Contents

[1. Create new Maven Project 2](#_Toc446329290)

[2. Create a new Tomcat Server 6](#_Toc446329291)

[3. Getting rid of javax.servlet.http.HttpServlet jsp problem 20](#_Toc446329292)

[3.1. Method A 21](#_Toc446329293)

[3.2. Method B 22](#_Toc446329294)

[4. Configuring our maven project 23](#_Toc446329295)

[4.1. Configuring pom.xml 23](#_Toc446329296)

[4.2. Configuring web.xml 23](#_Toc446329297)

[4.3. Adding our beans configuration file 23](#_Toc446329298)

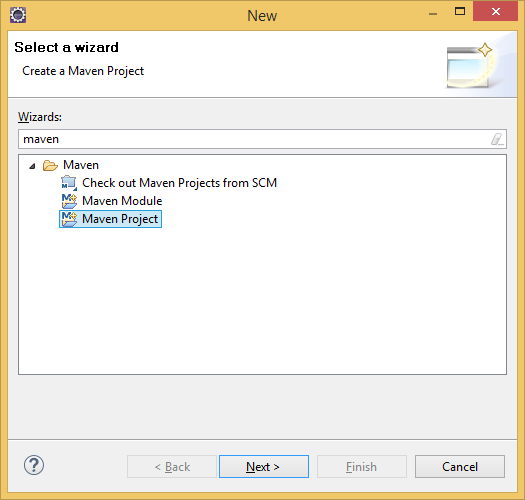
[4.4. Adding source (java) folder 24](#_Toc446329299)

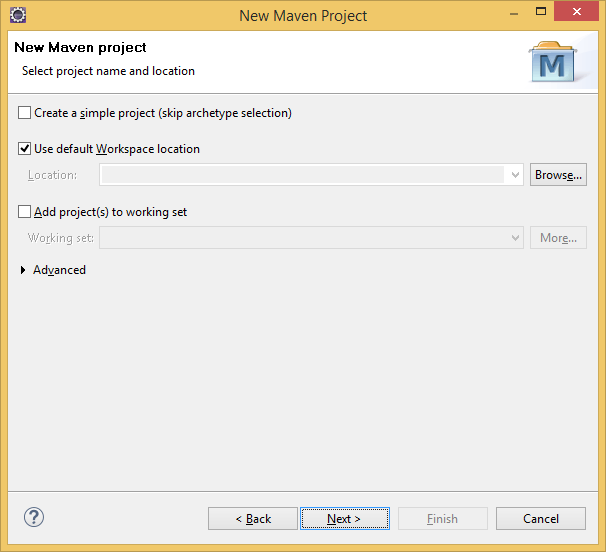
[4.5. Add your first package and first class 24](#_Toc446329300)

[4.6. Publish your application to the server 26](#_Toc446329301)

[5. Import a maven project 27](#_Toc446329302)

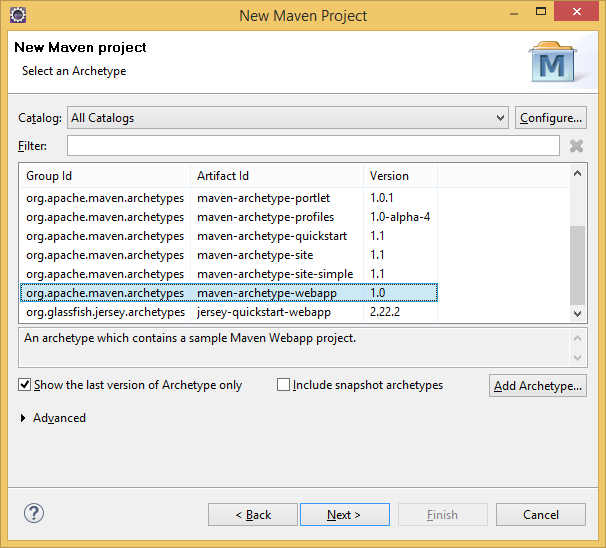
# Create new Maven Project



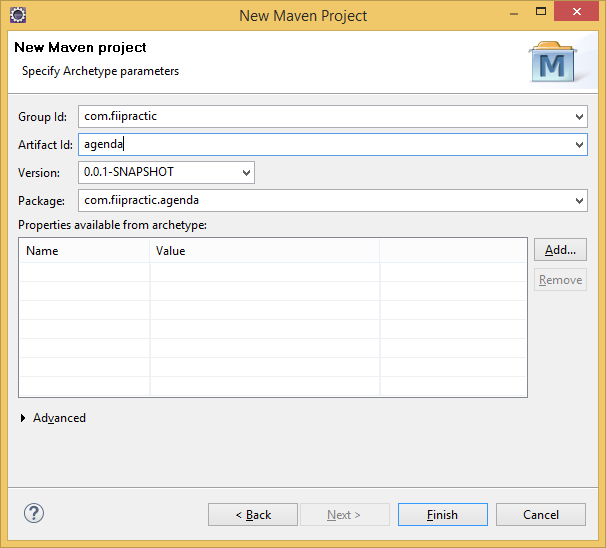
Next  


Next

Select artifact Id: maven-archetype-webapp (version 1.0)



Fill project details:

