Spring-Maven step by step tutorial **Date:** 25June2014

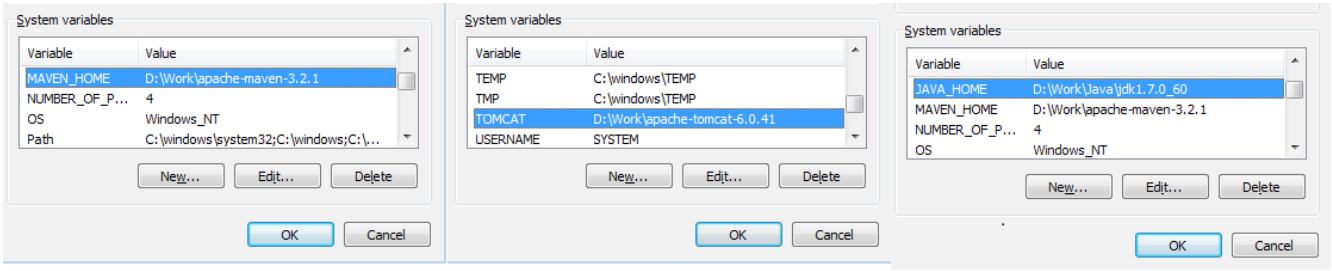
Reference: **Create Spring-Maven Project**

Requirements:

1. JDK7
2. Spring 3.2
3. Maven 3.2
4. Eclipse Kepler
5. Apache Tomcat 6
6. Create a work directory in D-Drive and name it as **work**
7. Install maven in eclipse (m2e)

Step -1:

Download jdk7 and install. After installing goto the C drive and copy the installed Java folder to work folder. Set the JDK7 path to system variable by giving name as **JAVA_HOME,** download maven 3.2 and Tomcat 6 and place them inside work folder. Also give the path of these in the system variables.

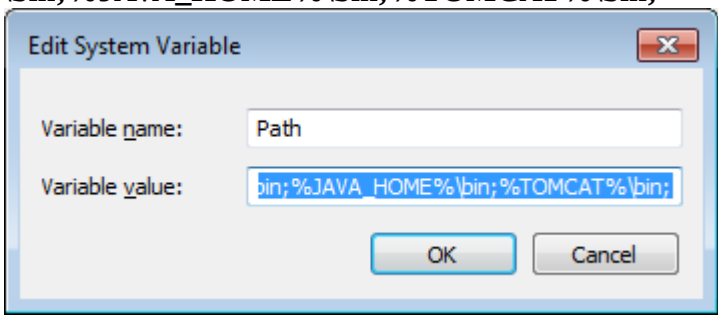


Step – 2:

Inorder to make the above Environmental variables available we need to mention them in the **path** system variable.

So open the system variable **path** and append it with,

%MAVEN_HOME%\bin;%JAVA_HOME%\bin;%TOMCAT%\bin;

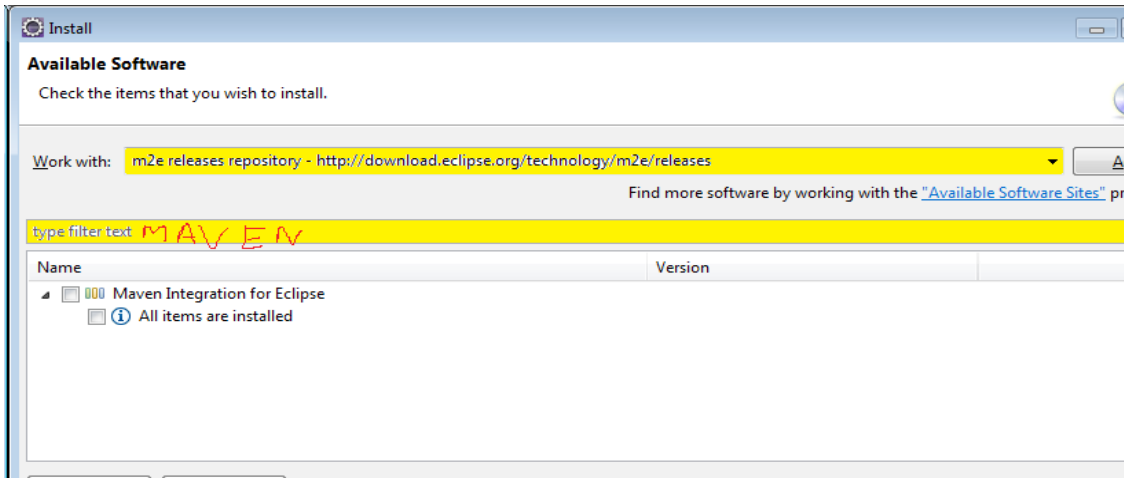


You can check by running, **mvn or javac or java** on command prompt.

