Web applications: (jsp/servlet): framework

Spring MVC: Model View Controller

Request -🡪 Controller 🡪 Populate ur data (model object/ pojo{account/customer}) -> view

Controller:

1. Request mapping
2. Populating the data
3. Return ur view name

**Spring MVC**:

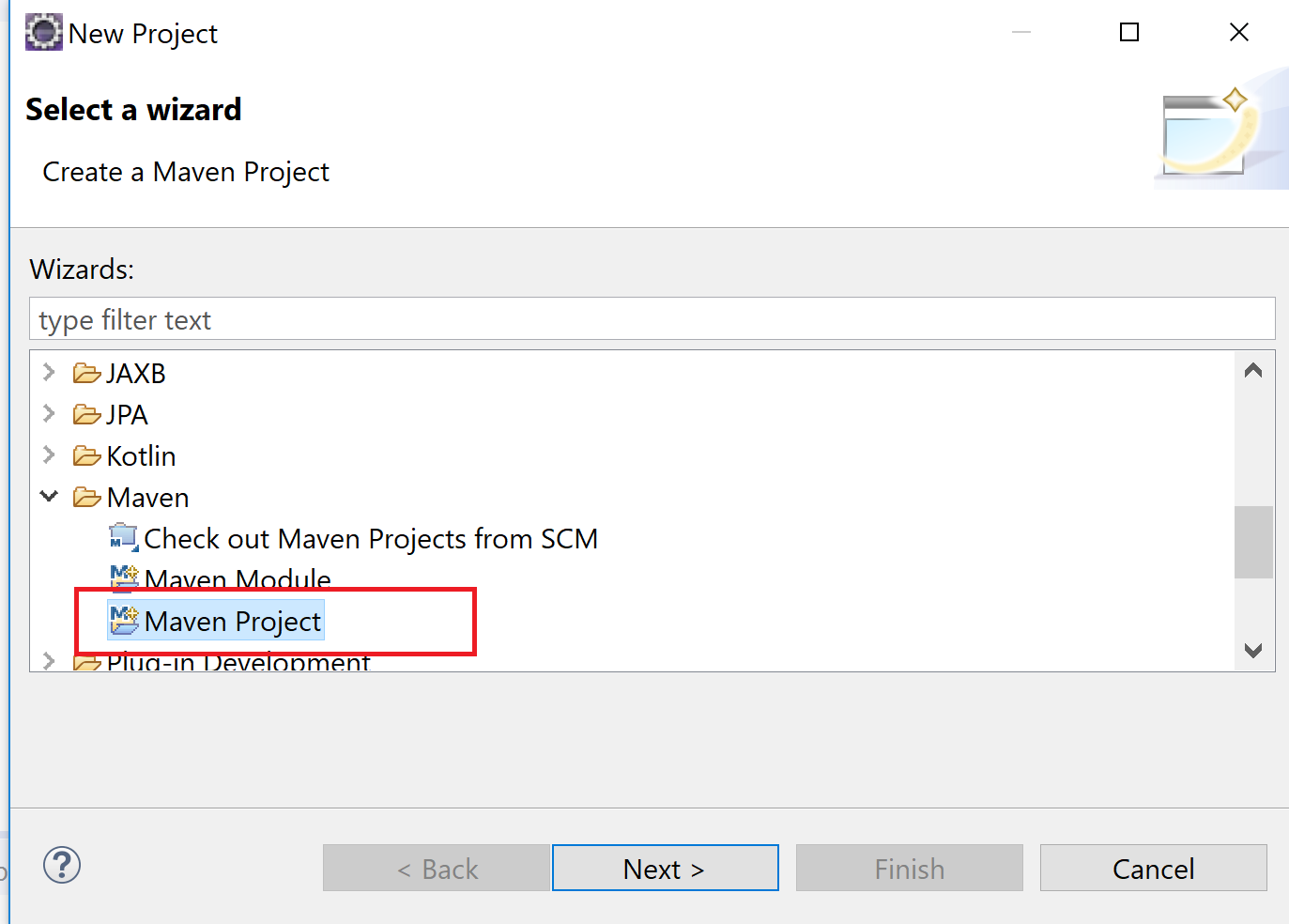
1. DispatcherServlet: front controller (url u will configure for this servlet, the request would be handled by ur spring container)
2. HandlerMapping
3. ViewResolver

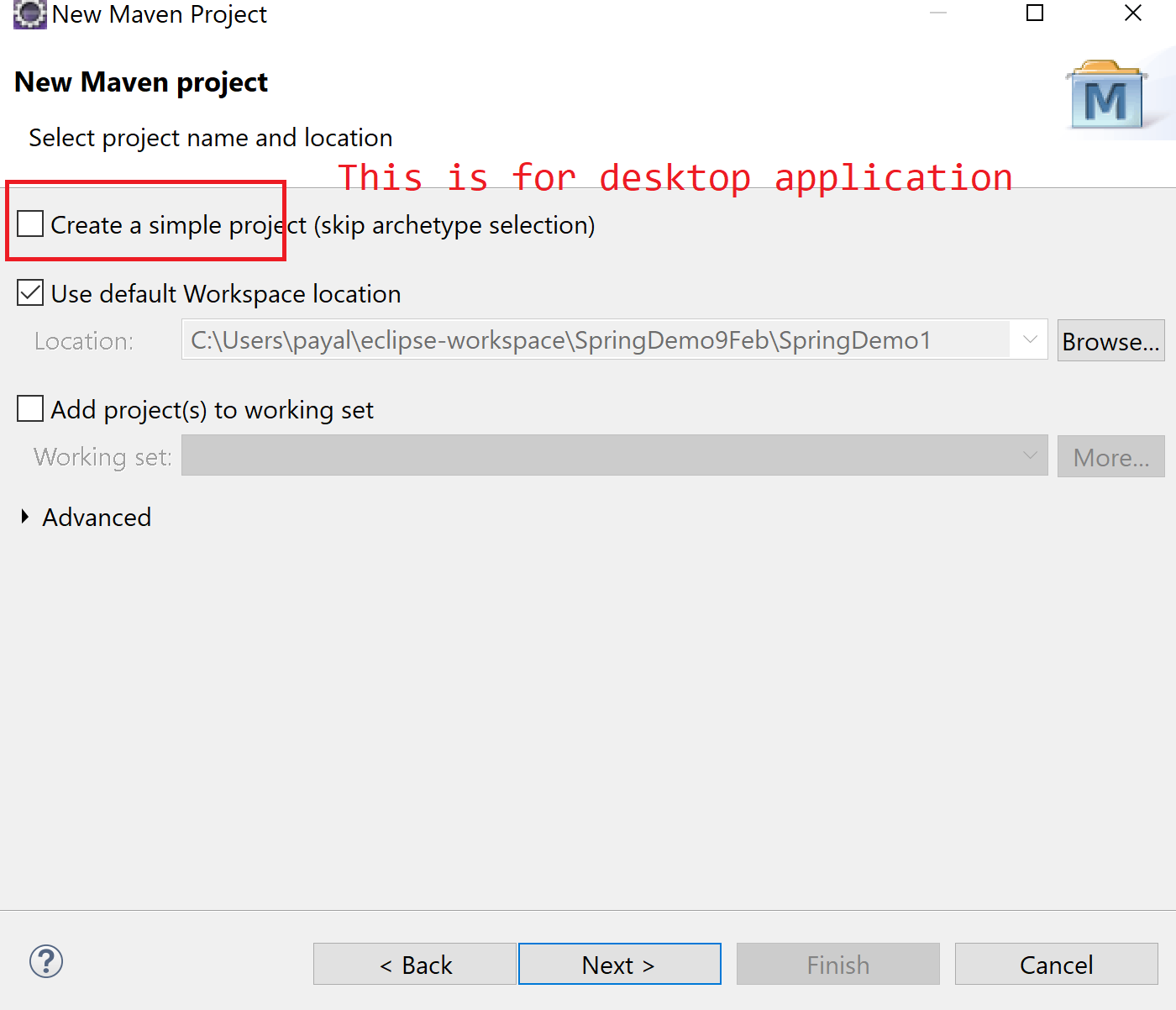
**Jsp servlet:**

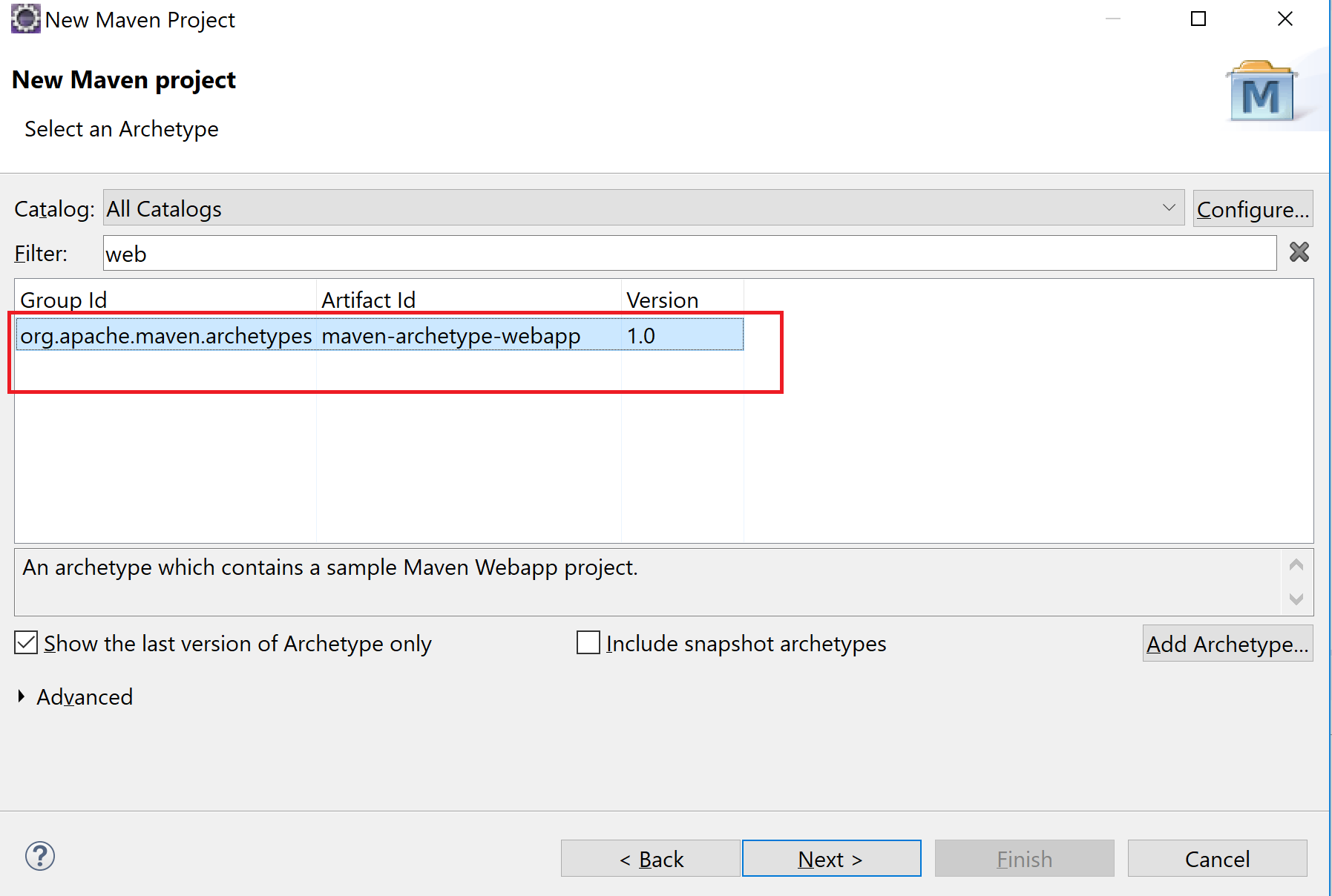
Web.xml: servlet mapping (url/ servlet)

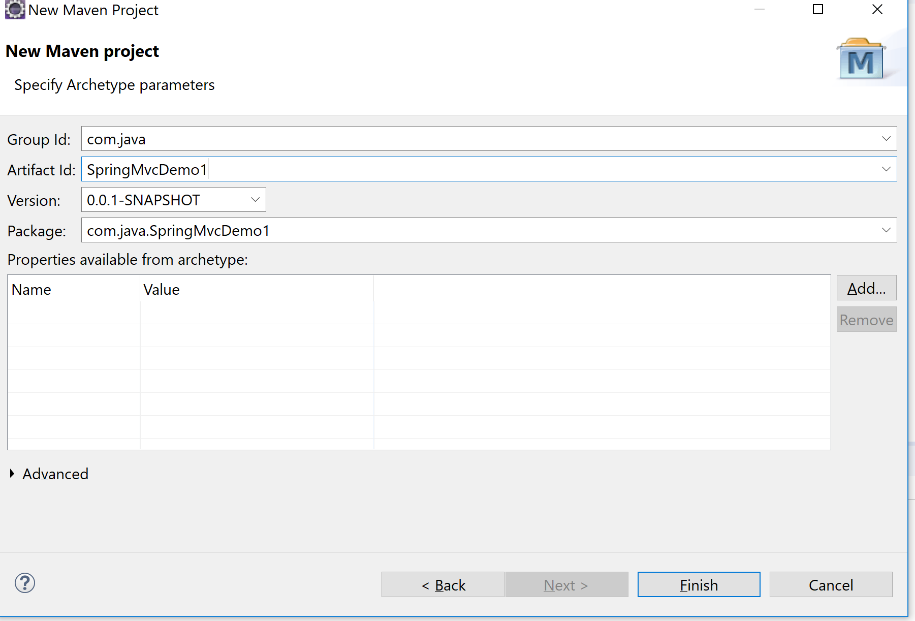
1st Request 🡪 Object -> init() 🡪 service()

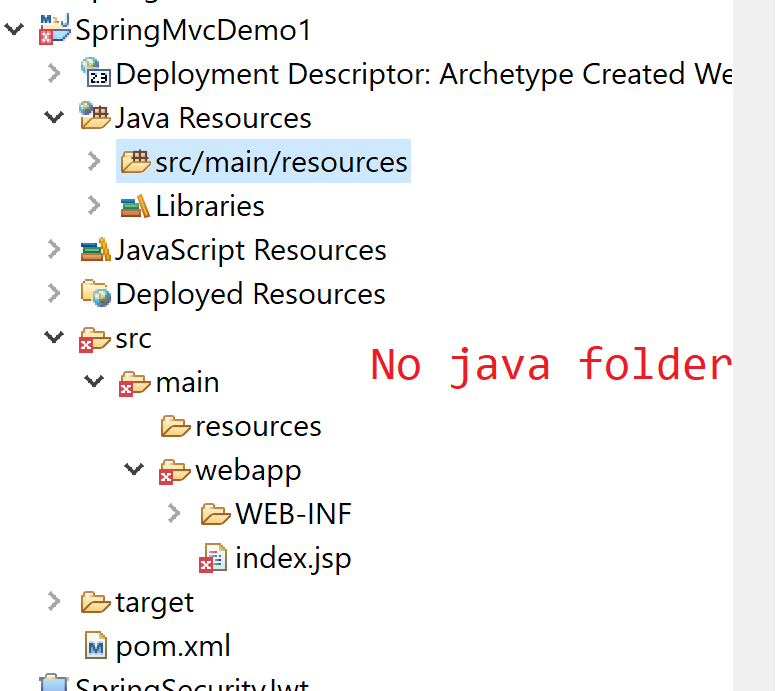
Other requests -> service();