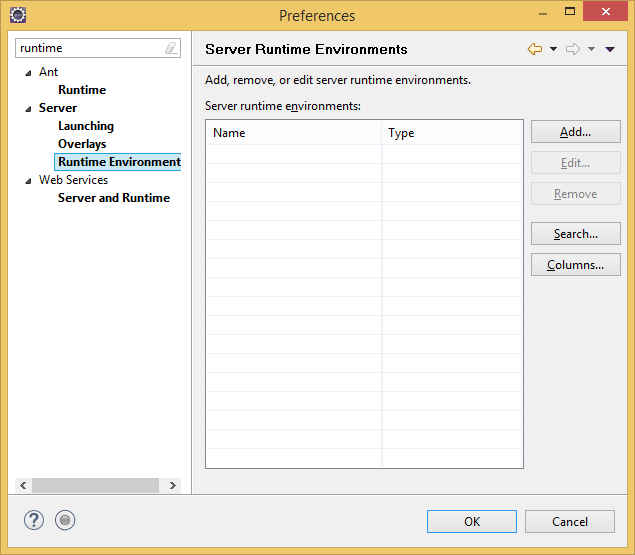


Group Id: com.fiipractic  
Artefact Id: agenda

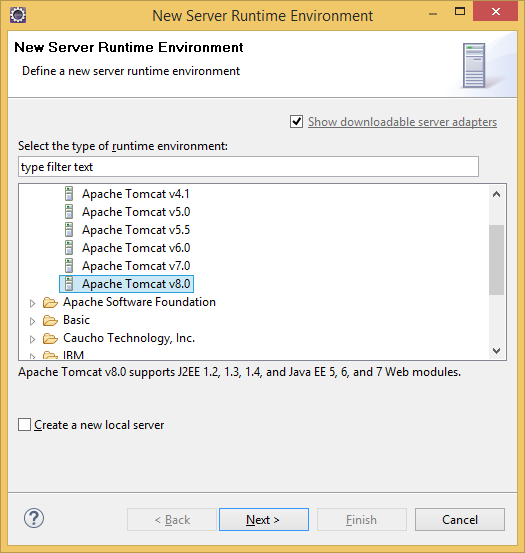
And click Finish ☺

# Create a new Tomcat Server

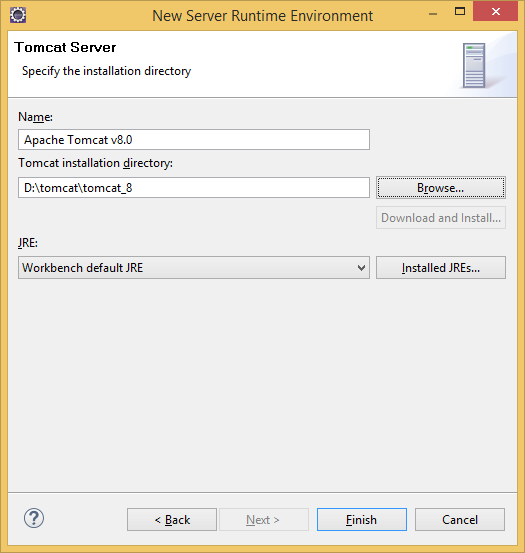
Go to Window -> Preferences and write runtime in ‘type filter text’ box, and go to Server -> Runtime Environment as below:



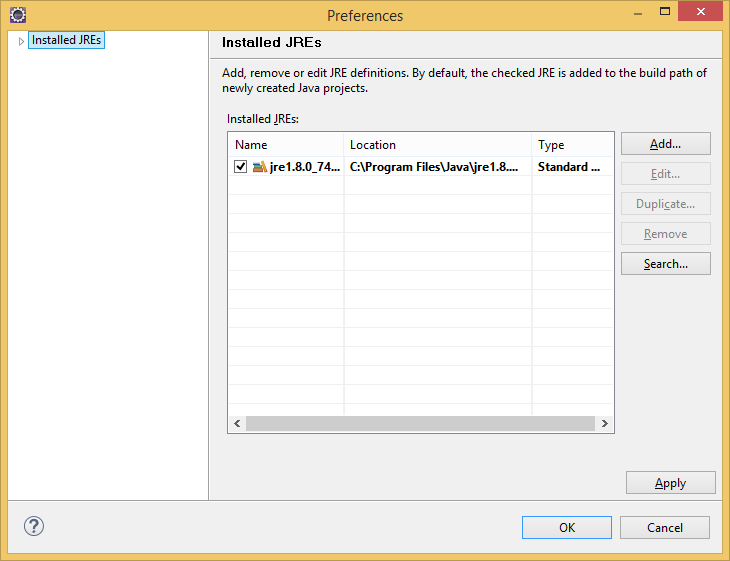
Click on Add and select Apache Tomcat v8.0:



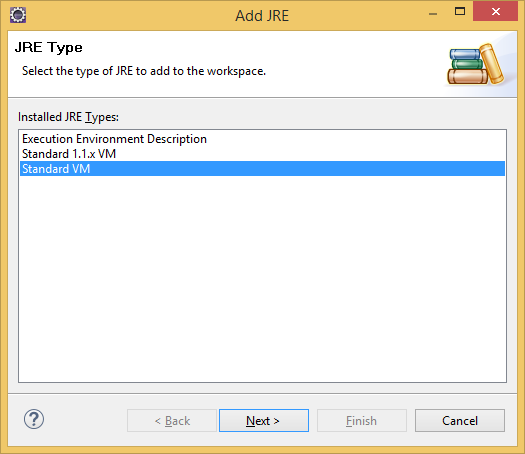
On the next screen browser for your server disk location:



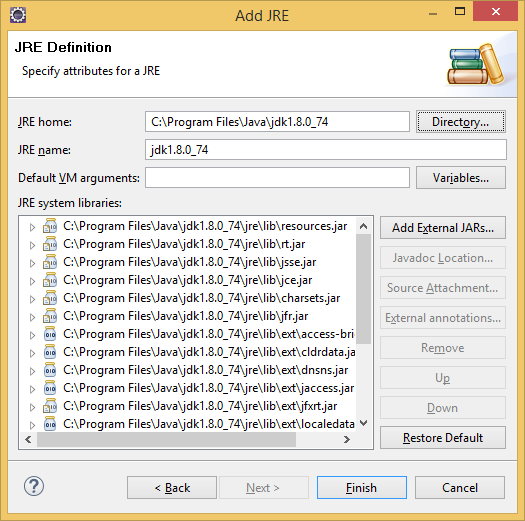
Update the JREs you have on your workspace click on Installed JREs:

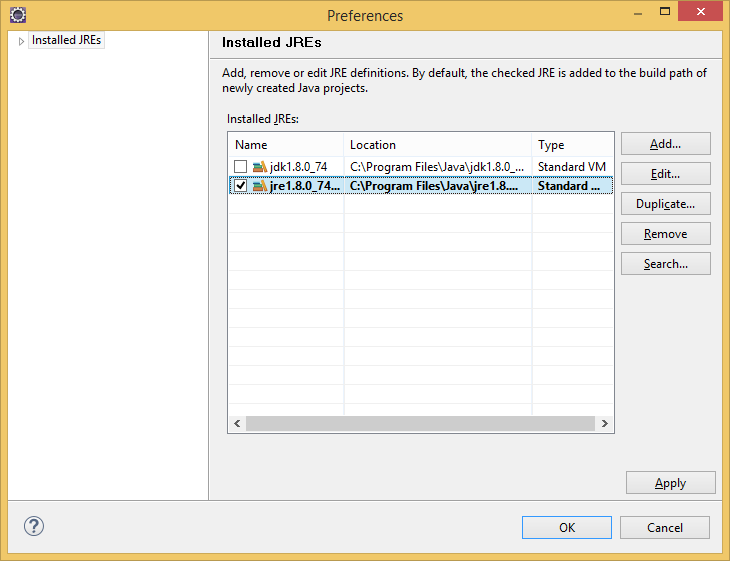


Click on Add and select Standard VM



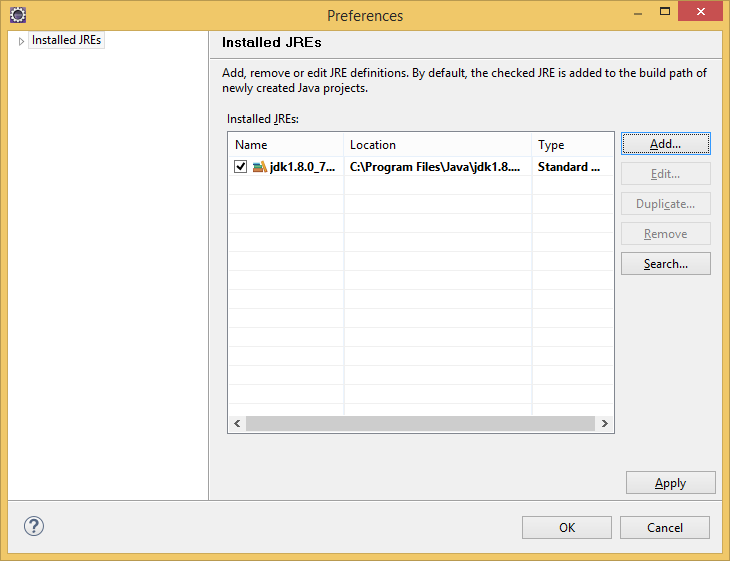
Next and browse for your JDK installation directory (click directory):



Finish ☺  
Remove the current jre and   


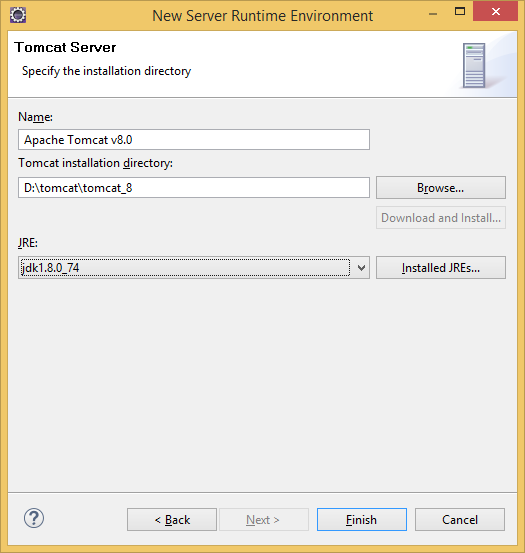
Click remove ☺

You should have something like this until now:

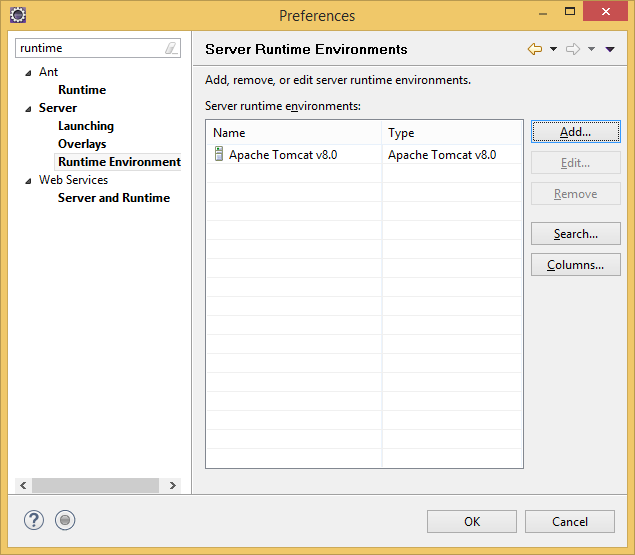


Click OK

Change the JRE from runtime environment definition window:

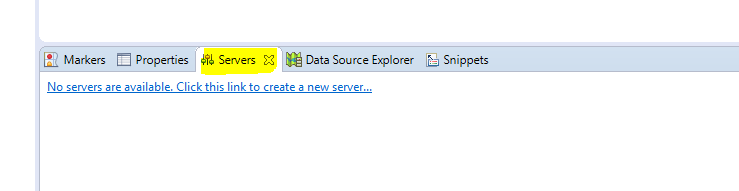


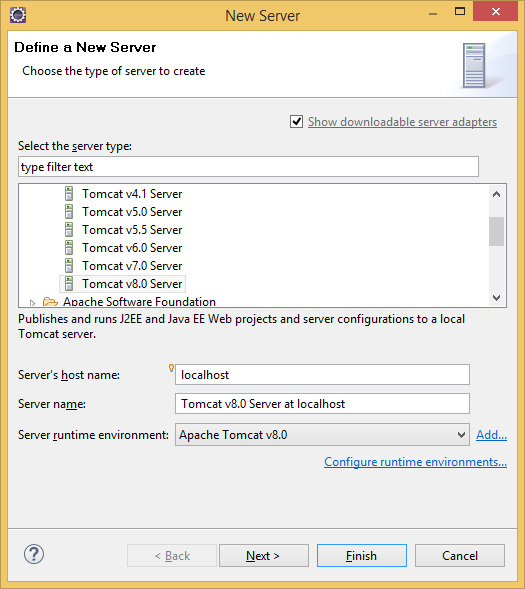
And Finish (now you have a template runtime defined).

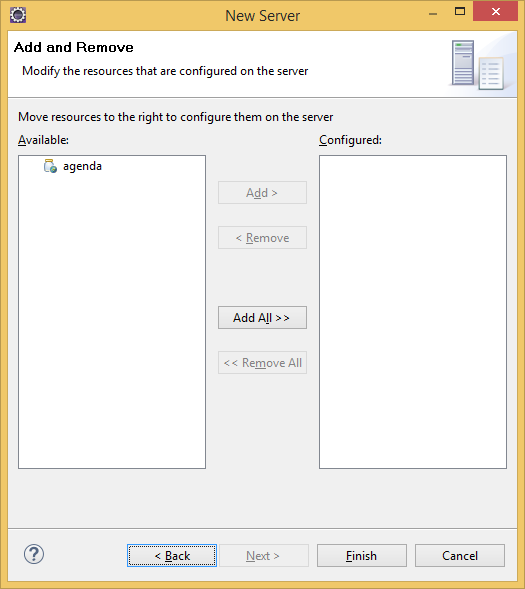


OK and you are done.

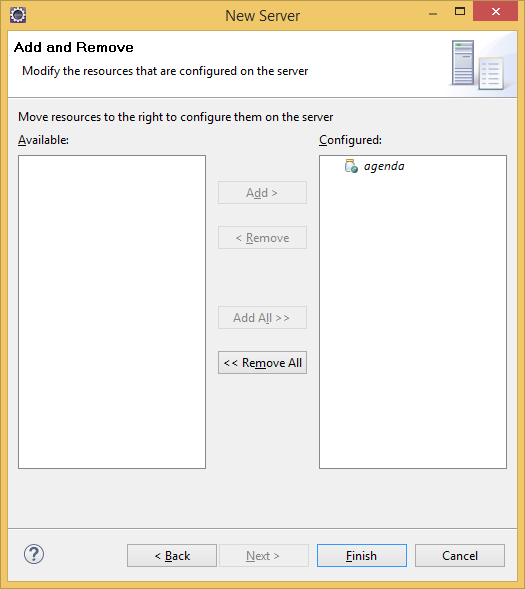
Go to Servers tab:



