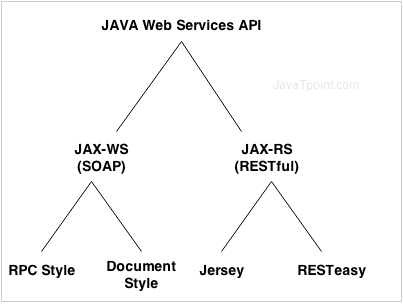
Java RESTful Web Services (JAX-RS) Jersey

Java Web Services API

There are two main API's defined by Java for developing web service applications since JavaEE 6.

1) **JAX-WS**: for SOAP web services. The are two ways to write JAX-WS application code: by RPC style and Document style.

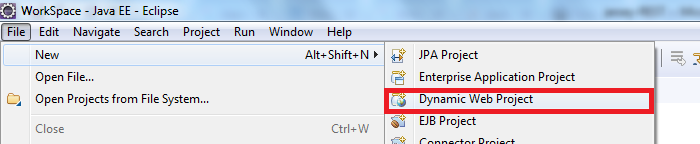
2) **JAX-RS**: for RESTful web services. There are mainly 2 implementation currently in use for creating JAX-RS application: Jersey and RESTeasy.

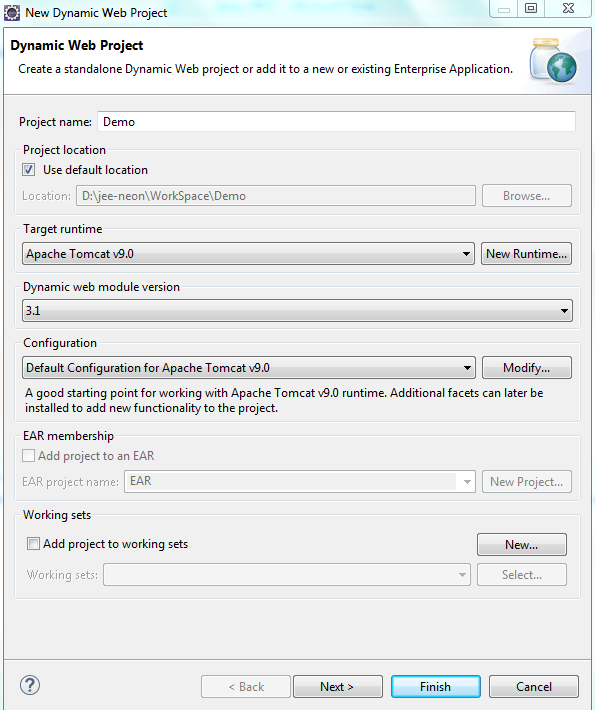


We are going to look at Jersey.

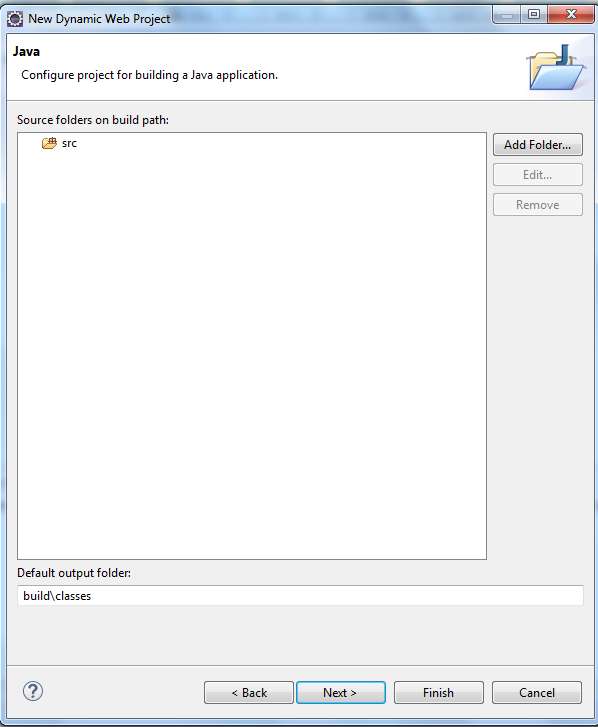
## Creating a Sample RESTful application in Eclipse

* Open eclipse .
* File -> New - > Dynamic Web project .