Step – 3:

Installing **Maven** in eclipse.

Eclipse → Help → Install new software → and give the URL as **m2e releases repository - http://download.eclipse.org/technology/m2e/releases** and in search type **maven.** You will be seeing the **Maven Integration for Eclipse** select it and install.



Step - 4:

So we are all set with all the installations and classpaths.

Now create a workspace for eclipse. Say **Code**.

Goto Code workspace and run the following command,

```
> mvn archetype:generate -DgroupId=com.javacodegeeks
-DartifactId=SampleWebApplication -DarchetypeArtifactId=maven-archetype-webapp
-DinteractiveMode=false
```

This will create a directory and you can see the project structure as below.

about:blank

```
SampleWebApplication
|-- pom.xml
`-- src
    |-- main
    |   `-- resources
    |    `-- webapp
    |           `-- index.jsp
    |   `-- WEB-INF
    |         `-- web.xml
```

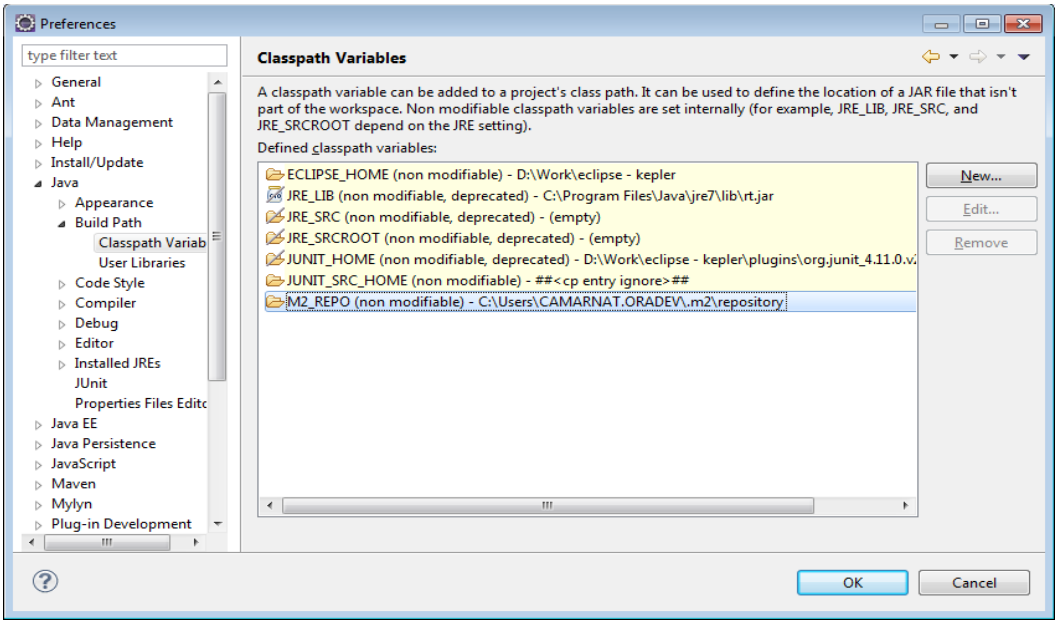
You will have **pom.xml**, **web.xml**, **index.jsp** files generated.

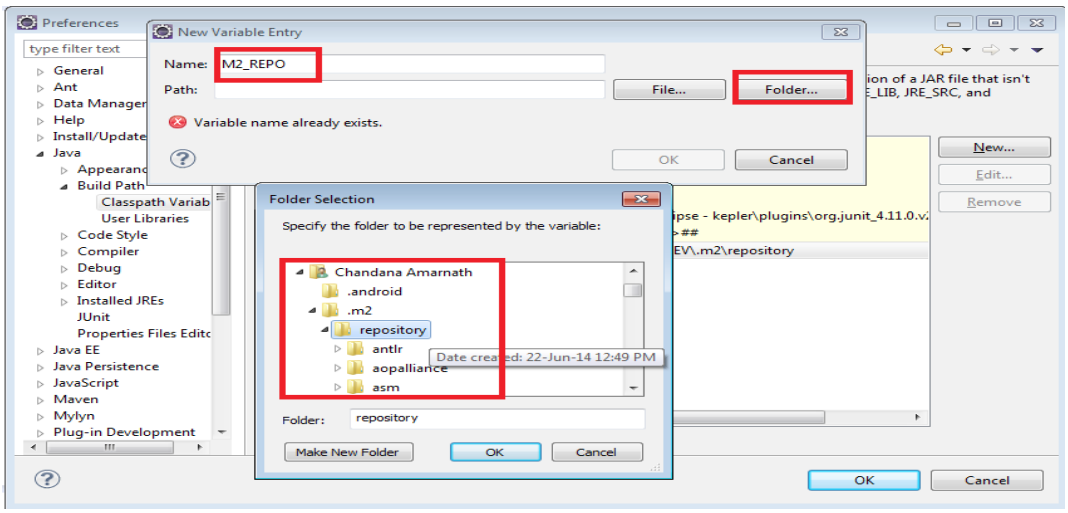
Step – 5:

Enrich the **pom.xml** file. The files generated are rather outdated so add Spring dependencies, compiler plugin for knowing that jdk7 has to be used, Junit later version.
Similarly add update the **web.xml file**.
You can check both the above file code in the SampleWebApplication project.
Note: Servlet dependency is missing in the reference site for pom.xml which we have added in the SampleWebApplication's pom.xml

Step-6:

Add **M2_REPO** to the eclipse IDE inorder to locate the dependencies inside the maven project.
Eclipse → Windows → Preferacne → on left click Java → Build path → Classpath Variable → on right click **New** → Give name as **M2_REPO** and click on **Folder** and select repository inside .m2. You will find .m2 at **C Drive → users → .m2**

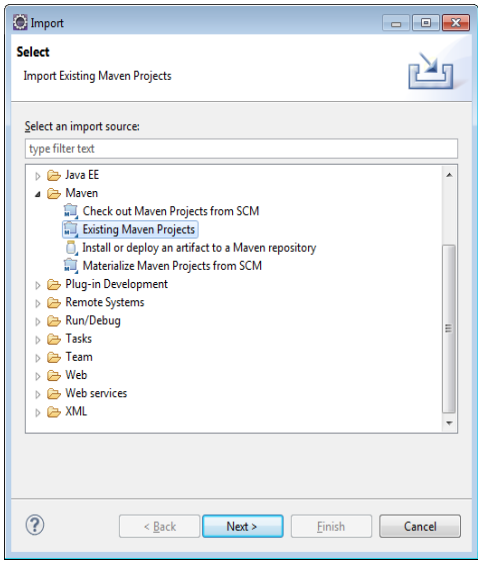




Step – 7:

Import this project into Eclipse.

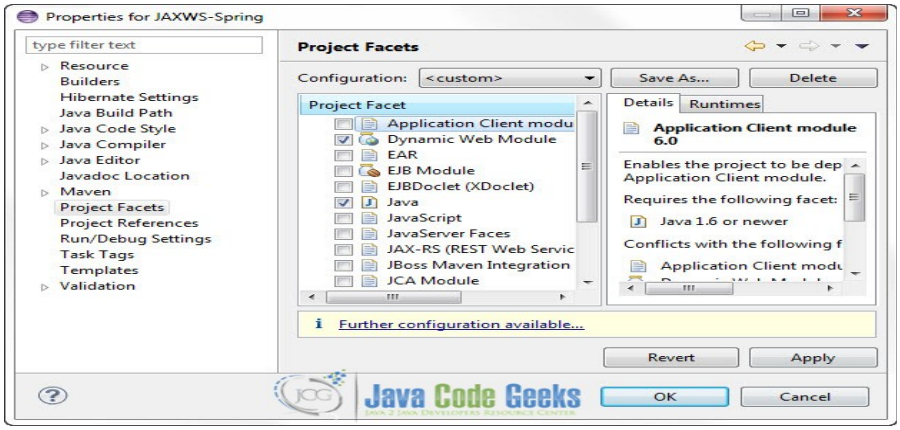
Eclipse → Import → Maven → Existing maven Project → Browse and select the project.



During this import sometimes I got few errors by maven. When you get these just delete the .m2 repository and then do the import again.