Click on the blue link to create a new instance of the server:  


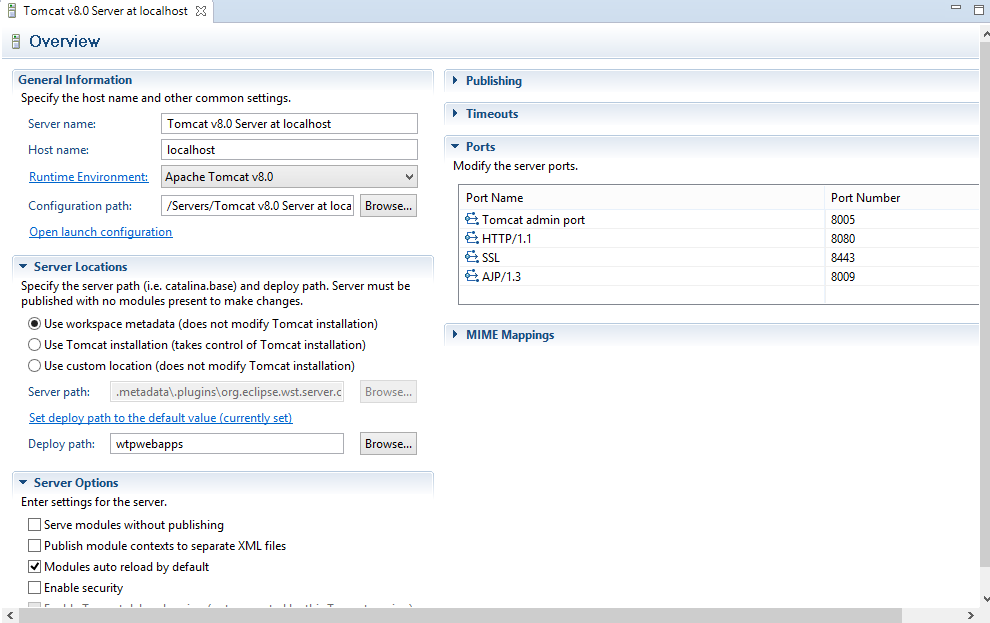
Select Tomcat v8.0 Server, click Next.  


Click on Add All to have your agenda project deployed on the server.

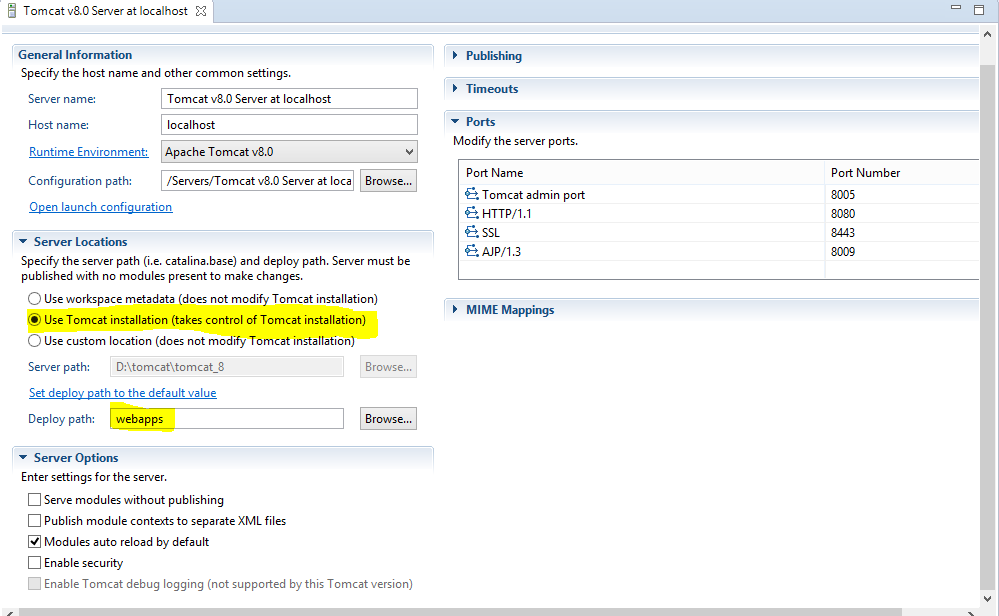


And then click finish.

Now let’s do some basic settings to our server. Double click on your server. You should get to this window:



In Server Locations select de second radio button and change from wtpwebapps to webapps:



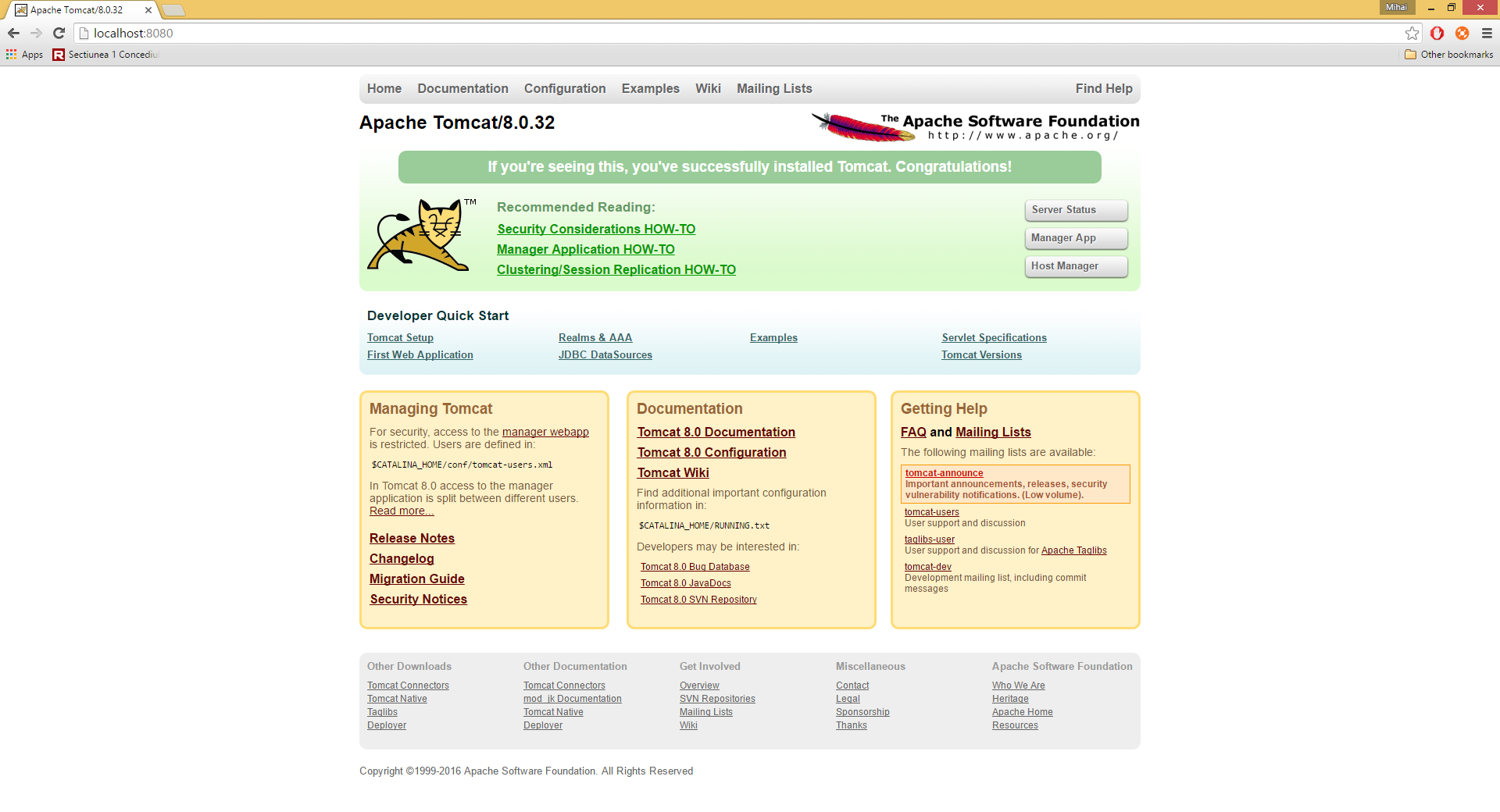
If some of you want to change the port of the application feel free to do it in the right side by changing the port with Port Name HTTP/1.1 and the value of it ☺

Save the modifications (Ctrl+s) you server should now work.

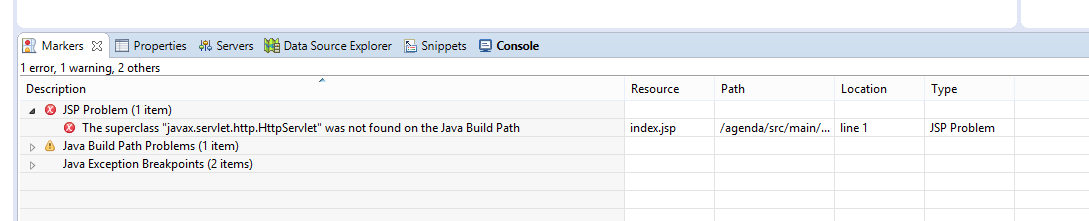
Click on the little green bug in order to start the server in debug mode and your server should display the following info: [Debugging, Synchronized]



And if you go to your browser to <http://localhost:8080/> you should get the following details:

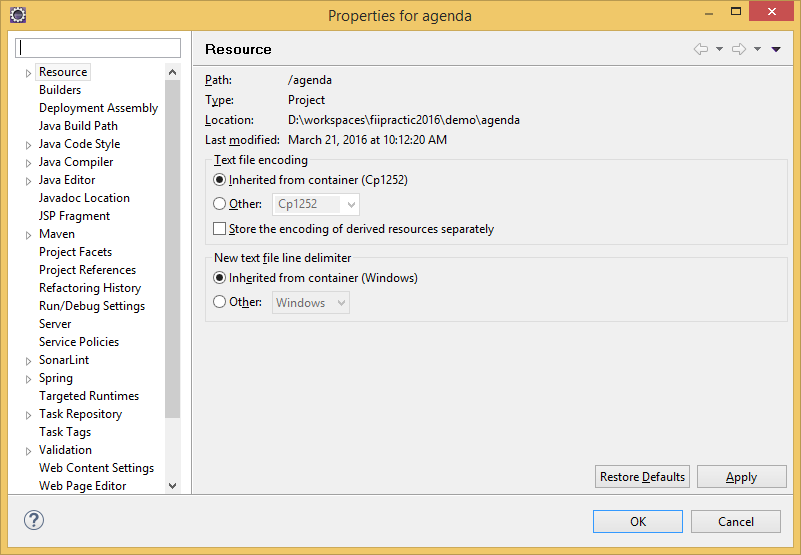


# Getting rid of javax.servlet.http.HttpServlet jsp problem

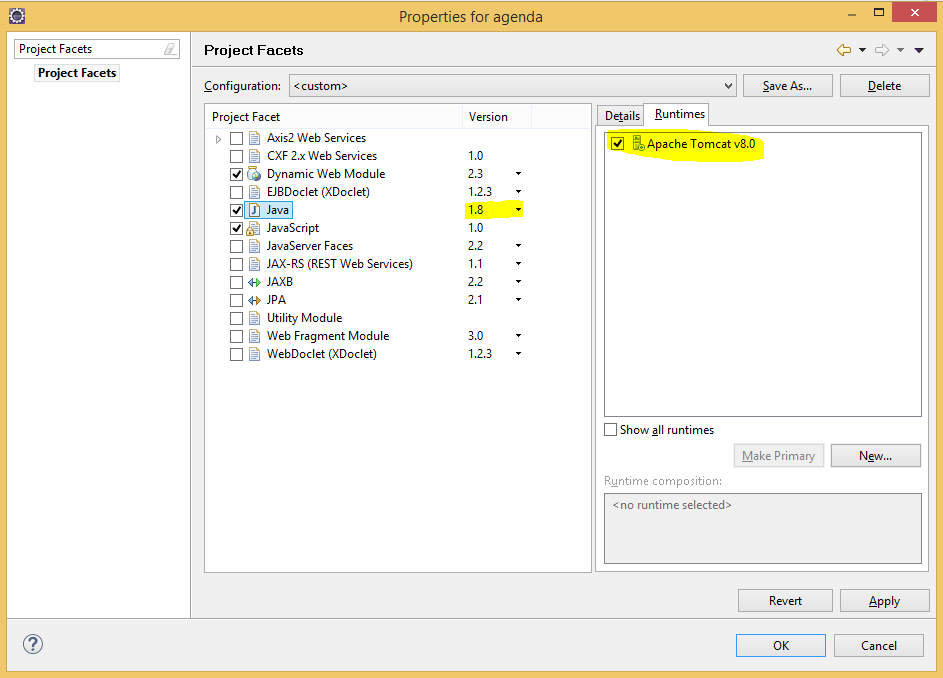


## Method A

In Project explorer right click on the agenda project and select properties:



In the type filter text write Project Facets and then click on it. Go to Runtimes and select Apache Tomcat v8.0, also if you are here you can select the java version to be 1.8.



## Method B

Go to your pom.xml file.

And add the following dependency:

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

The provided scope means that the Application Server (in our case Tomcat) provides the javax.servlet.http.HttpServlet.

I usually go with method B if I know that I’m going to import a lot the project.

# Configuring our maven project

## Configuring pom.xml

Open your pom.xml file and add dependency for spring-webmvc  
Go to <http://mvnrepository.com/artifact/org.springframework/spring-webmvc/4.2.5.RELEASE> and copy the maven dependency group and paste it to your dependencies group in your pom.xml. You should have this info in your pom.xml:

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.2.5.RELEASE</version>

</dependency>

</dependencies>

## Configuring web.xml

Now let’s take care of our web.xml configuration file.

1. Add a new servlet

<servlet>

<servlet-name>rest</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

</servlet>

1. Add a new servlet mapping

<servlet-mapping>

<servlet-name>rest</servlet-name>

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

</servlet-mapping>

By doing this we are telling our application that any request that are being sent to <http://localhost:8080/agenda> are going to be handled by the servlet defined by the class: org.springframework.web.servlet.DispatcherServlet. This way we are giving the control of requests to the Spring framework.

## Adding our beans configuration file

I would suggest to add to your eclipse the Spring IDE tools (easy to write bean definitions and others). Get it from [here](https://marketplace.eclipse.org/content/spring-ide).

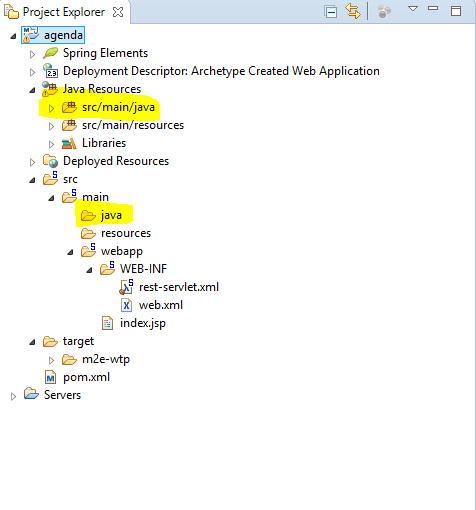
Add rest-servlet.xml configuration



## Adding source (java) folder

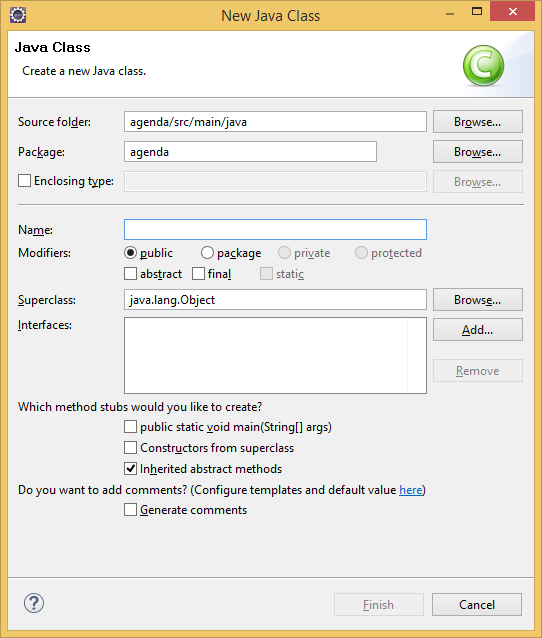
Right click on src/main and add a new folder called java.

You must have the following structure on your project:



## Add your first package and first class

Right click on the src/main/java from Java Resources and select New -> Class (see image below)

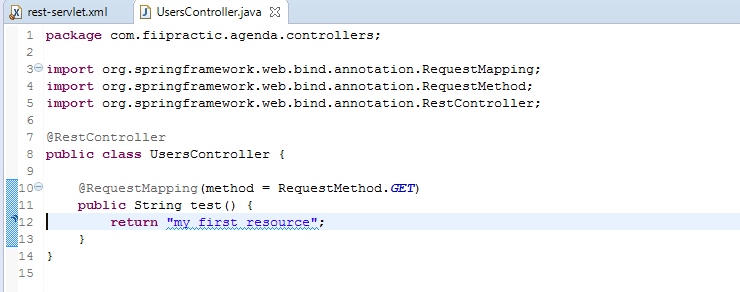


Fill with the following data:

Package: com.fiipractic.agenda.rest.controllers

Name: UsersController

You should get a new java file open in which you should fill like in the image below:



## Publish your application to the server

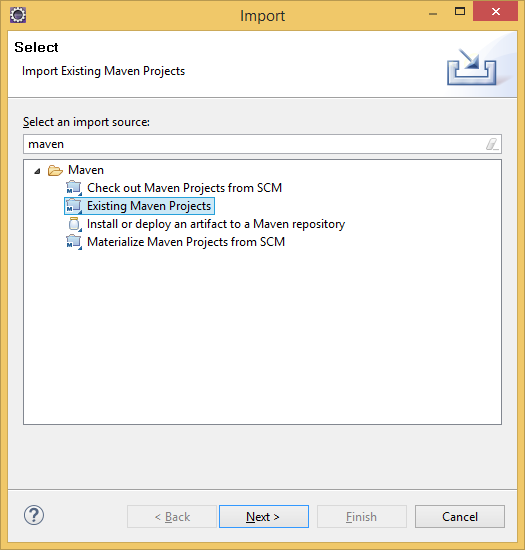
Go to Servers tab and right click on your server and select Publish (Ctrl+Alt+P). Start the server in debug mode (the little green bug button or Ctrl+Alt+D).

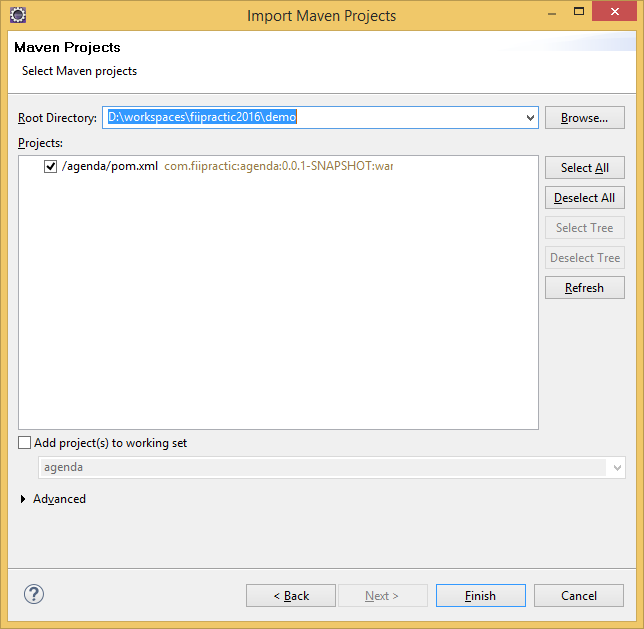
Go to <http://localhost:8080/agenda> to access your first resource.

# Import a maven project

File -> Import.

In the Select an import source write maven and select Existing Maven Projects.



Click next and browse for your pom.xml file. (E.g. mine is located at: D:\workspaces\fiipractic2016\demo)  


Click Finish and your project is imported.

In order to add it to your server you need to go to Server tab and right click on your server and select Add and Remove … and move agenda from available to configured location, the same way you did it when you defined the server.



Start the server and your application should be up and running.