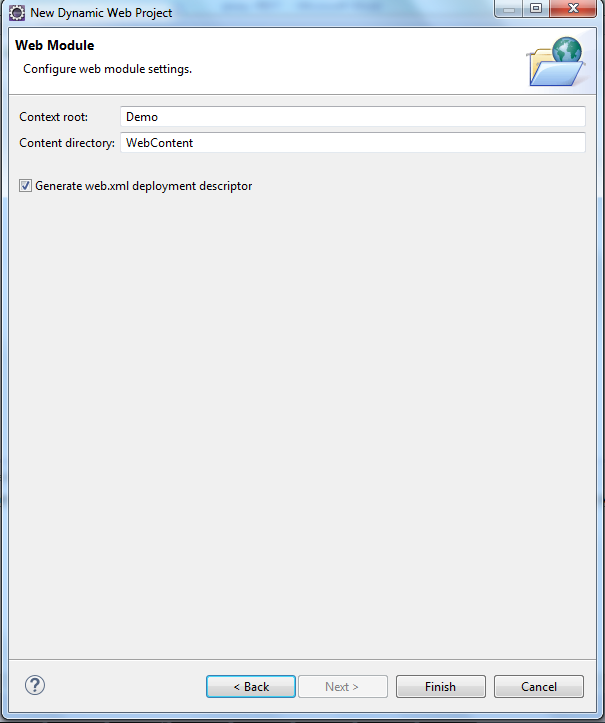




Next.



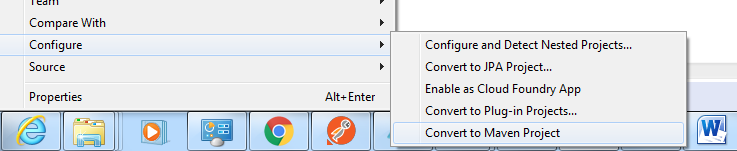
Next.

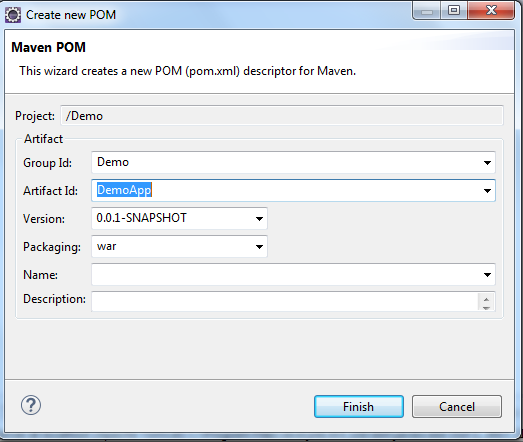


Enable Check Box Generate Web.xml . Finish .

## Converting project to maven project

Right click Project (Demo) -> Configure - > Convert to maven project.

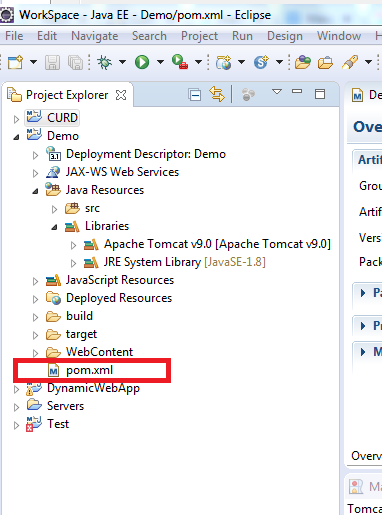




Give Artifact id and finish.

## Working on pom.xml

Open pom.xml



Add following lines of code after packaging tag .

<!-- https://mvnrepository.com/artifact/com.sun.jersey/jersey-servlet -->

<dependencies>

<dependency>

<groupId>com.sun.jersey</groupId>

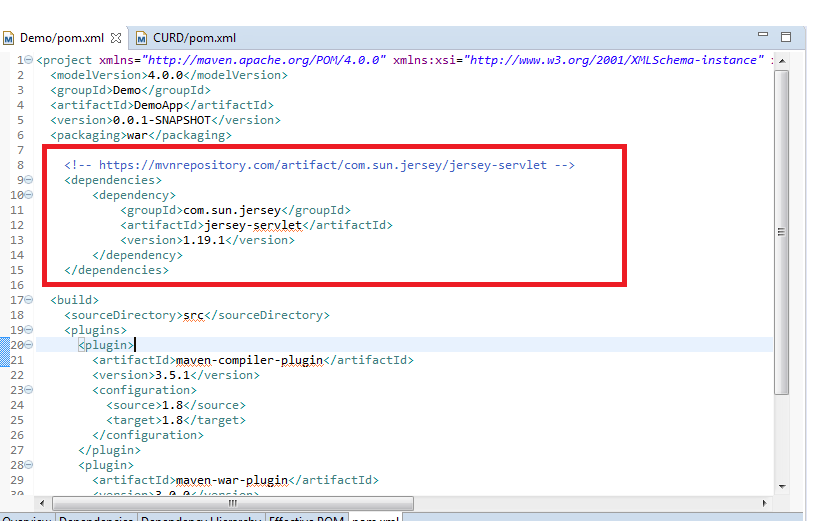
<artifactId>jersey-servlet</artifactId>

<version>1.19.1</version>

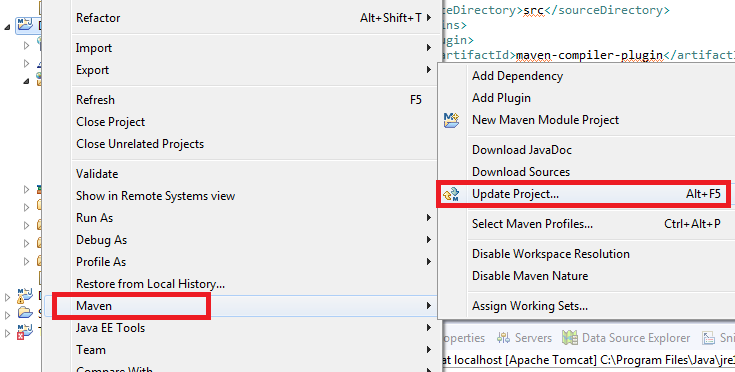
</dependency>

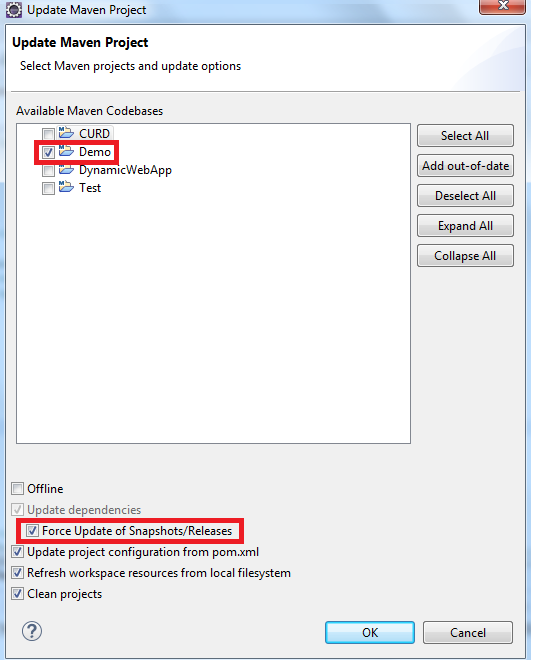
</dependencies>

Save the file .



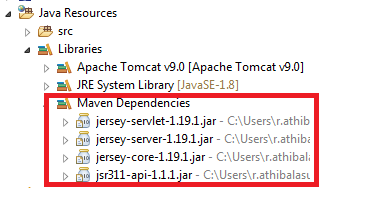
Right click Project -> Maven -> Update Project .





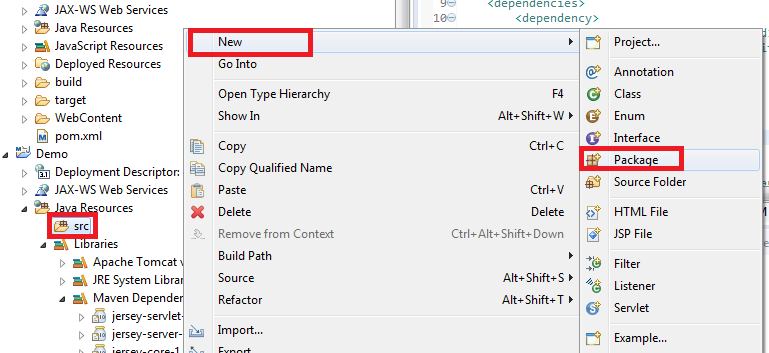
Ok.

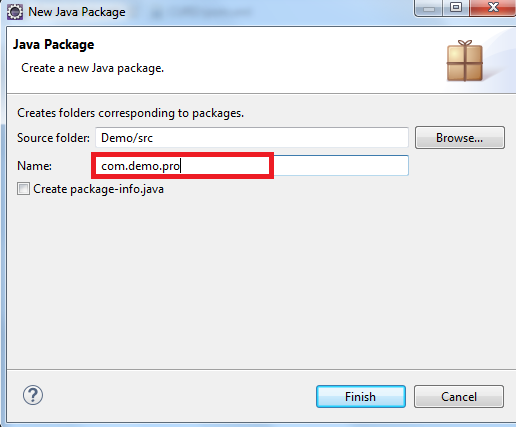
After this maven update,Maven Dependencies is added.



Right click Src -> new -> package .

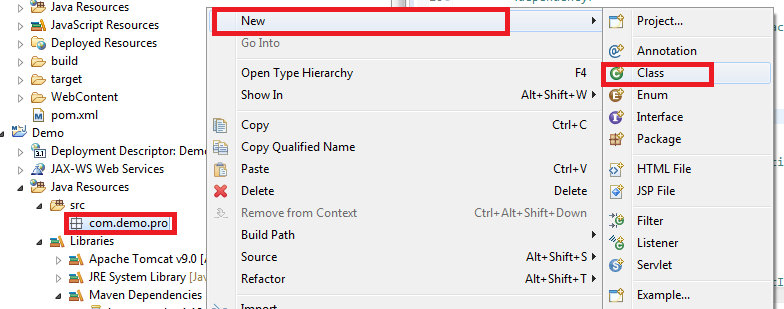
## Creating new package and classes to test Restful services .

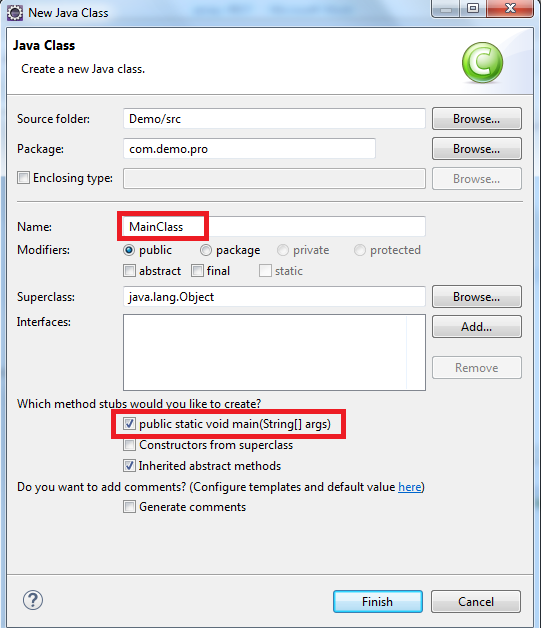




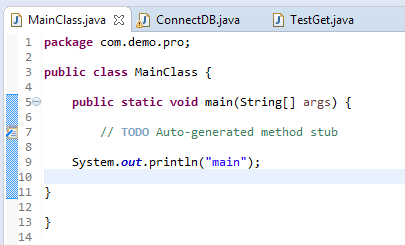
Finish .

Right click Package -> new -> Class .

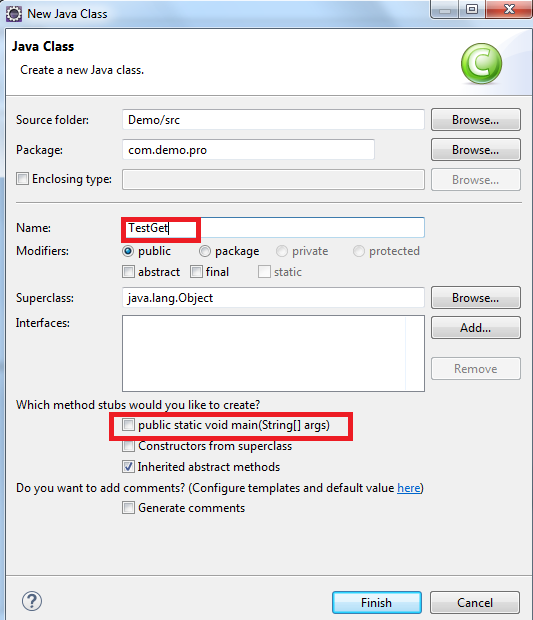




Finish .

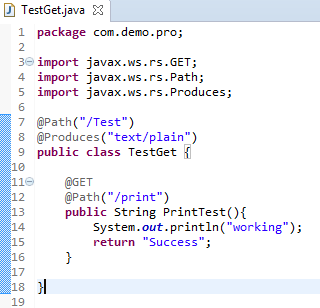


Create one more class . Right click Package -> new -> Class .



Uncheck public static void main.

Write below code in TestGet.java



**package** com.demo.pro;

**import** javax.ws.rs.GET;

**import** javax.ws.rs.Path;

**import** javax.ws.rs.Produces;

@Path("/Test")

@Produces("text/plain")

**public** **class** TestGet {

@GET

@Path("/print")

**public** String PrintTest(){

System.***out***.println("working");

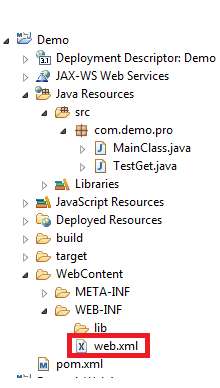
**return** "Success";

}

}

## Edit web.xml

Open web.xml .



Add following code below like mentioned in below screenshot.

<display-name>Archetype Created Web Application</display-name>

<servlet>

<servlet-name>jersey-servlet</servlet-name>

<servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>

<init-param>

<param-name>com.sun.jersey.config.property.packages</param-name>

<param-value>com.demo.pro</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

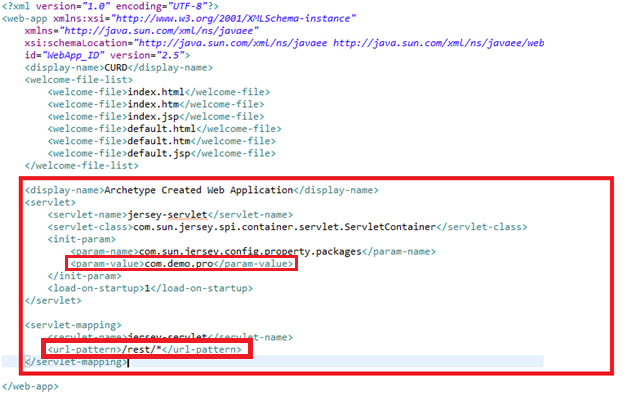
</servlet>

<servlet-mapping>

<servlet-name>jersey-servlet</servlet-name>

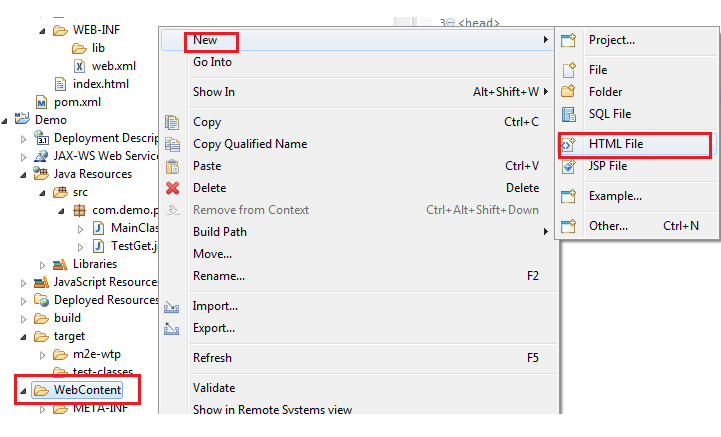
<url-pattern>/rest/\*</url-pattern>

</servlet-mapping>



## Create Index.html

Create index.html .



<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

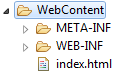
</head>

<body>

<p>hello</p>

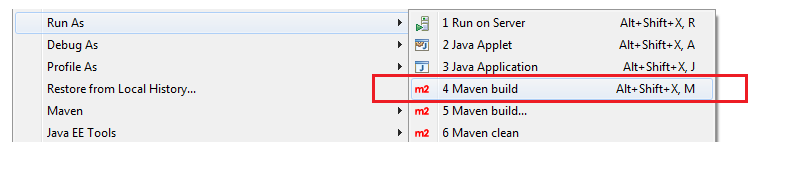
</body>

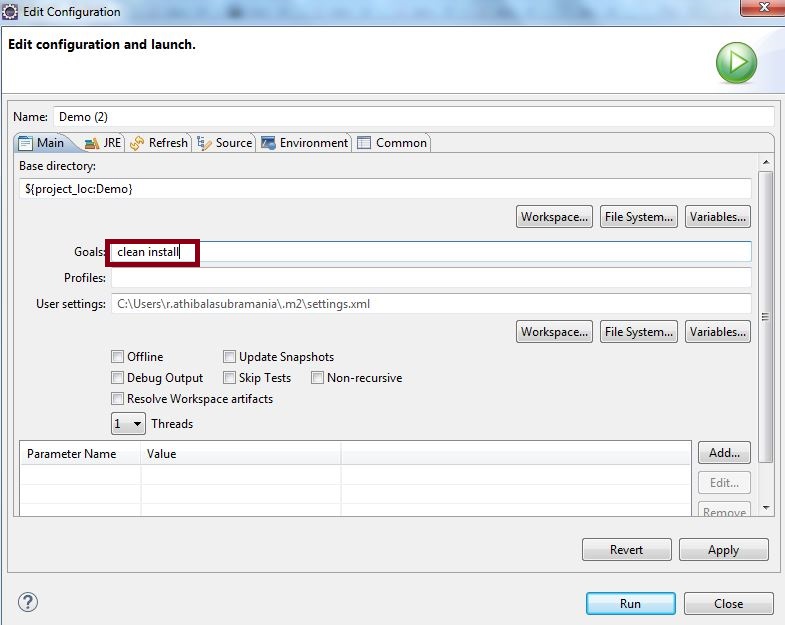
</html>



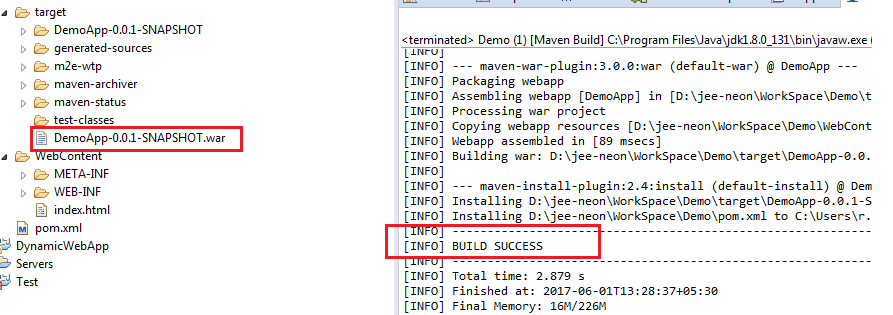
## Maven Build

Maven build .





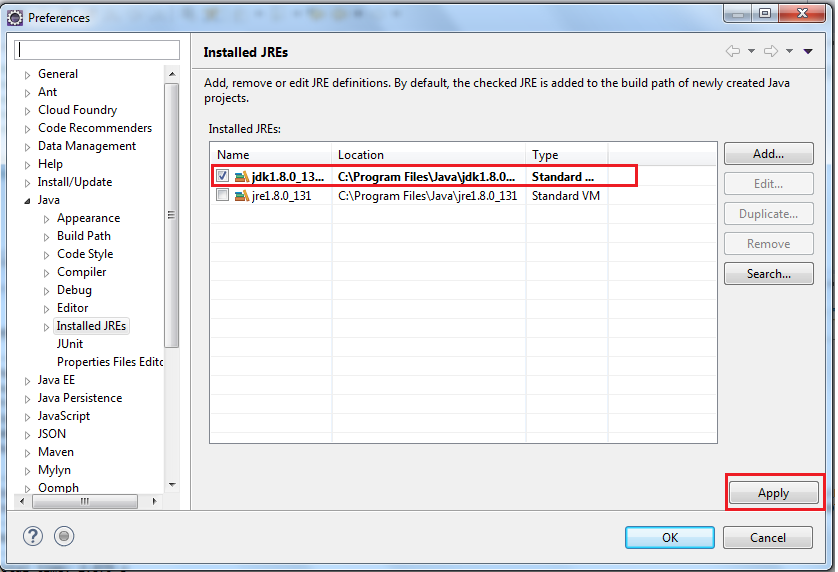
In the goals we have to give Clean install.

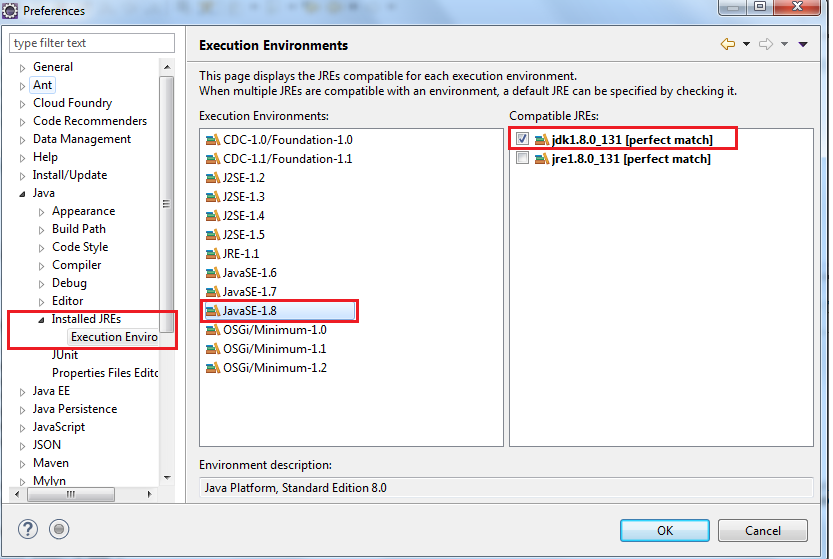


By **clean** statement ,target folder will be cleaned demoApp-0.0.1-SNAPSHOT.war will be deleted .By **install** statement again demoApp-0.0.1-SNAPSHOT.war will be generated .So to update java code changes to war file we need to build maven .Build log should show **success** like above screenshot.

Check the setting to assure required java selected .

Window -> preferences .

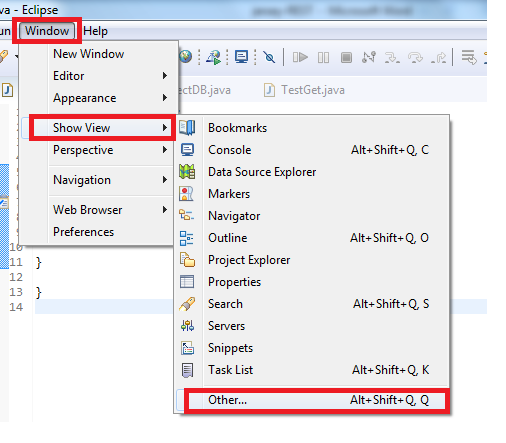


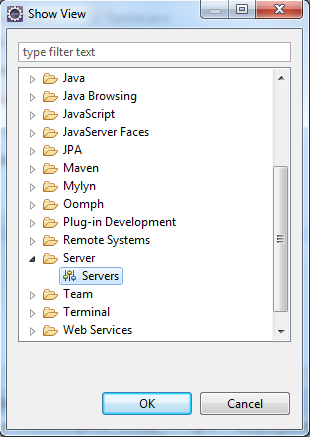


## Run on server

Extract Apache tomcat server zip file to any of your directory .

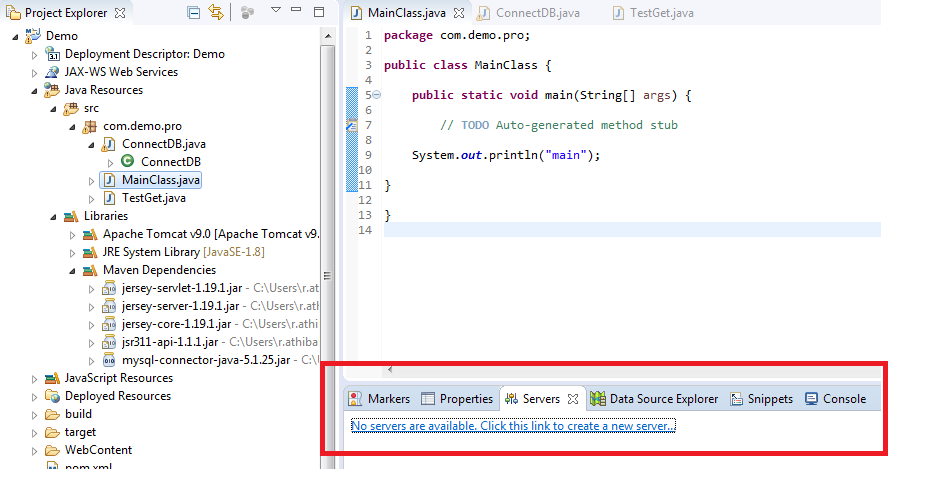
To configure apache tomcat server in eclipse . Windows -> Show view -> others .



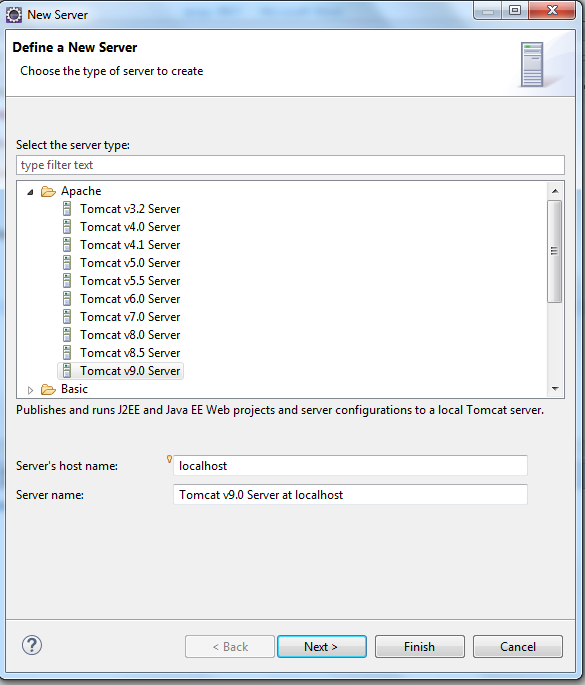


Okay .

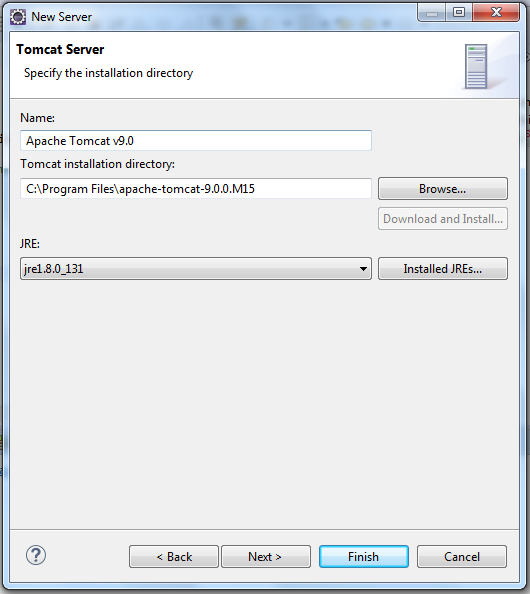
Server tab will added like below .



Click on No server are available in above screenshot.



Next .



Finish .

Run on server .