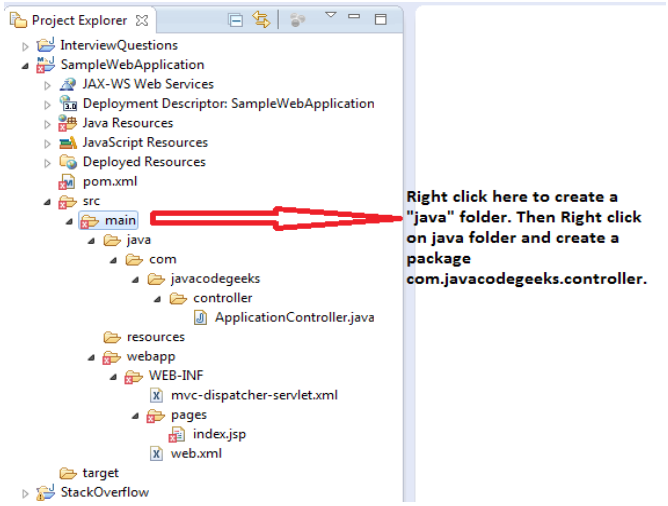
Step – 8:

Now after import **Right Click on Project → Properties → Project Facts → Check Dynamic Web Project and Java → OK.**



Step – 9:

Create a controller class inside the "/src/main/java/com/javacodegeeks/controller".
In order to do this, first we need to create fodler **java** in the src/main. Then create a package with name as **com.javacodegeeks.controller**.
Copy the java code which is attached in the project to this class.



Step – 10:

Update web.xml using the code attached in the SampleWebApplication project. In this code we have a tag called **<servlet-name>** which is having **mvc-dispatcher**. Spring will append **servlet** to this name and search for the file mvc-dispatcher-servlet.xml.
This is given wrong in the reference site.

Step – 10:

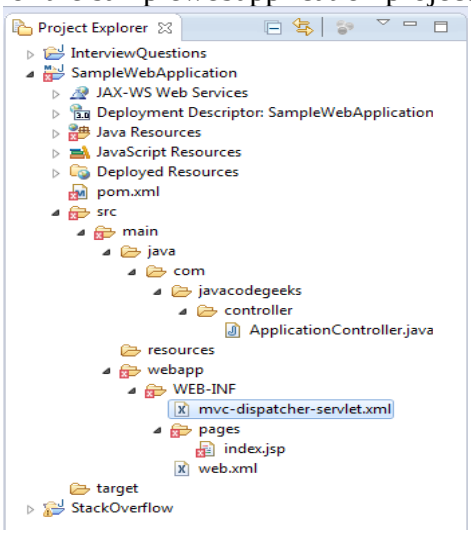
Create a servlet with name **mvc-dispatcher-servlet.xml** inside "**/src/main/webapp/WEB-INF/"**
Note: Very important part of the Spring framework. Spring frame work will search for this class by using the name given in web.xml.

Step – 11:

Create a folder **pages** inside **WEB-INF,** and move **index.jsp** to this folder.
For the content of index.jsp check the SampleWebApplication project.

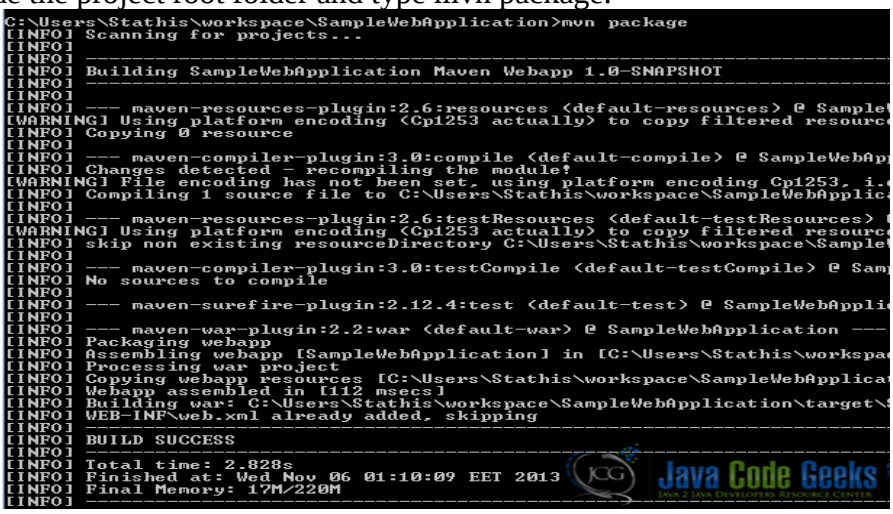
Step – 12:

Below is the final folder structure of the samplewebapplication project.



Step – 13:

Finally goto inside the project root folder and type mvn package.



Step – 14:

When we do mvn package a war file is generated inside target folder which is inside the project root folder. **Workspace → SampleWebApplication → target → .war**

Step – 15:

Finally copy this to Tomcat → Webapps.

And start the tomcat using start.bat which is inside the bin folder. When you start please check in the tomcat logs whether there is any error or not.

Step – 16:

You can run using the following URL: localhost:8080/SampleWebApplication/Test

Final Output: