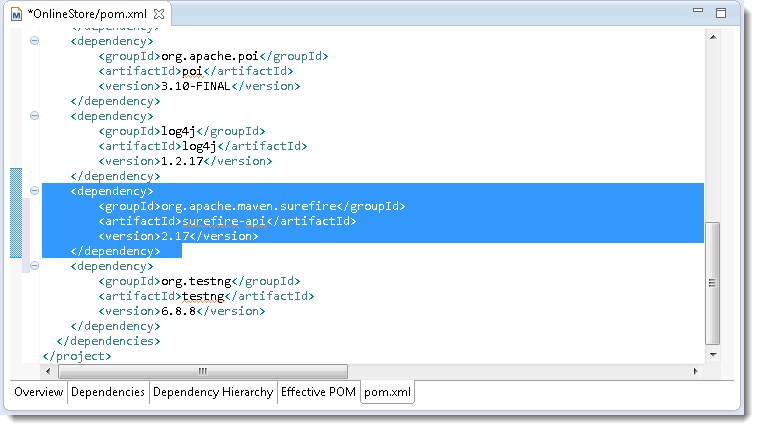
This time lets look at the second way of adding dependencies to the project. Click on the ***POM***at the bottom bar of the Eclipse view and it will open the ***xml***structure of the pom file.



14) Select the text from the Maven Repository website.



15) Paste the selected text in the pom file under the ‘***<dependencies>***‘ tag.



***Note:***I called this method difficult above, just because if you paste the dependency at the wrong place in the xml, it will create multiple errors, so by following the first method there is no chance of mistake.

Now we are done with adding all the dependencies to the project.

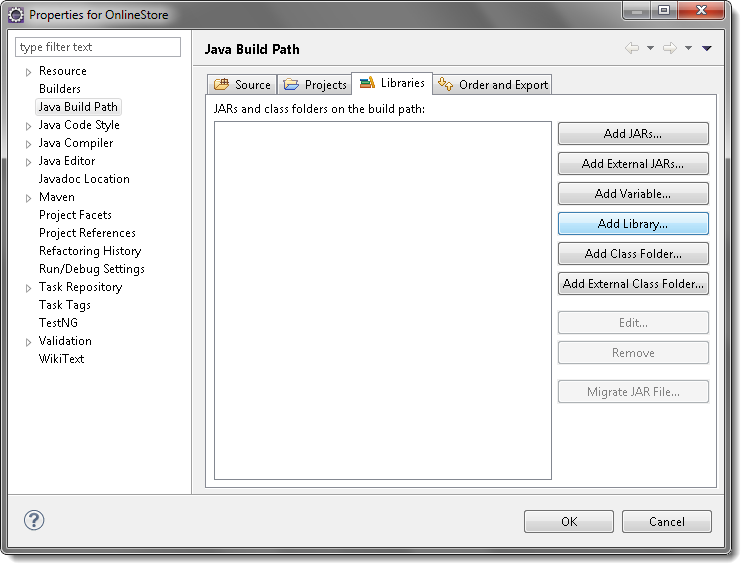
## Add JDK to the Project Build Path

The prerequisite of this step is the **JDK**should be installed on the computer. Please follow the [***Set Up Java***](http://toolsqa.wpengine.com/selenium-webdriver/download-and-install-java/) link to follow the step by step installation of JDK.

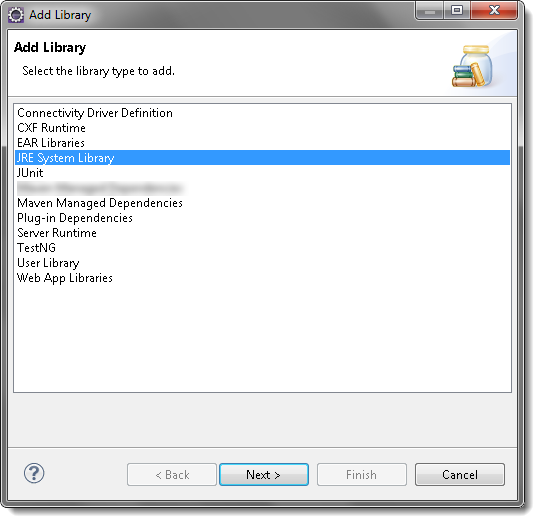
1) Right Click on the ***project explorer*** and select ***Build Path > Configure Build Path.***

2) Select the ***Libraries*** tab.

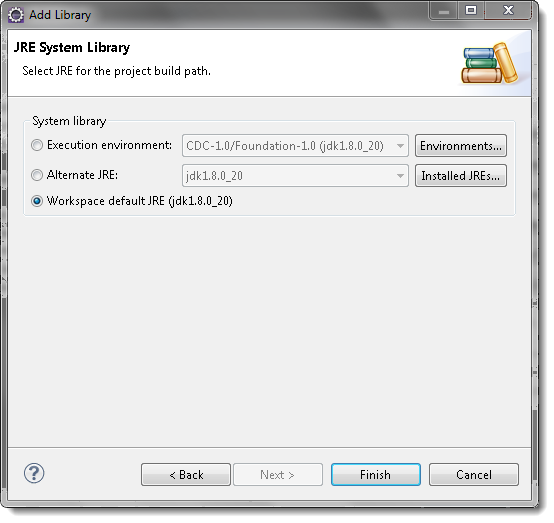
3) Click on ***Add Library..***.



4) Select ***JRE System Library***and click on Next button.



5) Select ‘***Workspace default JRE(jdk XXXXX)***‘ radio button and click on Finish button.



***Note:***JDK can be found also by clicking on the Installed JREs button.

## Add Maven Dependencies to the Project Build Path

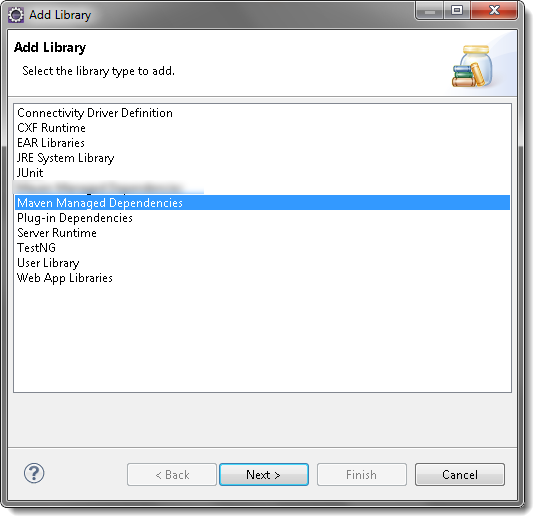
When we create a new Maven project, dependencies are added by default, so Maven Dependencies can be seen under the project explorer window. But when the Selenium project is convert to the Maven Project, maven dependencies needs to be added on to the project build path. So far we have just added the dependencies in the maven repository, now we need to link that repository to the project.

1) Right Click on the ***project explorer*** and select ***Build Path > Configure Build Path.***

2) Select the ***Libraries*** tab.

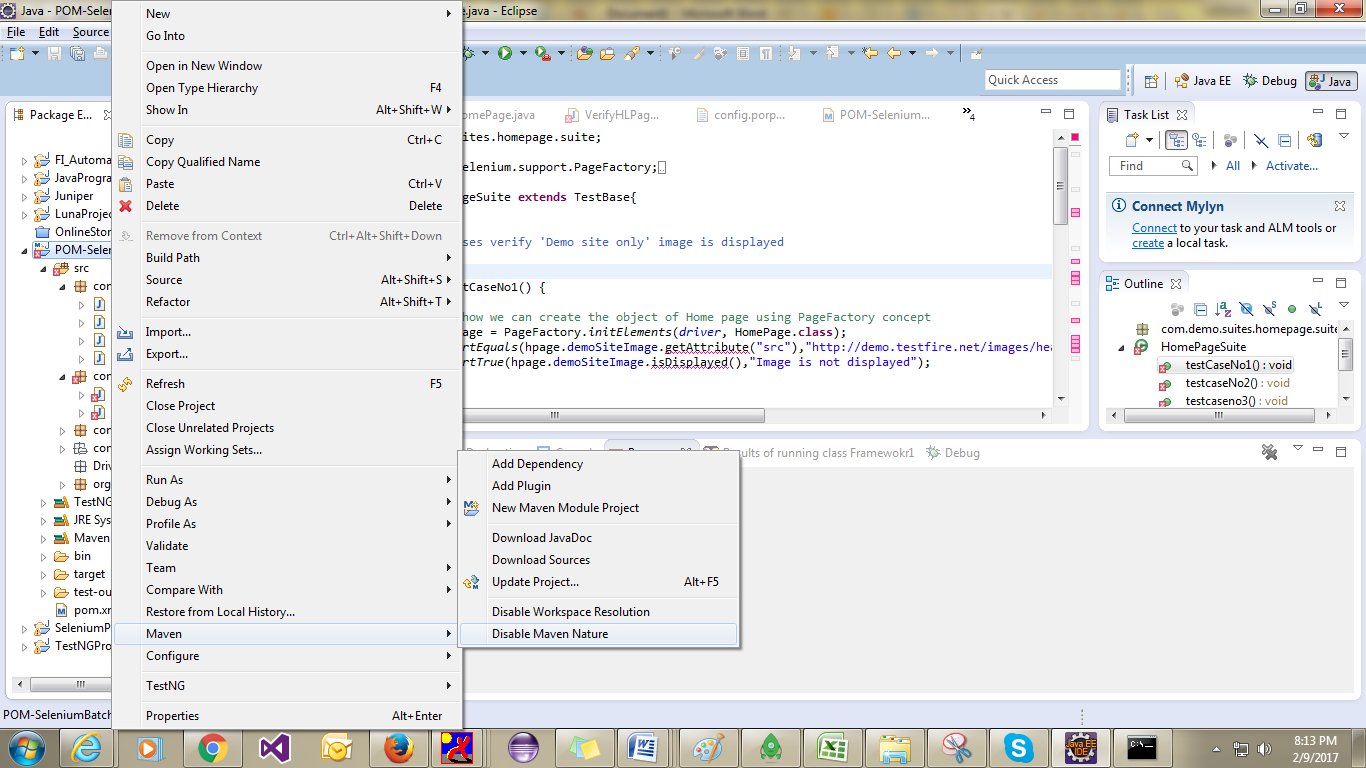
3) Click on ***Add Library..***.

4) Select ***Maven Managed Dependencies***and click on Next.



6) Take a look over the project explorer window, Maven Dependencies can be seen.

* M2Eclipse or m2e: Update the POM file (select, press *F5*), then right-click the project, *Maven > Update Dependencies*



1) Open the **.classpath** file at the root of your eclipse's project.

2) Insert the following entry to the file:

<classpathentry kind="con" path="org.eclipse.m2e.MAVEN2\_CLASSPATH\_CONTAINER">

<attributes>

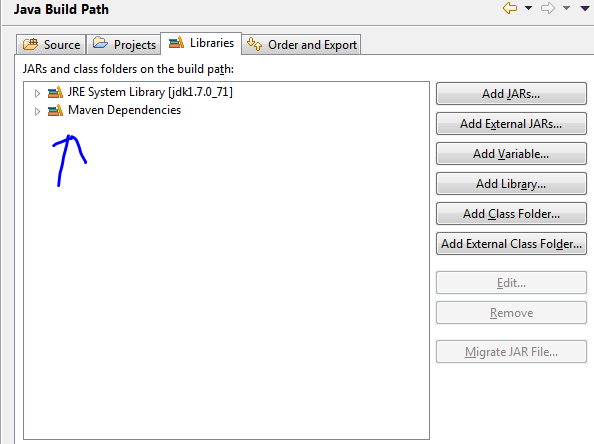
<attribute name="maven.pomderived" value="true"/>

<attribute name="org.eclipse.jst.component.nondependency" value=""/>

</attributes>

</classpathentry>

Then, rebuild your project at eclipse (Project->Clean-Build). You now can check at the Java Build Path of you project at the Libraries tab the Maven Dependencies included:



# [Configuring Integration Of Jenkins Reports](https://docs.experitest.com/display/SA/Configuring+Integration+Of+Jenkins+Reports)

As part of the SeeTestAutomation's integration with Jenkins, a post-build action is supported which enables loading the generated HTML execution reports.

Prequisities

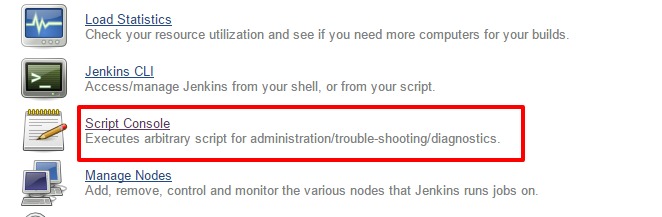
Jenkins from version 1.625 and later make use of *Content-Security-Policy* header, which by default set to very restrictive rules. In order to use Experitest plugin the default CSP rules have to be relaxed.

Unset Content-Security-Policy header

**Step 1**: In Jenkins Dashboard click on '**Manage Jenkins**'



**Step 2**: Click on '**Script Console**'



**Step 3**: Copy **System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")** into the Script box and click **Run**



**Step 4:** Clear Cache from your browser.

## Installation

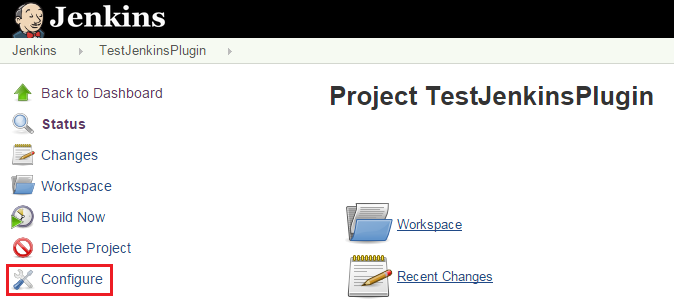
**Step 1:**Download the**Experitest\_ReportLinks.hpi** file [HERE](https://s3.amazonaws.com/experitest/Experitest_ReportLinks.hpi).

**Step 2:**After downloading the ***.hpi*** file, place it in your Jenkins plugins folder (directly under the root of the Jenkins installation folder, which is usually -C:\Program Files (x86)\Jenkins\plugins).

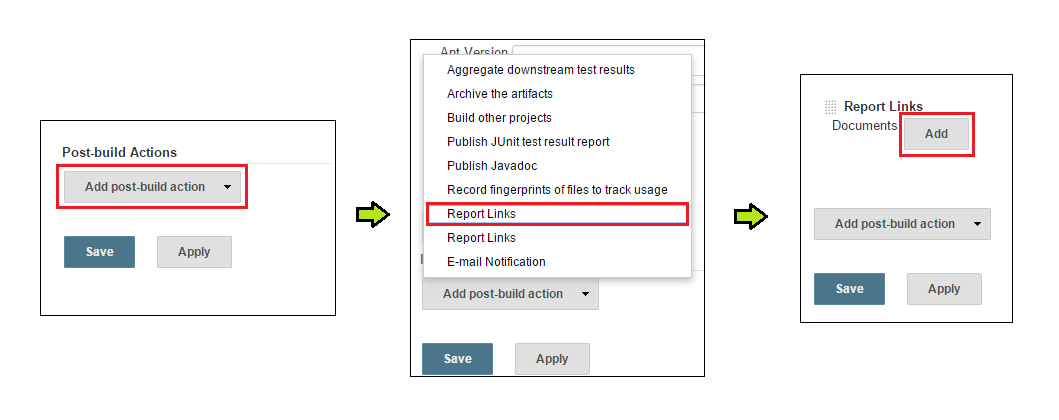
**Step 3:**Restart your Jenkins server.

Configuring Your Build

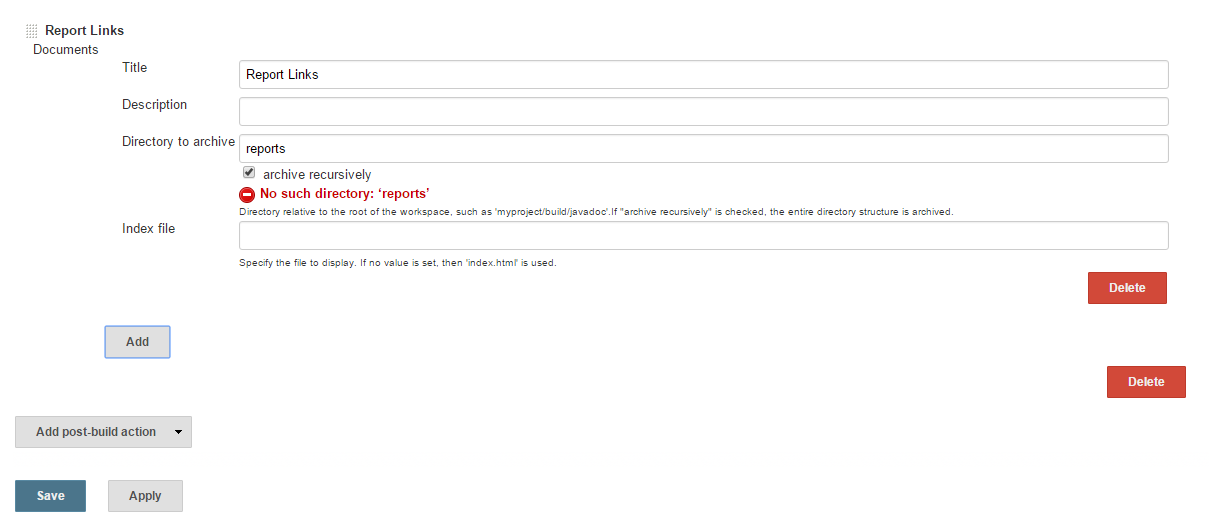
**Step 1:**In your project tab, in the Jenkins portal, click on **'Configure'**.



**Step 2:**Click on **'Add post-build action'**, select **'Report Links'** (choose the first option) and click on **'Add'**.



**Step 3:** The following configuration dialog box is now displayed. Make sure to place the build.xml at the same level as the src folder in your Java project.



* The 'report' folder does not exist in the project at this point, that is why the 'No such directory' error is displayed. The first report's generation invokes the folder generation.

**Step 4:**When entering details into this dialog box, pay particular attention when designating the directory to archive. This directory is relative to the tests project home directory, meaning that you will need to generate the reports to this directory. For example, below is a such a designation in Java:

@Before

public void setUp(){

    client = new Client(host, port, true);

    client.setProjectBaseDirectory(projectBaseDirectory);

    File reports = newFile(System.getProperty("user.dir"), "reports");

    reports.mkdirs();

    client.setReporter("xml", reports.getAbsolutePath(), "<Your\_Test\_Name>");

}

* In this example, the reports directory is set to the project home directory by contacting the *user.dir* with the reports directory.

**Step 5:**Run SeeTestAutomation as an administrator**.**

**Step 6:**Schedule a build though Jenkins.

**Step 7:**After executing the project, the summery report will be available as part of your Jenkins project tab under - **'Report Links'**.