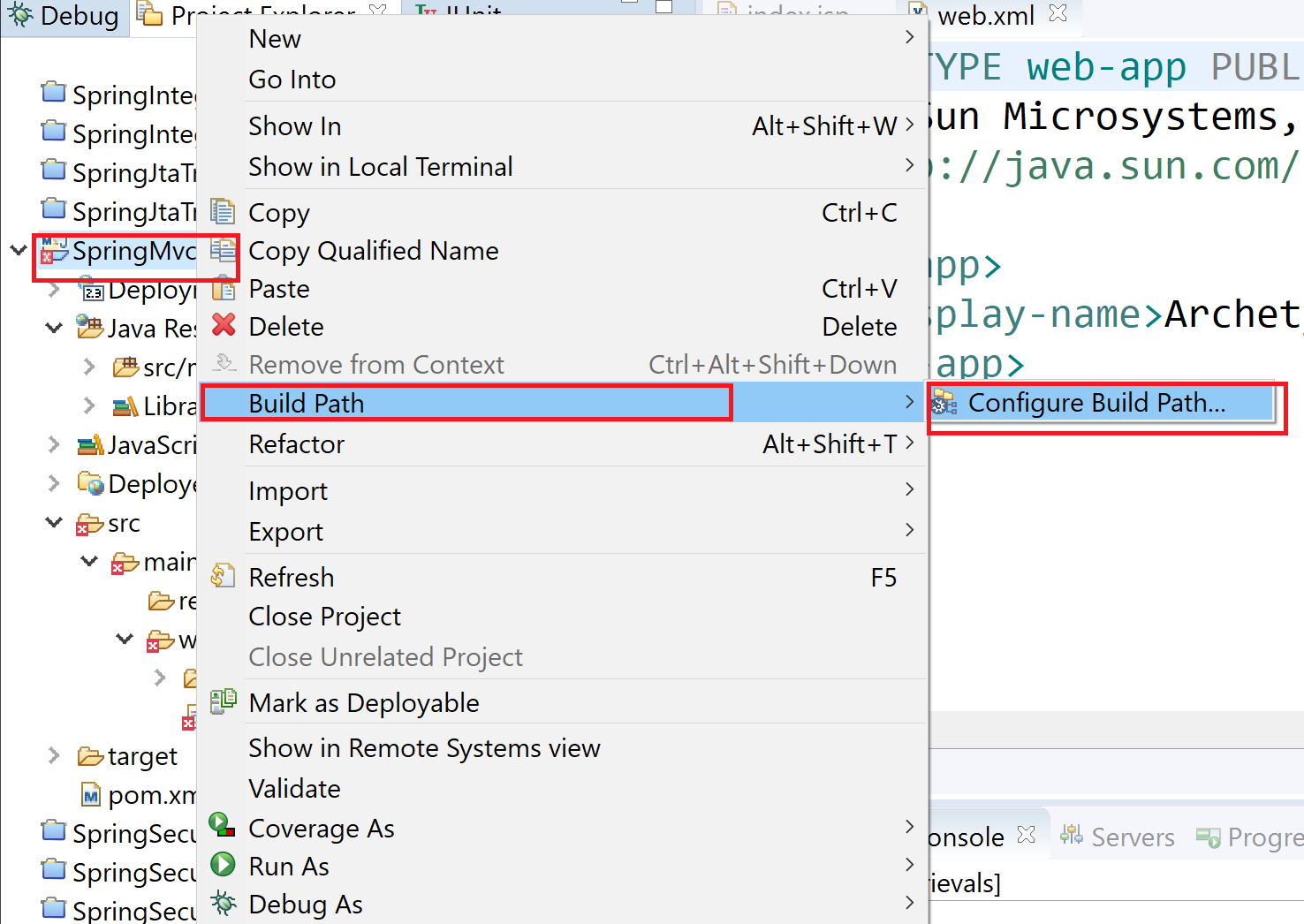


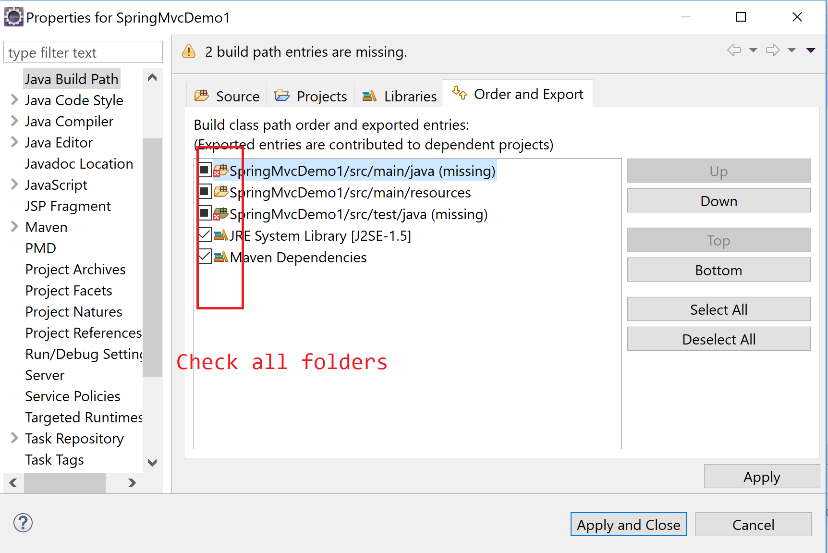


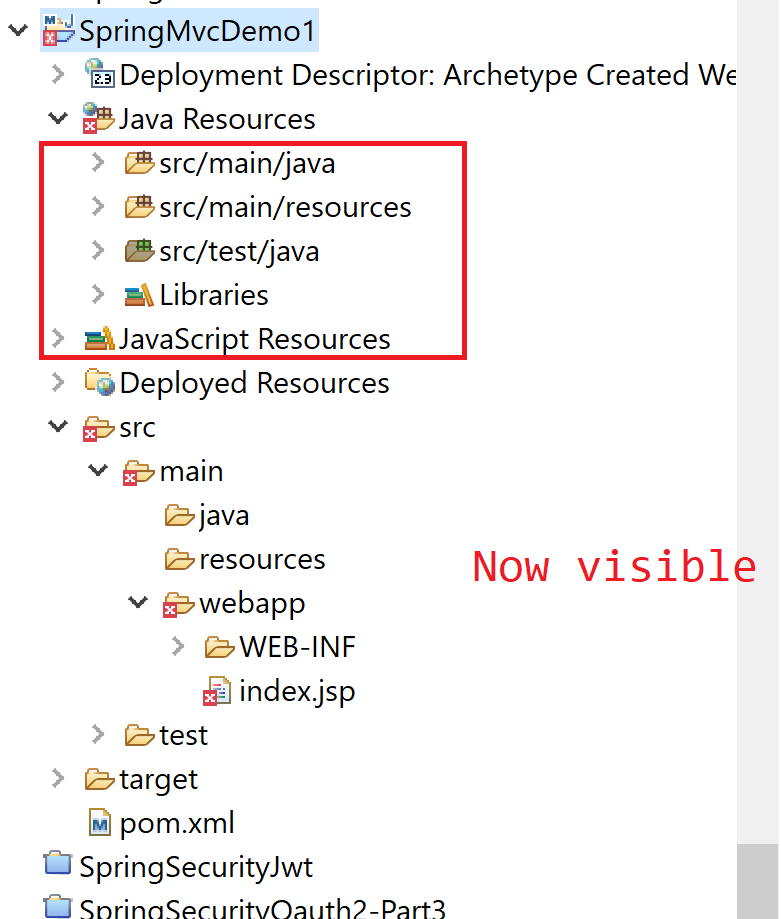


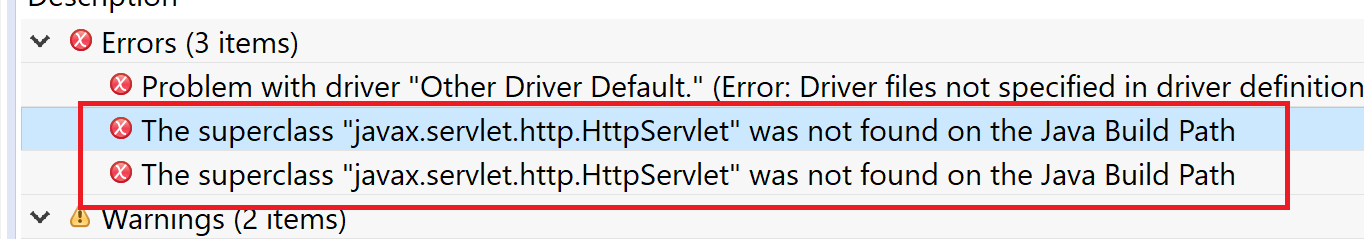


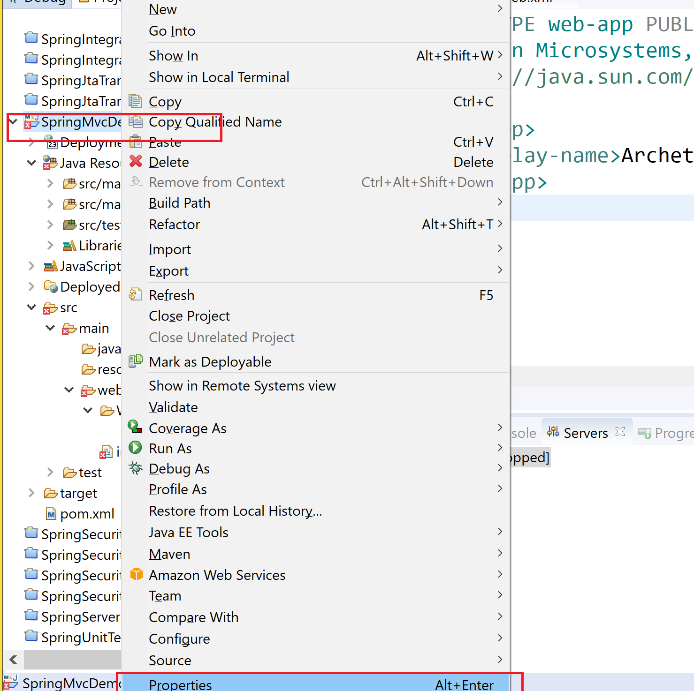


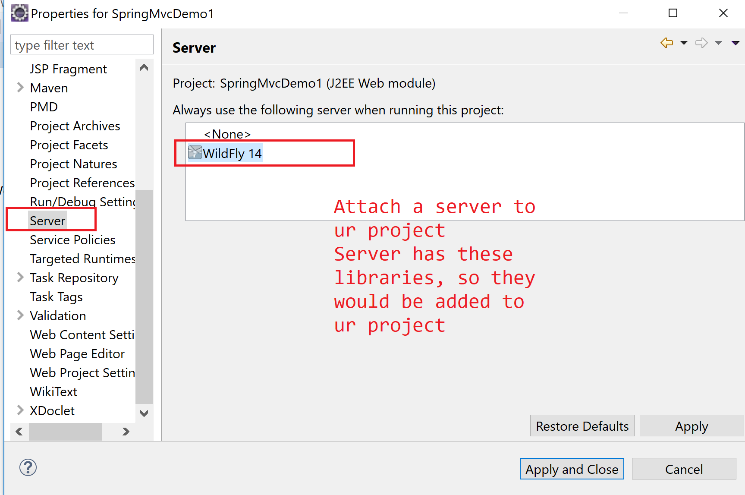


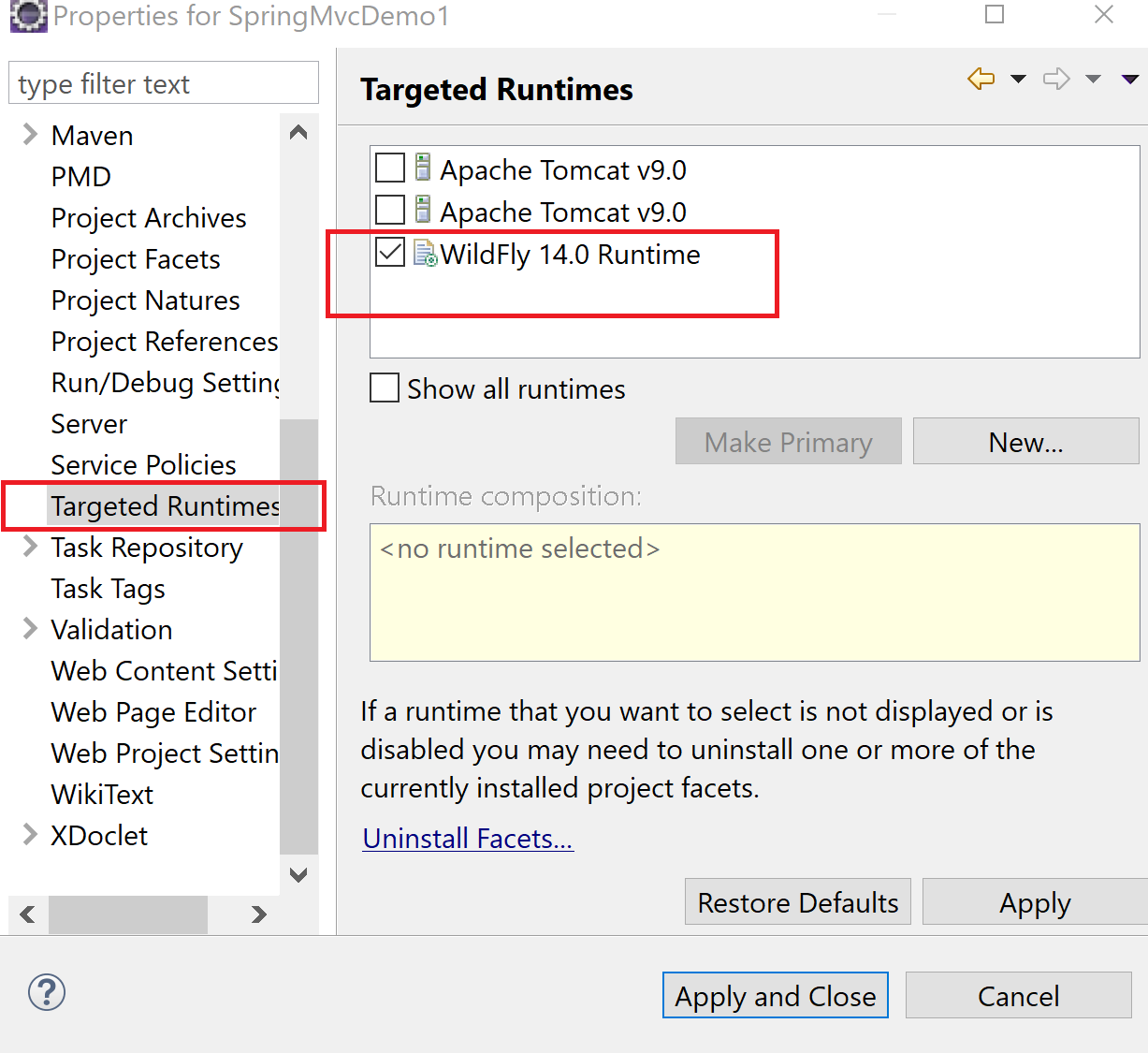


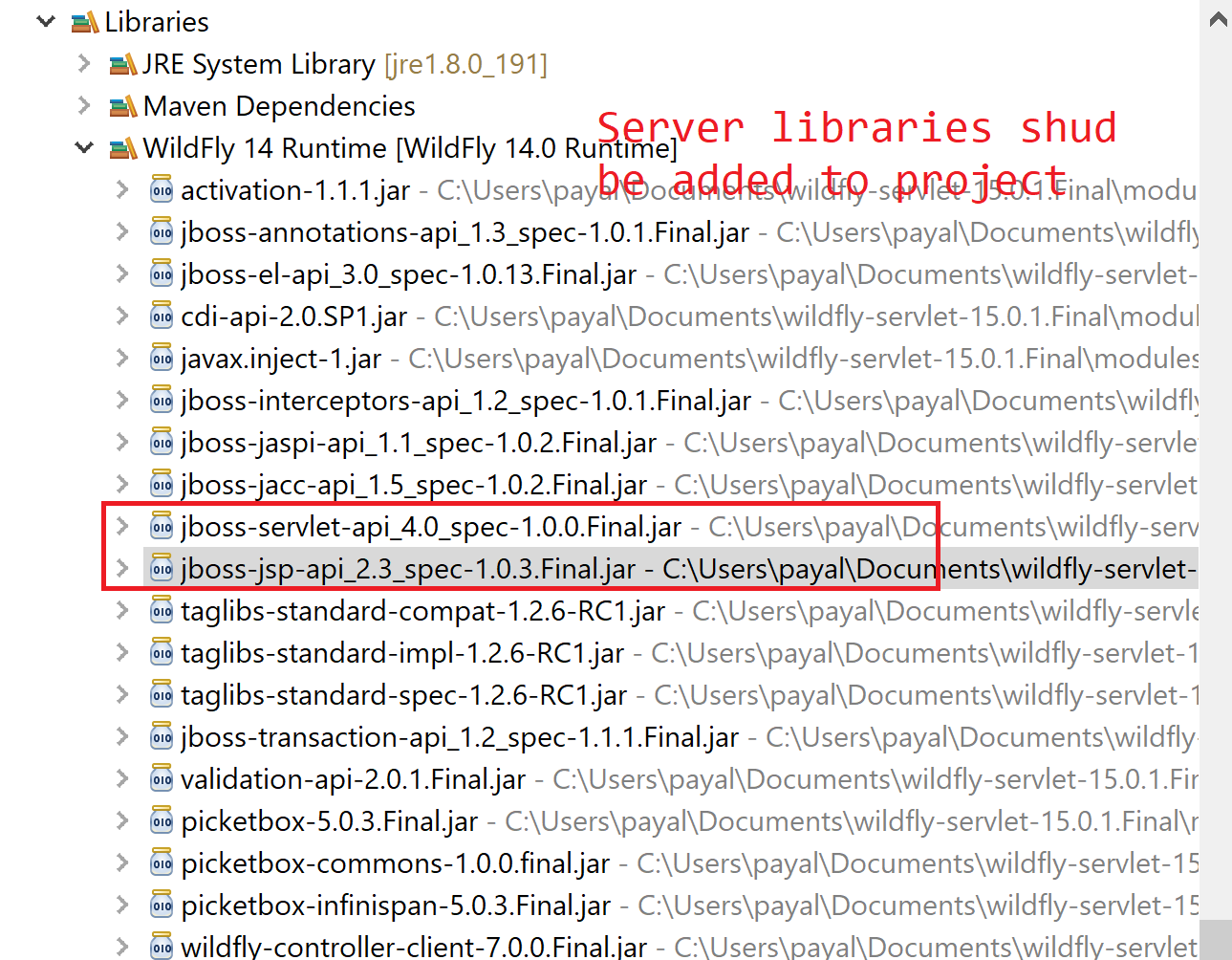






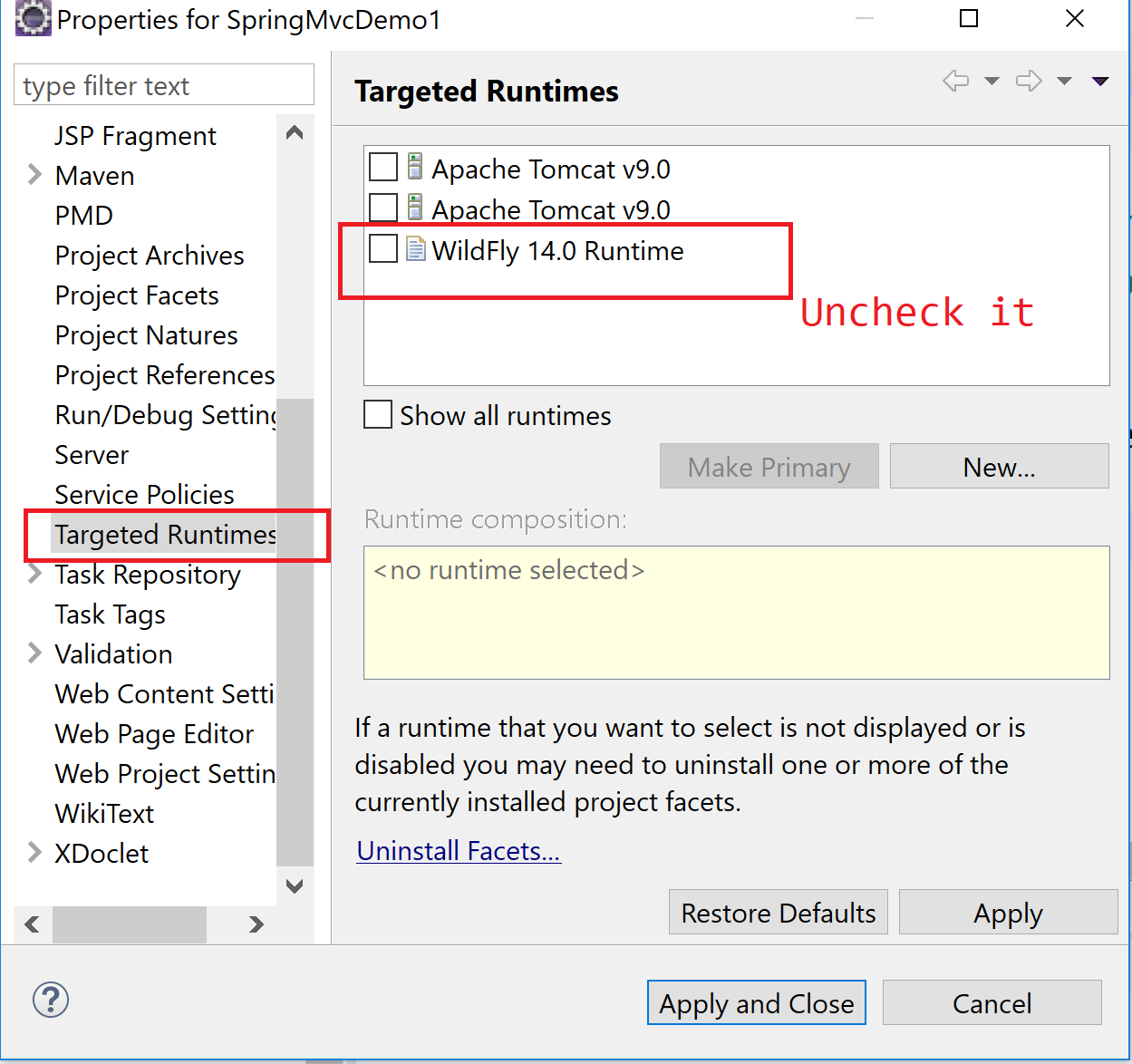


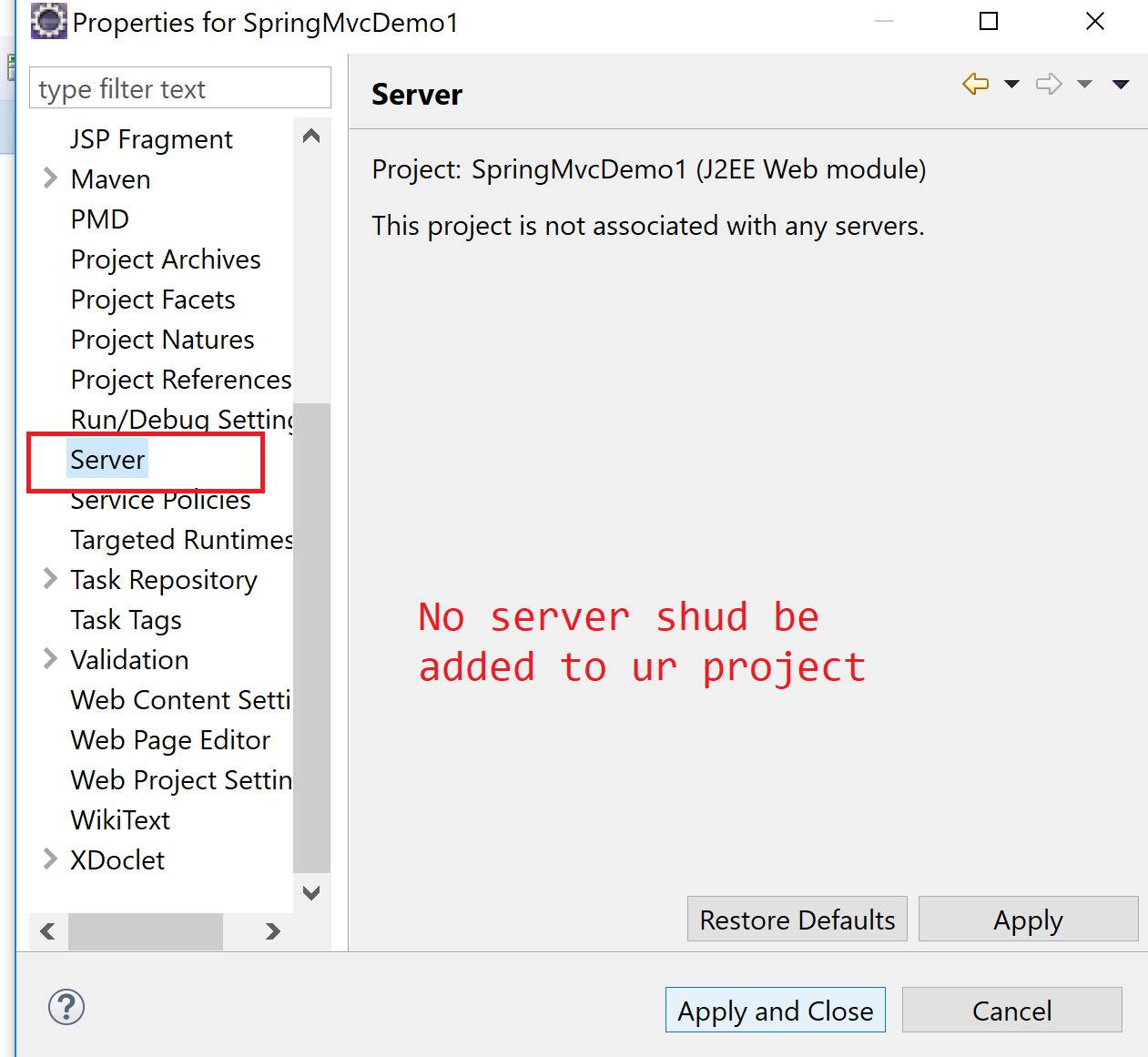


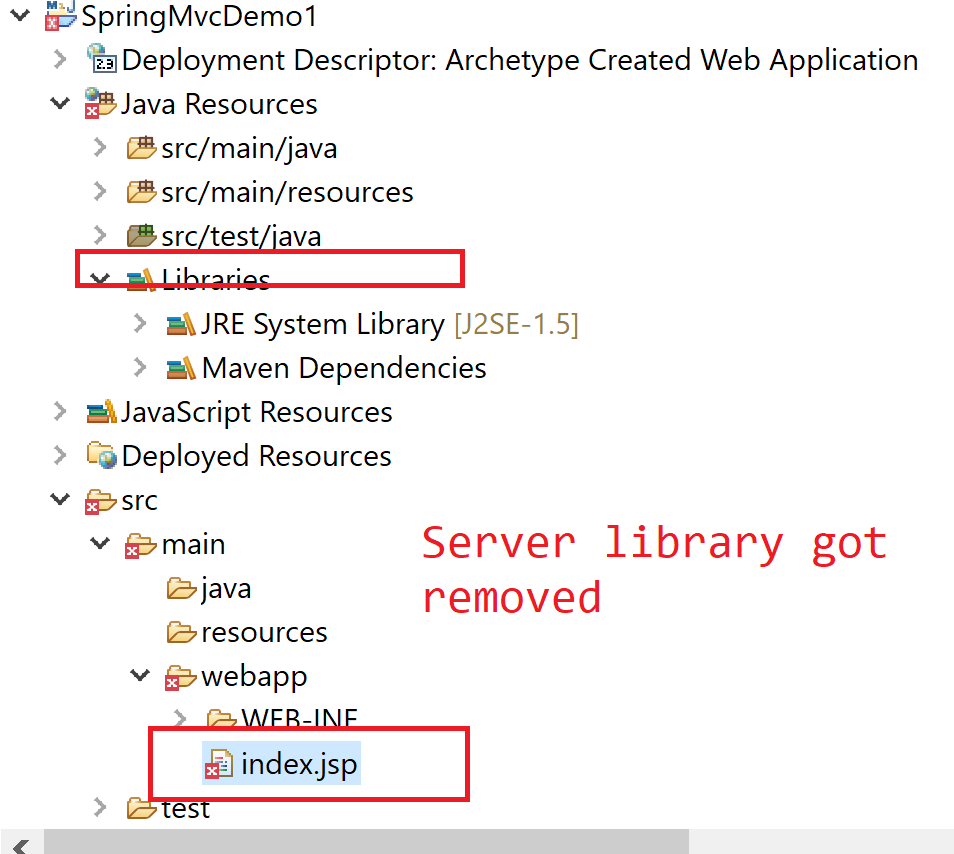


**Embedded server:**

Add a server as a plugin to ur project







Need to add libraries for jsp & servlet

Add dependencies for servlet-api

<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->

<dependency>

<groupId>javax.servlet</groupId>

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

<version>4.0.1</version>

<scope>provided</scope> //Needed only at compile time.

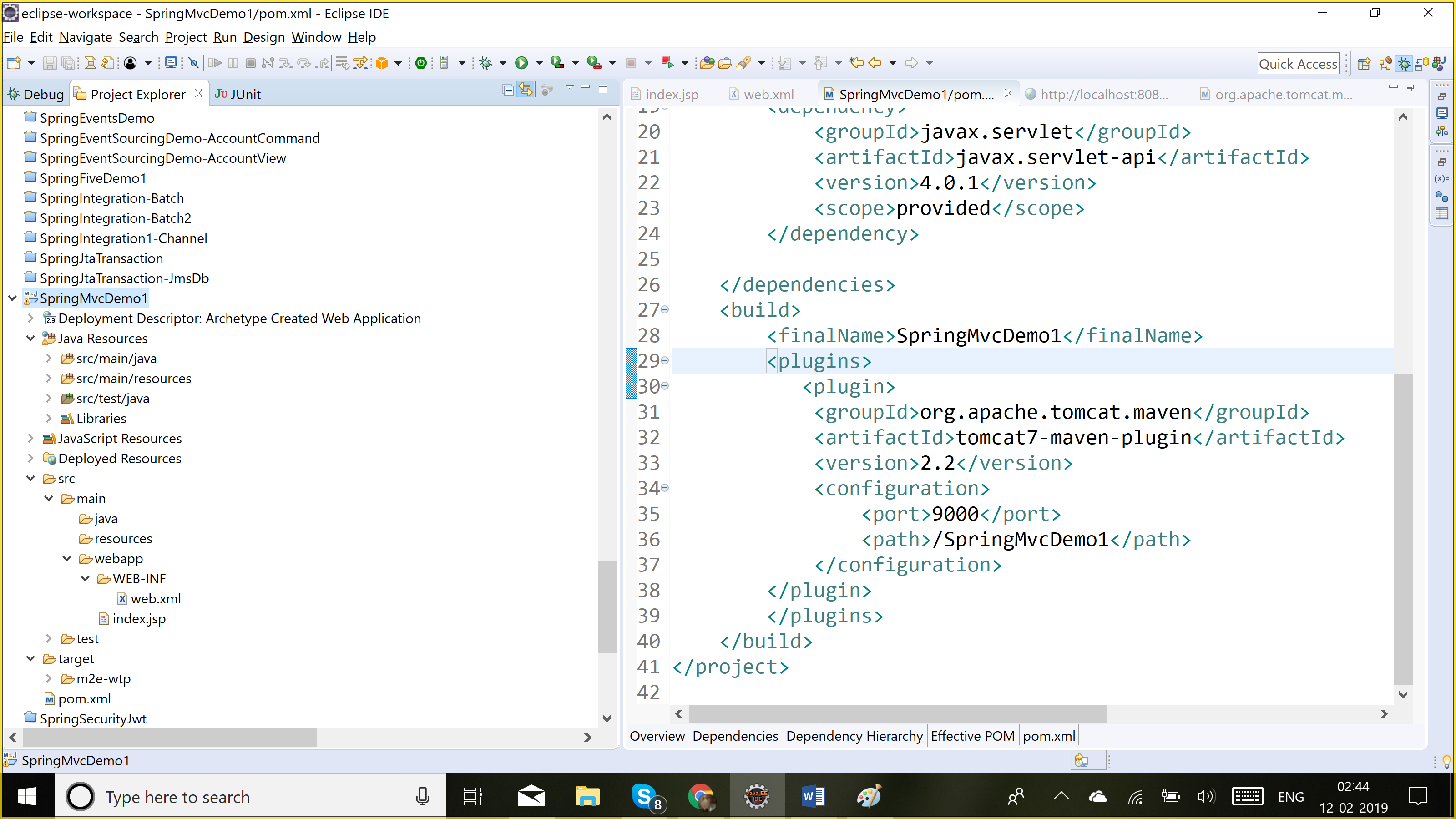
</dependency>

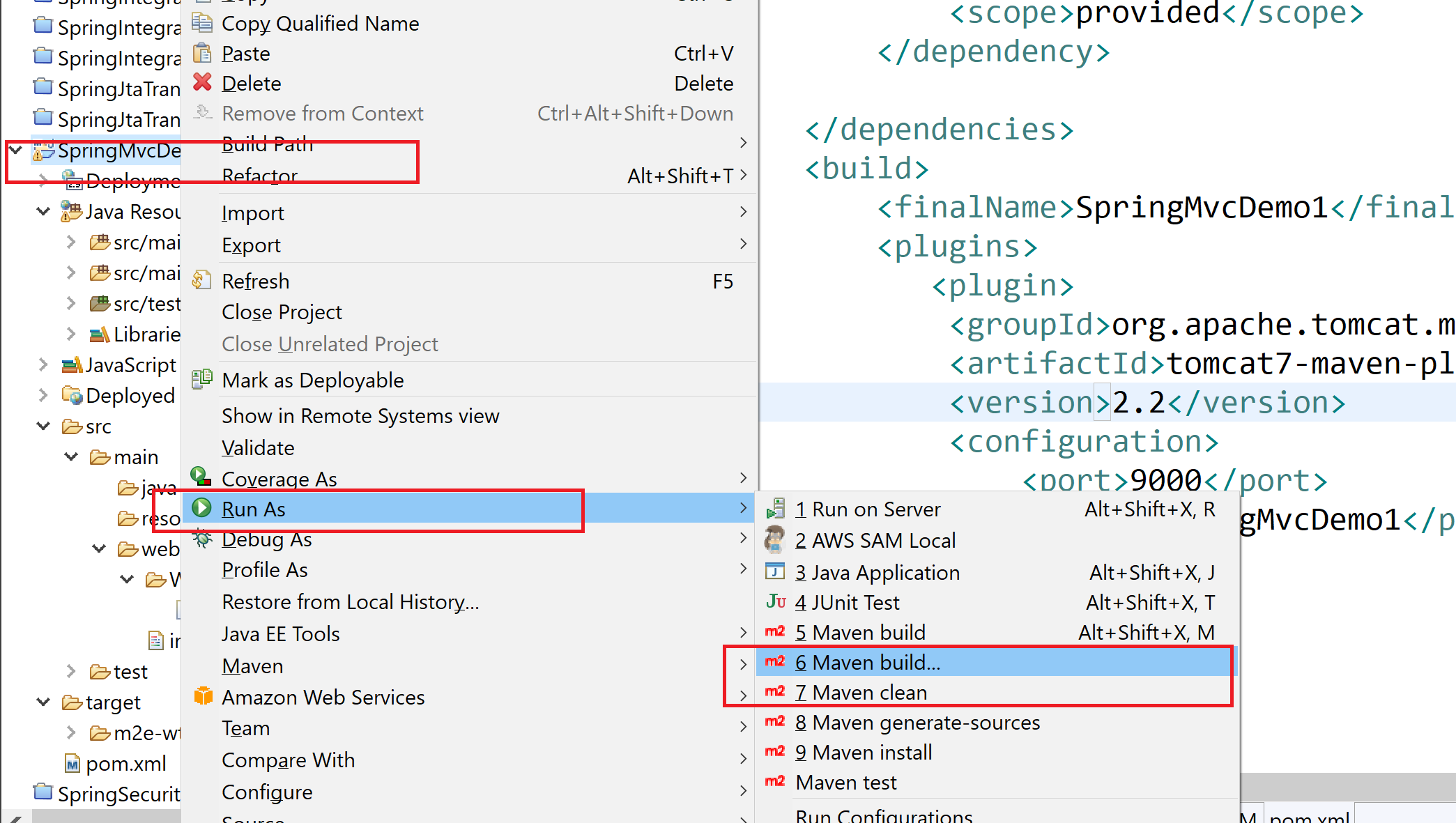
War file: classes+ libraries+ html/ css/js (webapp): web archive

Jar file: desktop application: class files + libraries: java archive

**Server is needed**

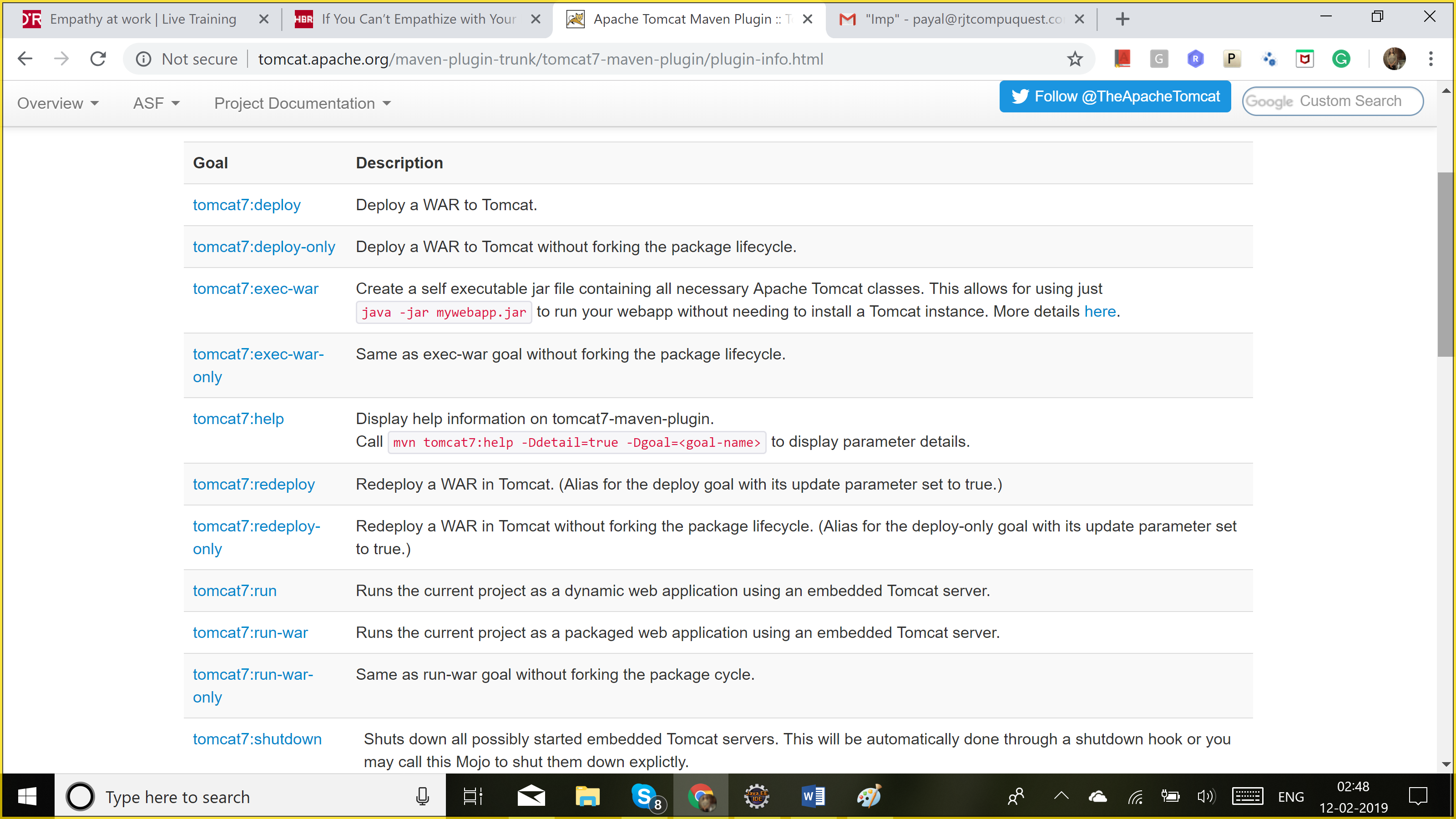
**Embedded tomcat server plugin**

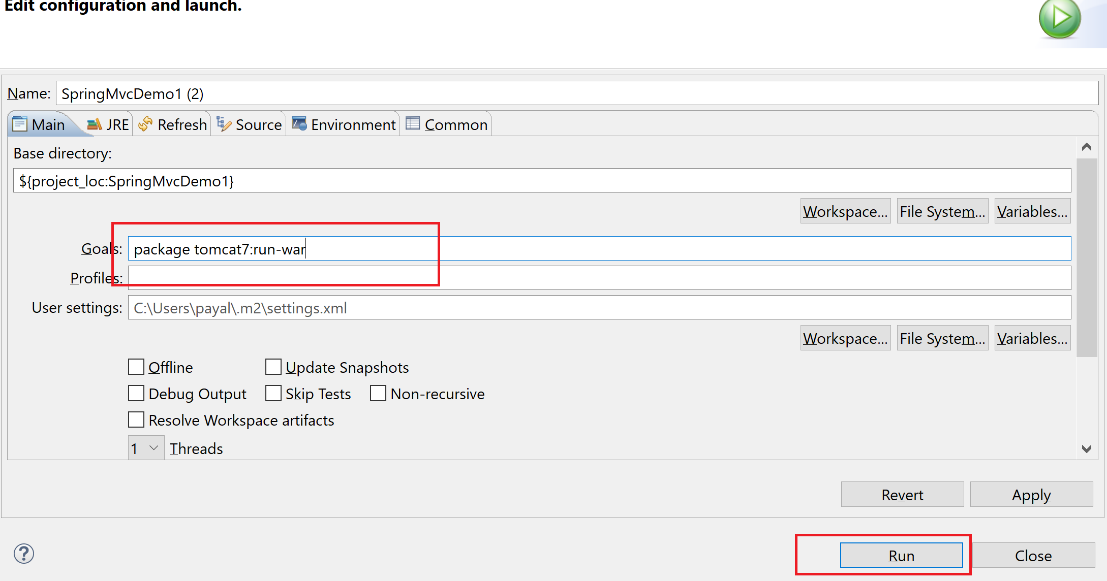
****

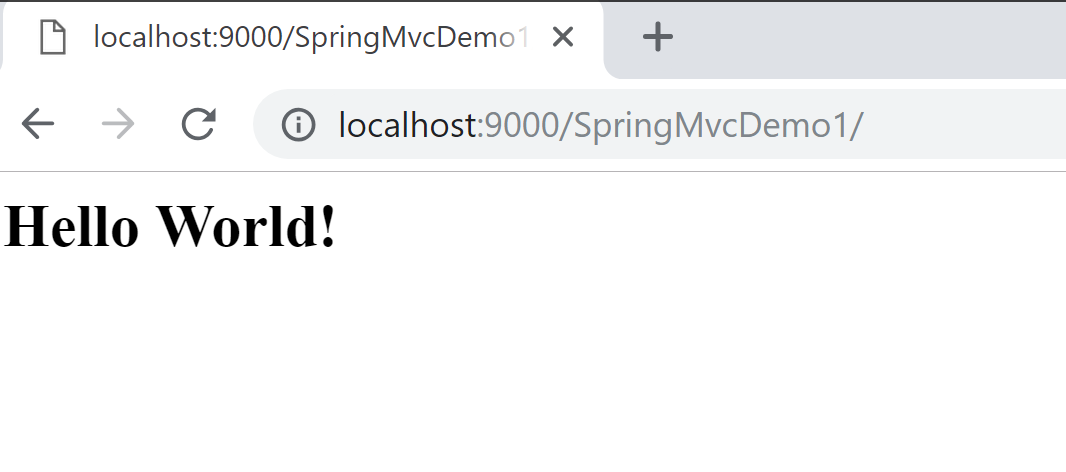
****

**Package:** compile ur java code: class file (maven-compiler-plugin)

Tomcat server and deploy ur app on tomcat



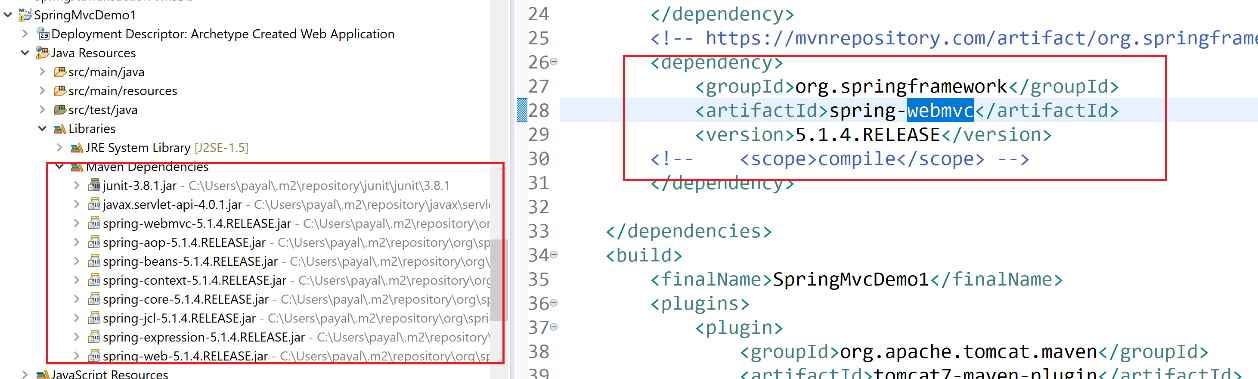




SpringMvc: spring core jars are needed

Maven: mvc {core}





**Web.xml**

Webcontainer will read ur web.xml: decide which servlet to take the request to.

<web-app>

<servlet>

<servlet-name>S1</servlet-name>

<servlet-class>com.java.Servlet1</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>S1</servlet-name>

<url-mapping>/servlet1</url-mapping>

</servlet-mapping>

</web-app>

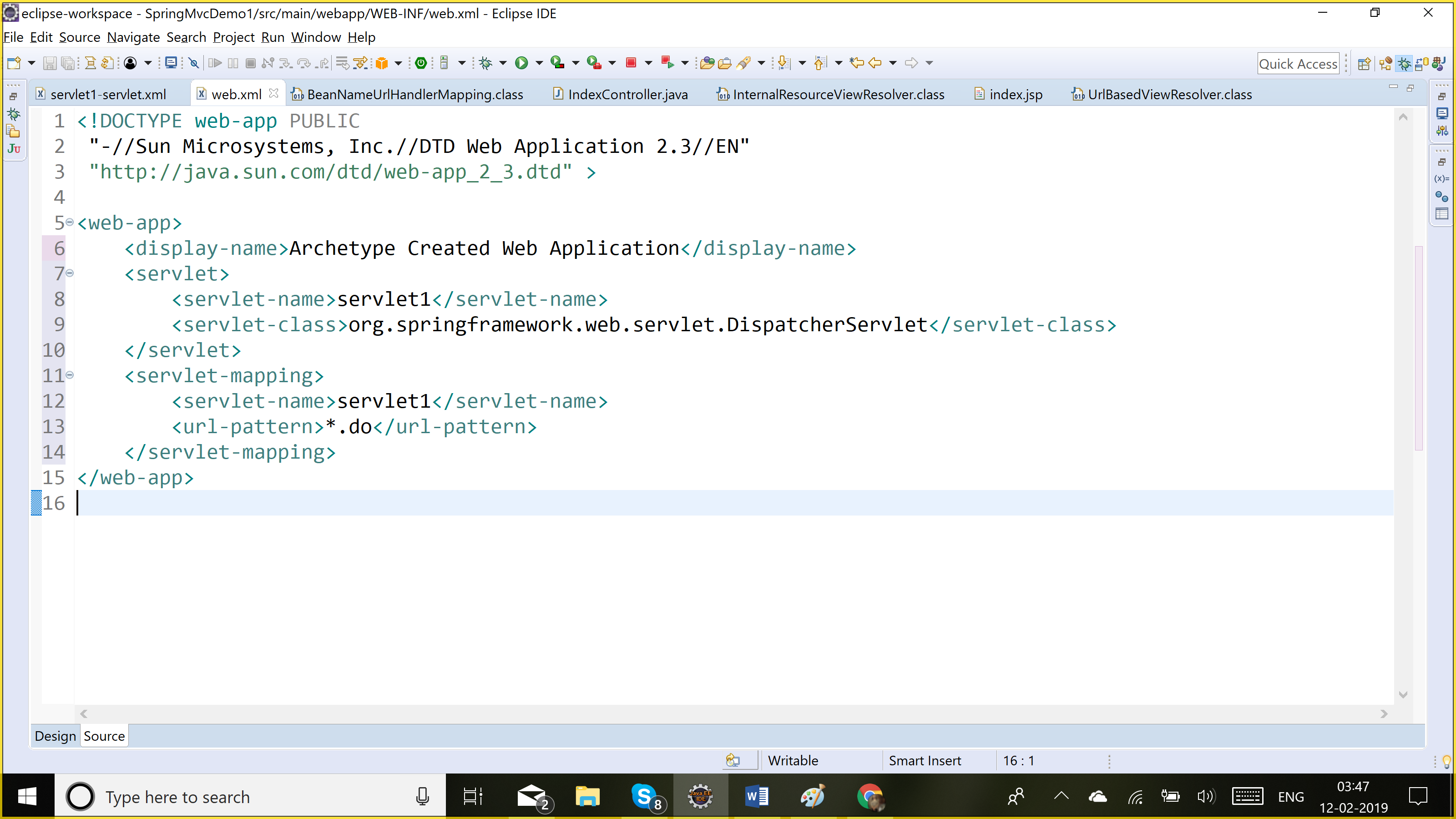
<http://localhost:8080/servlet1>

Spring: Spring container: managing the beans

Webapplication: WebApplicationContext

<http://localhost:8080/SpringMvcDemo1/index.do>: spring container

<http://localhost:8080/SpringMvcDemo1/index.html>: web container



Dispatcher Servlet acts as a front-controller. So any request coming for any url ending with .do would be handled by this controller.

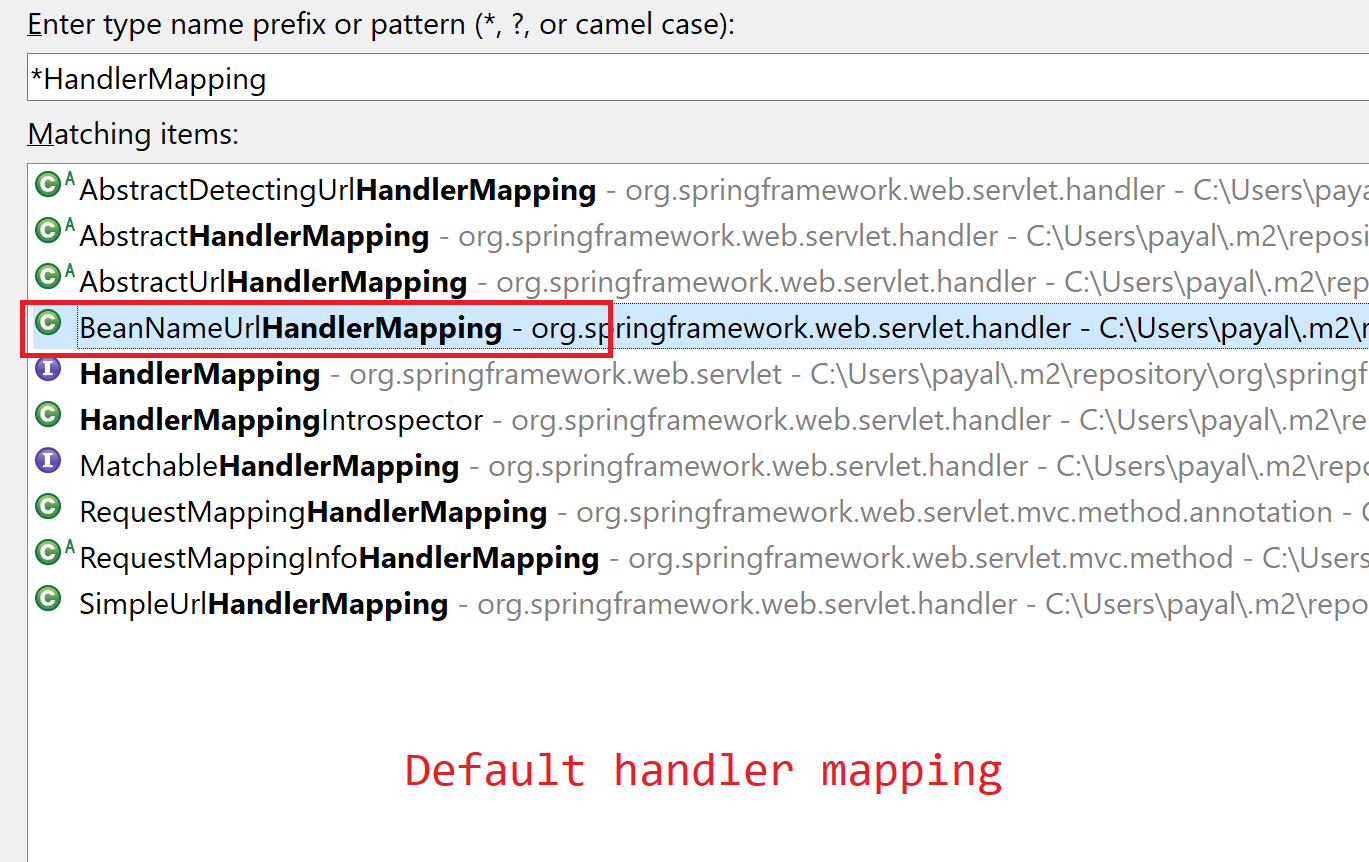
Dispatcher servlet will look for spring configuration file to figure out which spring controller to send the request to.

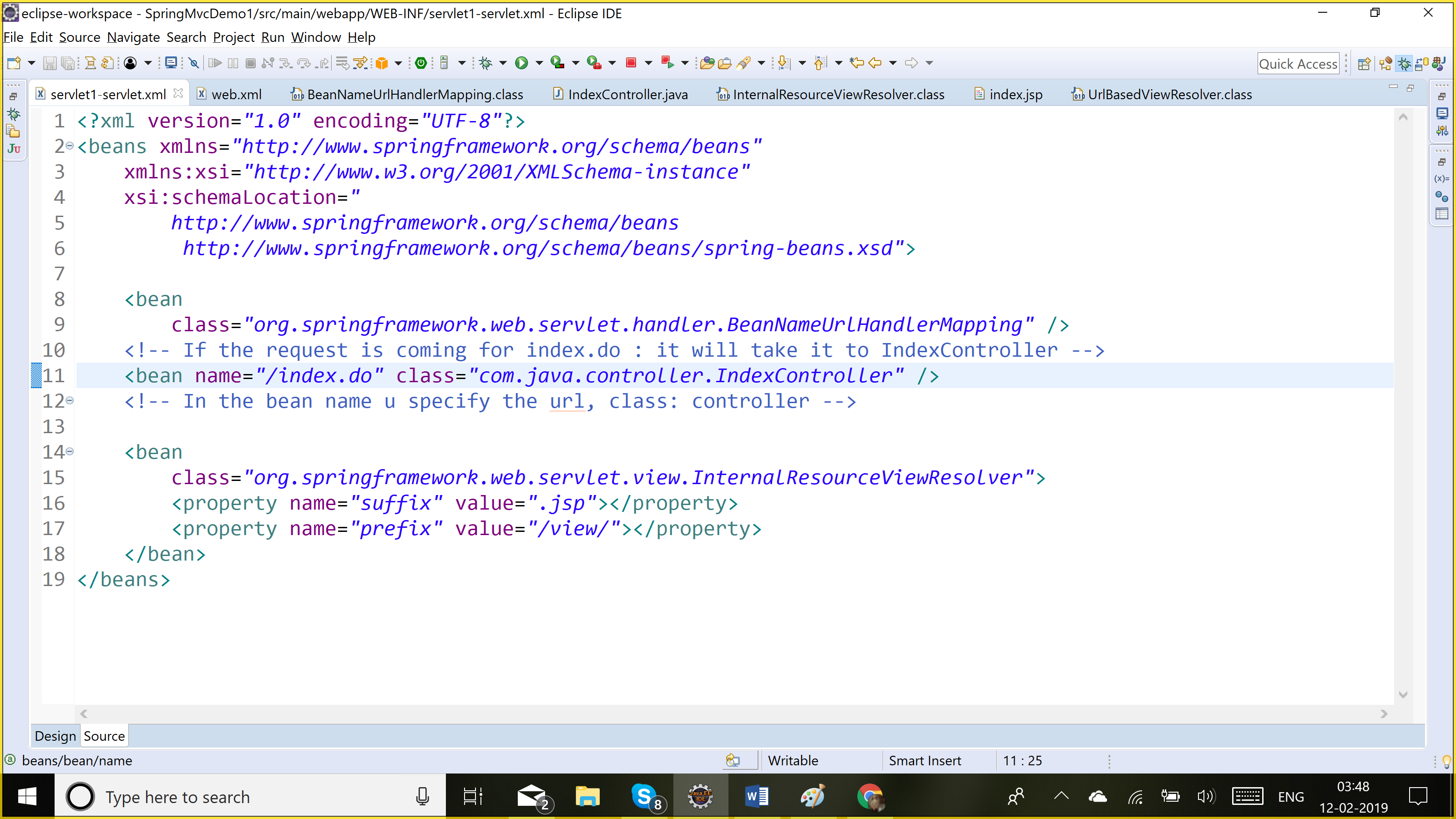
<servlet-name>-servlet.xml



<http://localhost:8080/SpringMvcDemo1/index.do>: DispatcherServlet -🡪 config file

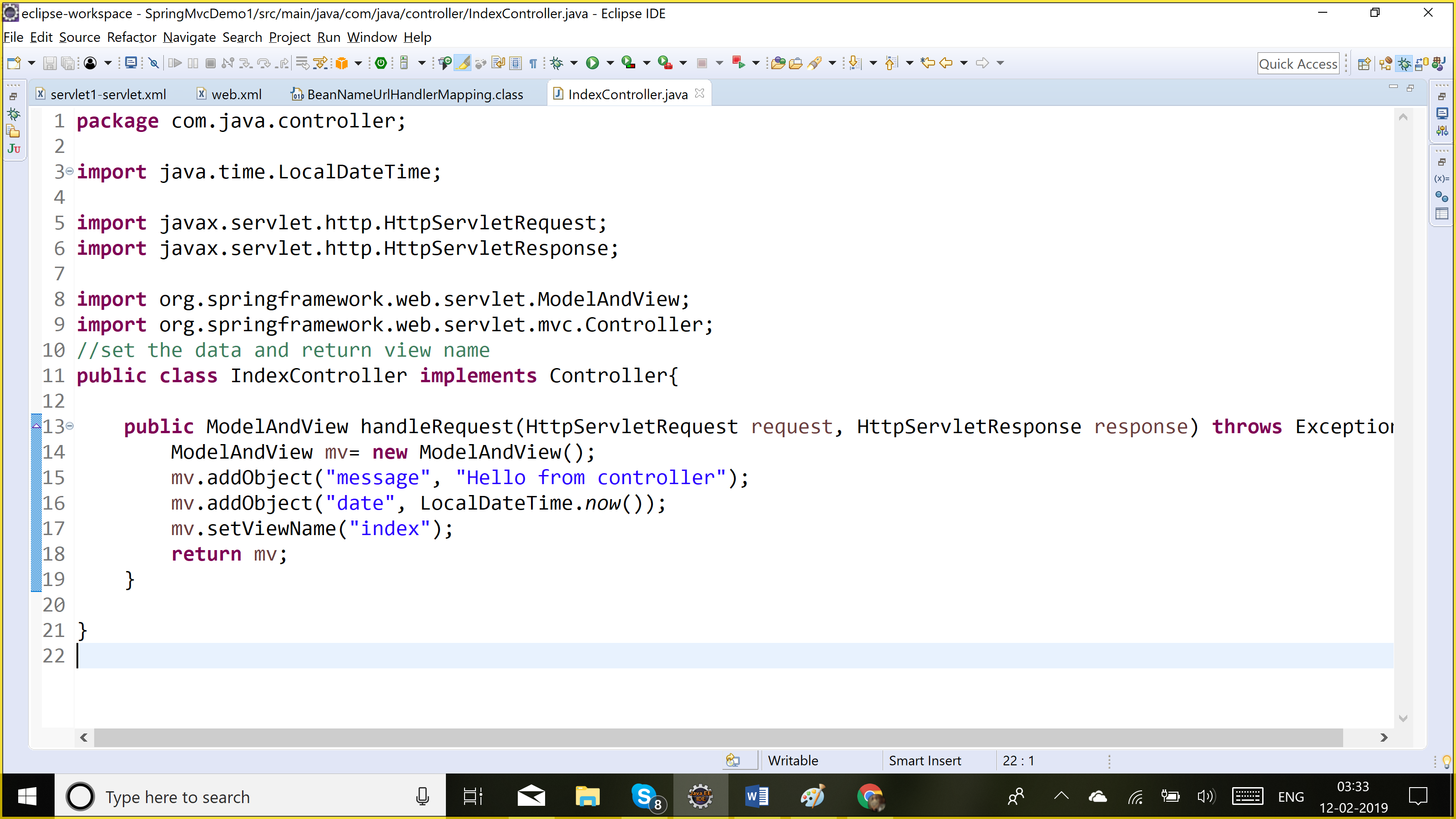
handlermapping: will tell how to find the controller



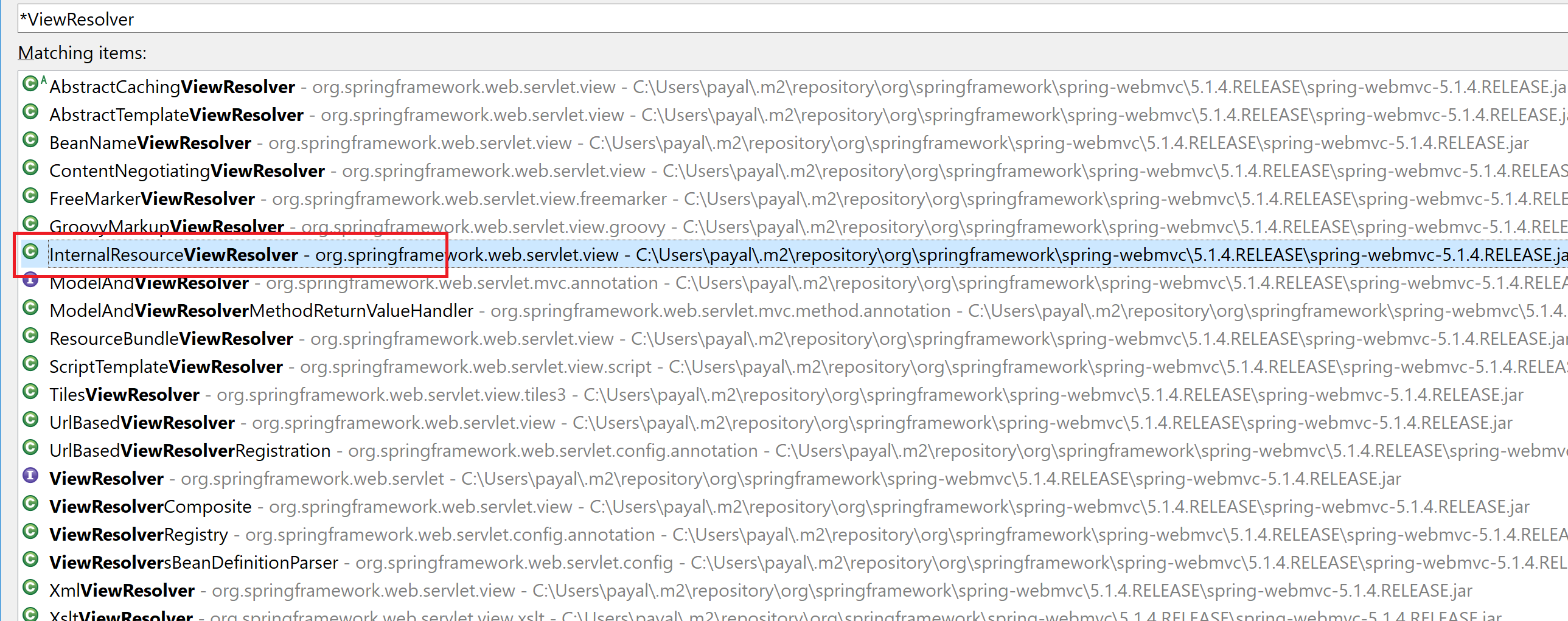


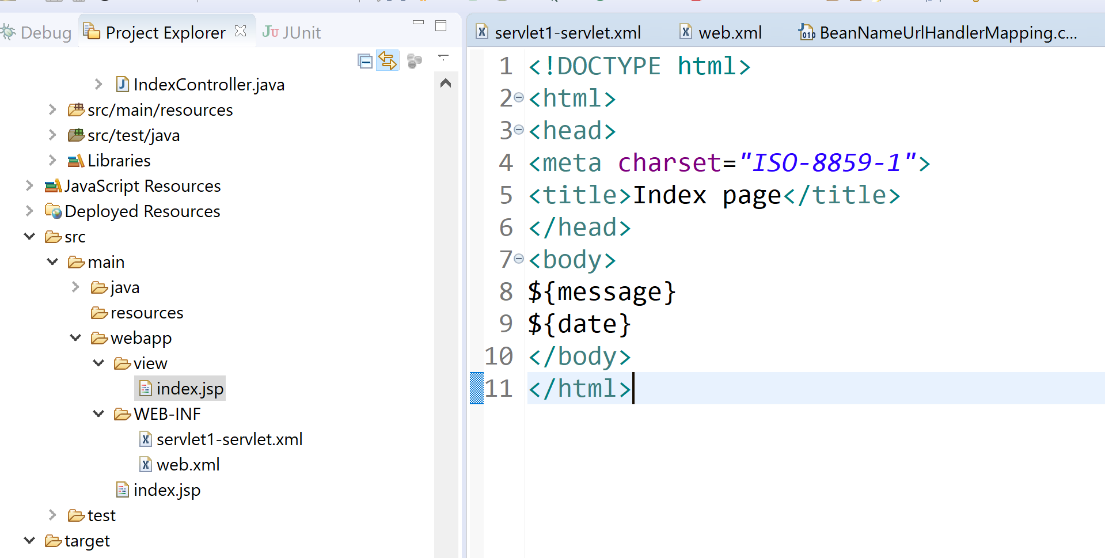
To create a servlet: Implements Servlet | extends GenericServlet| extends HttpServlet

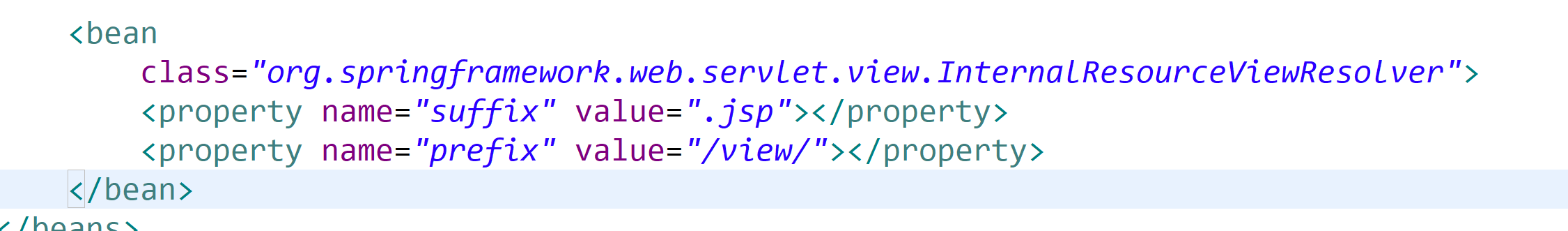
To create a spring controller: Implements Controller



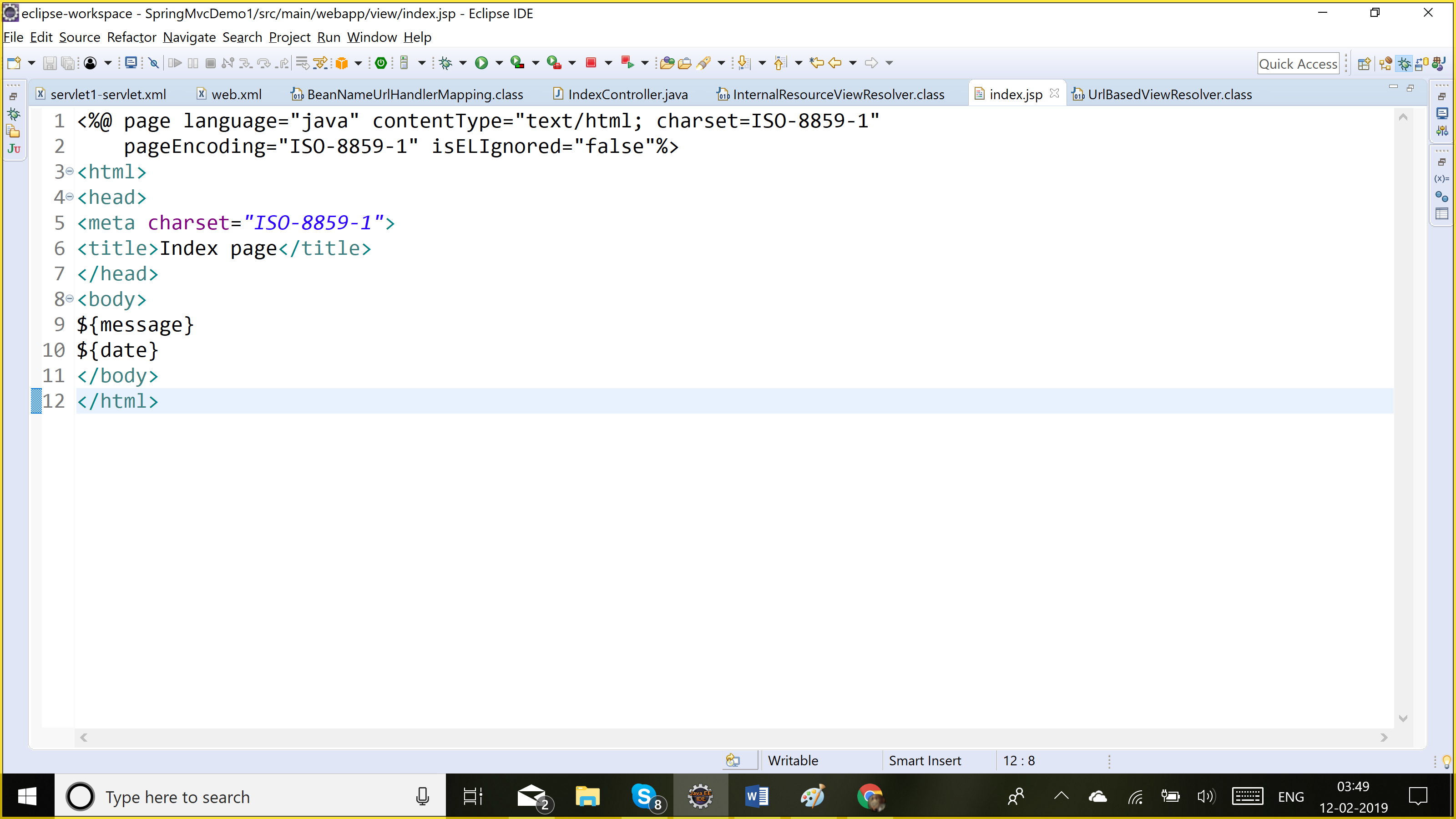
Dispatcher servlet has the view name, it takes help of viewResolver to find out where is ur view page (in which folder and what is the extension for it).







<http://localhost:8080/SpringMvcDemo1/index.jsp>: web container : display ur jsp page





Web.xml: urls have to be handled by spring container:

Dispatcherservlet : \*.do => spring container

Request -> spring config file -> controller mapping (beannameurlhandlermapping)

It will take it to the controller, controller shud set the data, return view name

Then dispatcher servlet will take help of view resolver to find view dir/ extension

It will display ur view page: index.jsp: web container will display it

Q) web.xml: welcome page

Login.html

Username

Password

Login

LoginController

Username&password: admin :

Welcome page : Welcome + admin

Error page: <Username> does not have access to this app