Spring MVC: web application (MVC based framework)

spring-webmvc module is also written on top of spring-core. (DI, autowiring)

u

MVC?? Model (dao+ dto) , View (UI: jsp/ html ), Controller (flow: servlets)

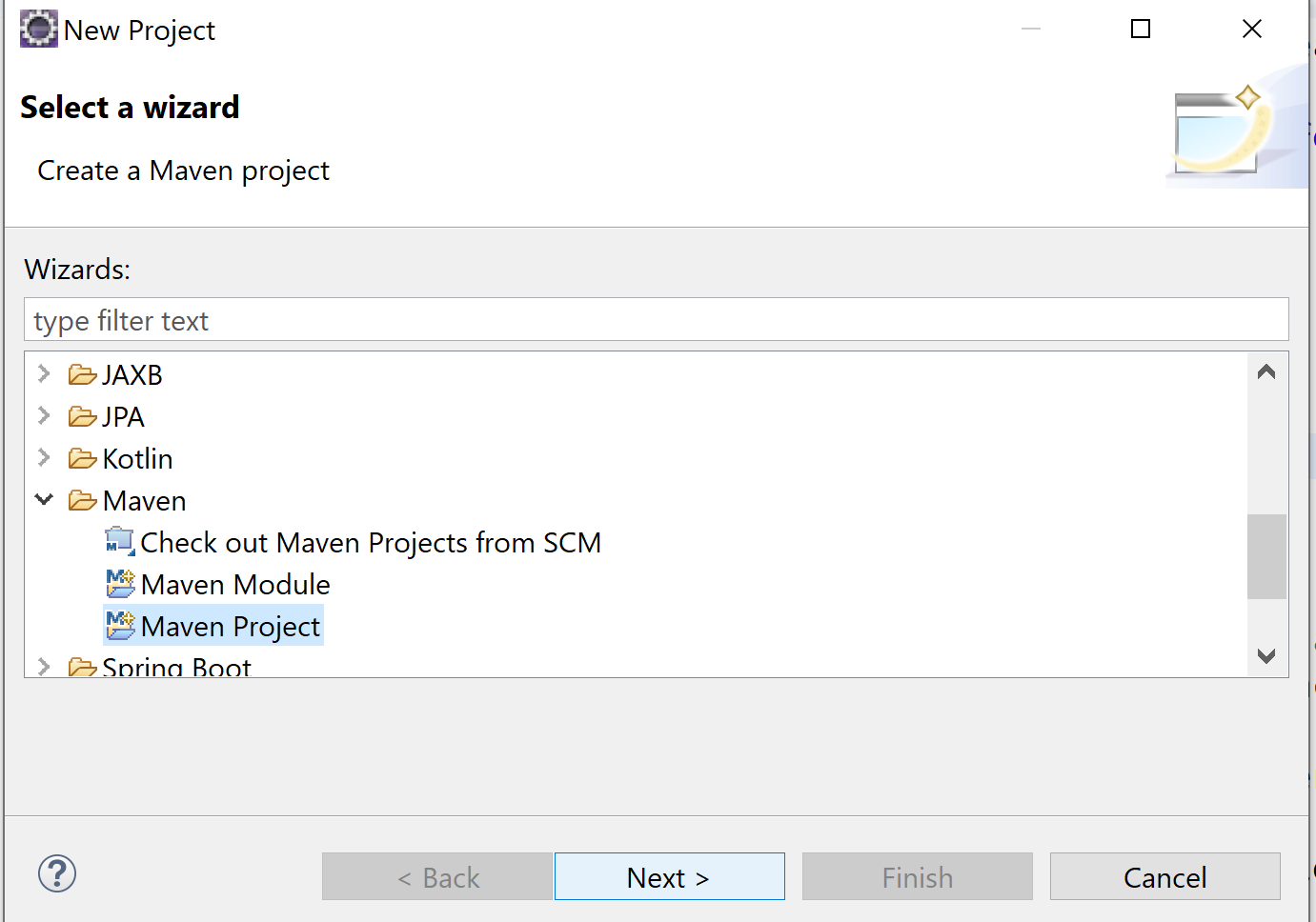
Spring MVC: Model (dto), View, Controller+ dao layer

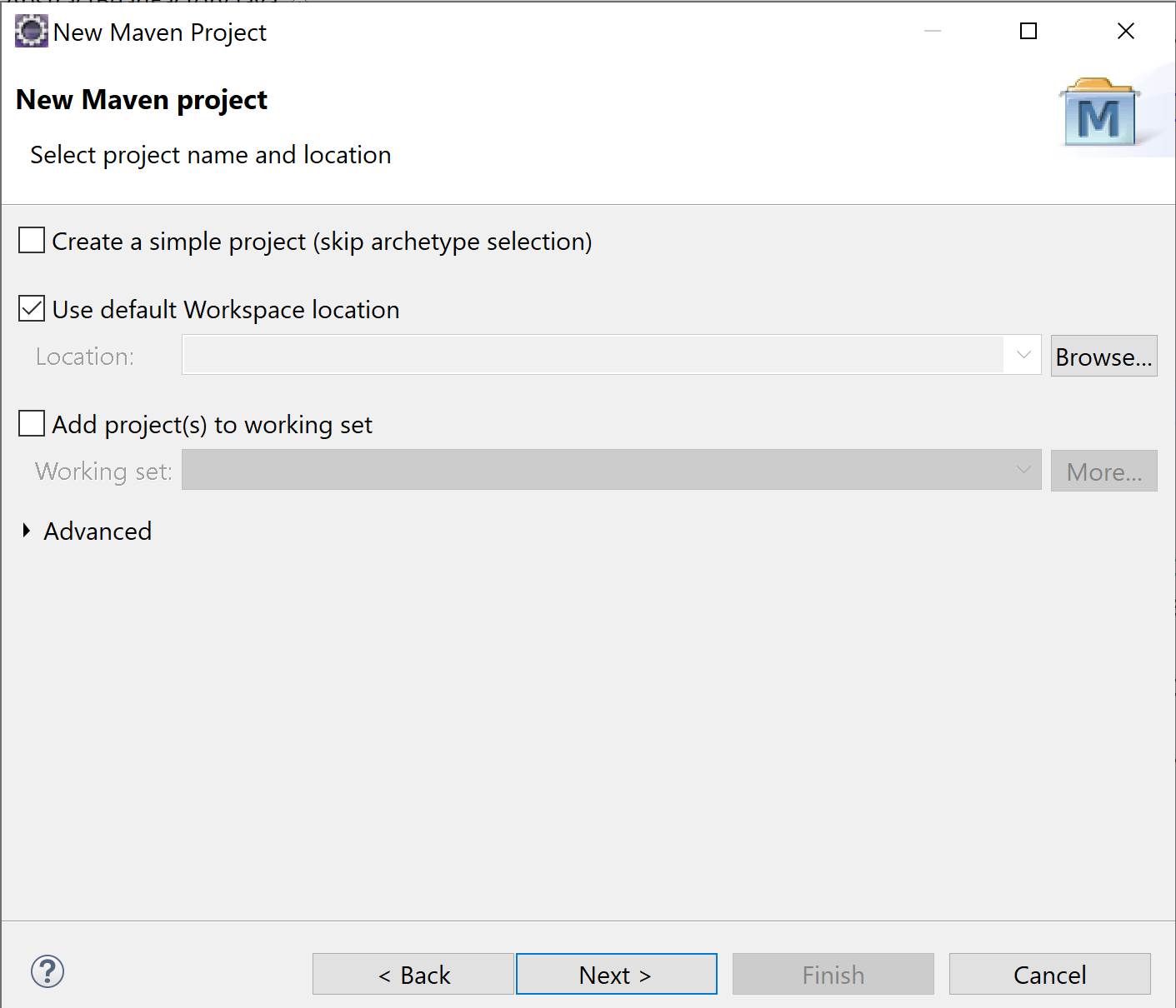
web application

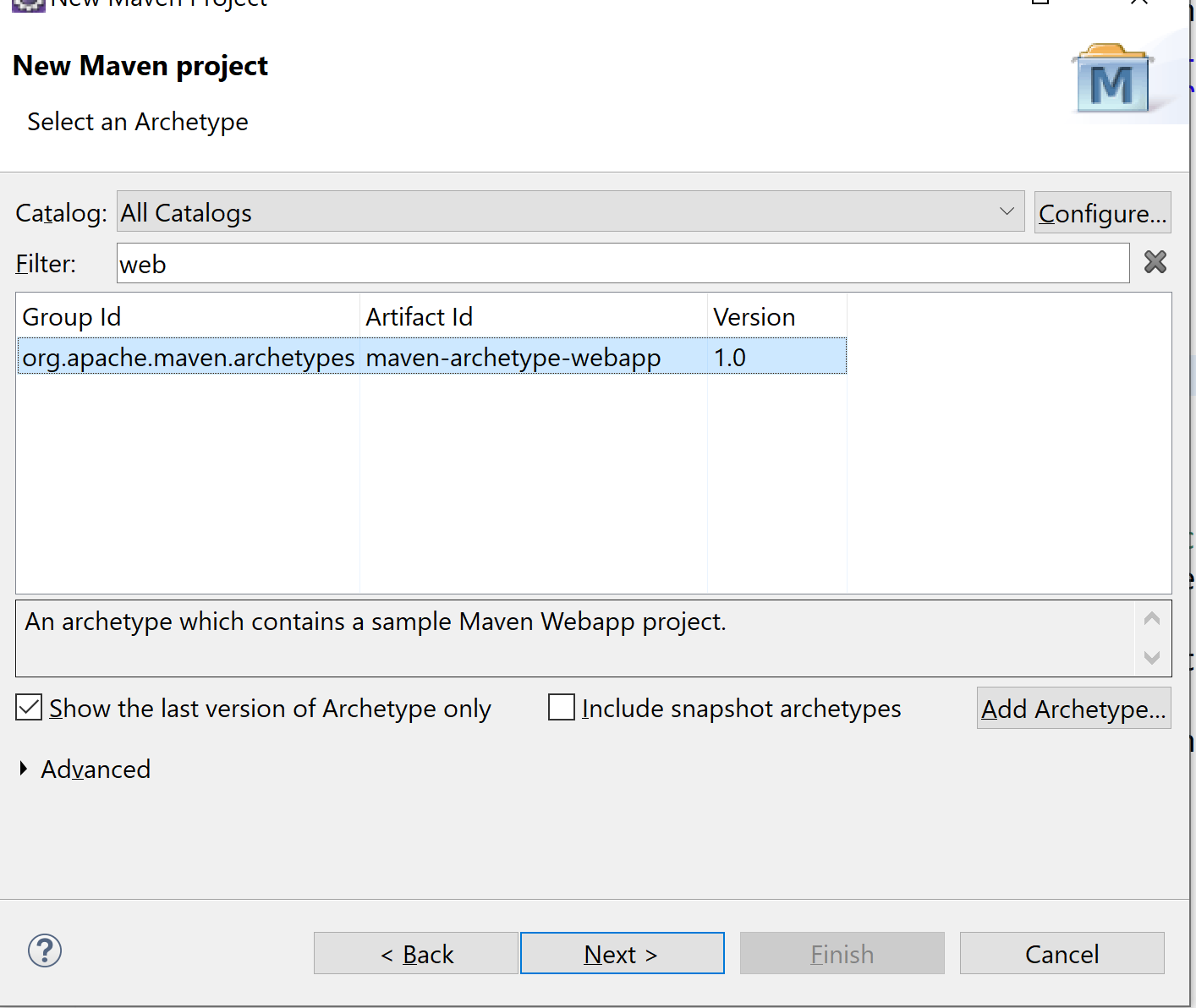
tomcat: web container (servlets)

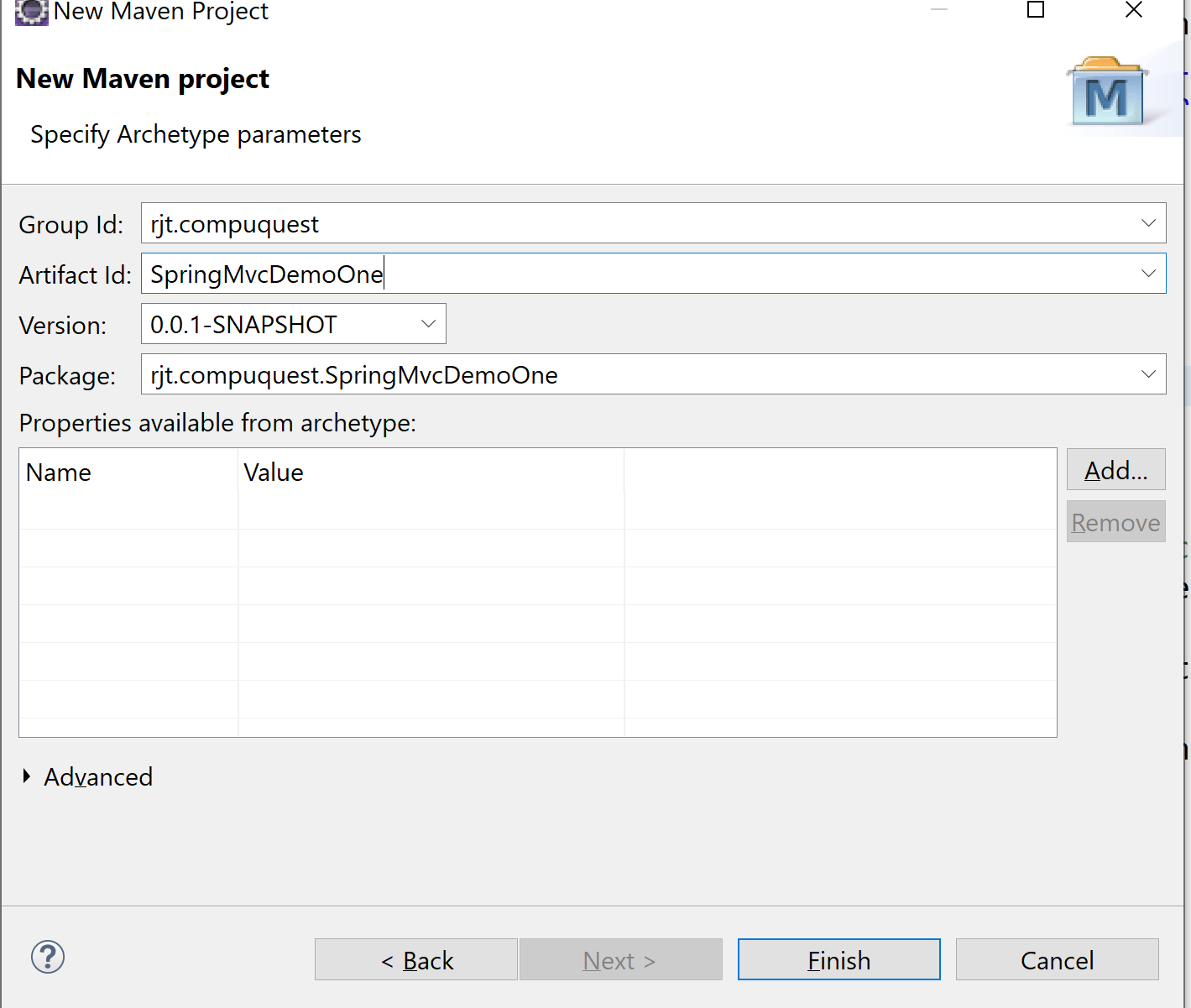
request -> web.xml:

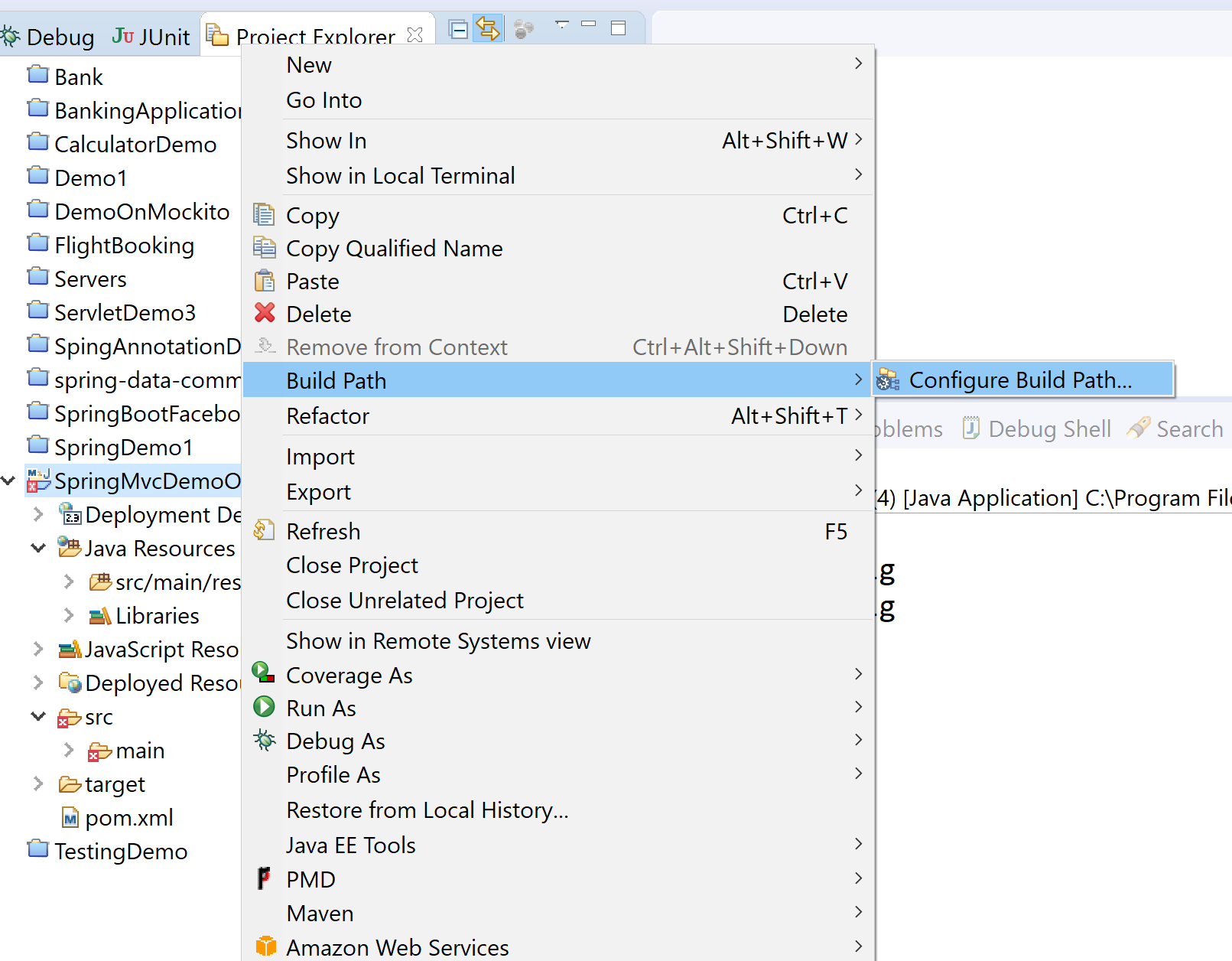
Spring web container

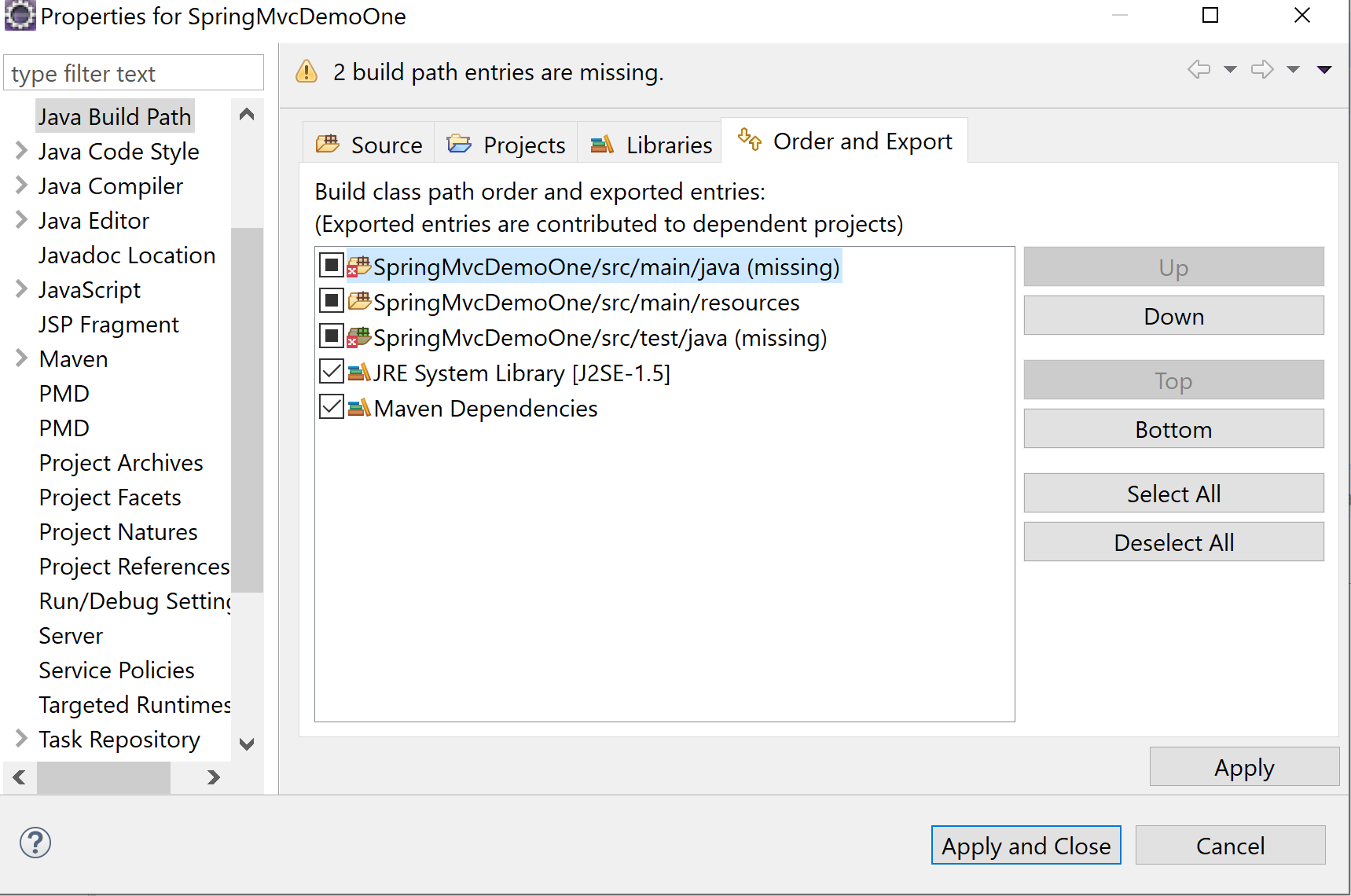


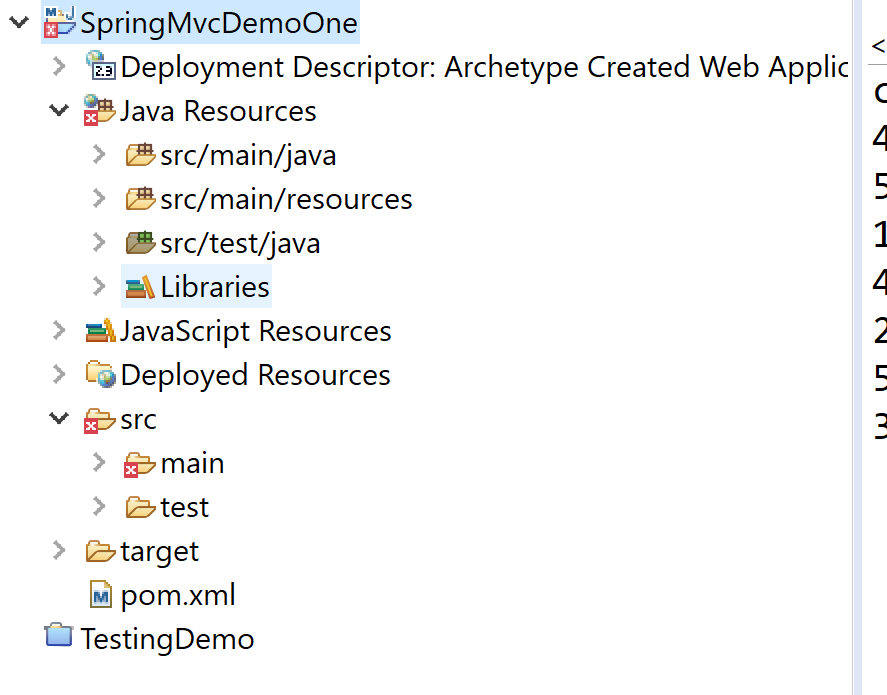


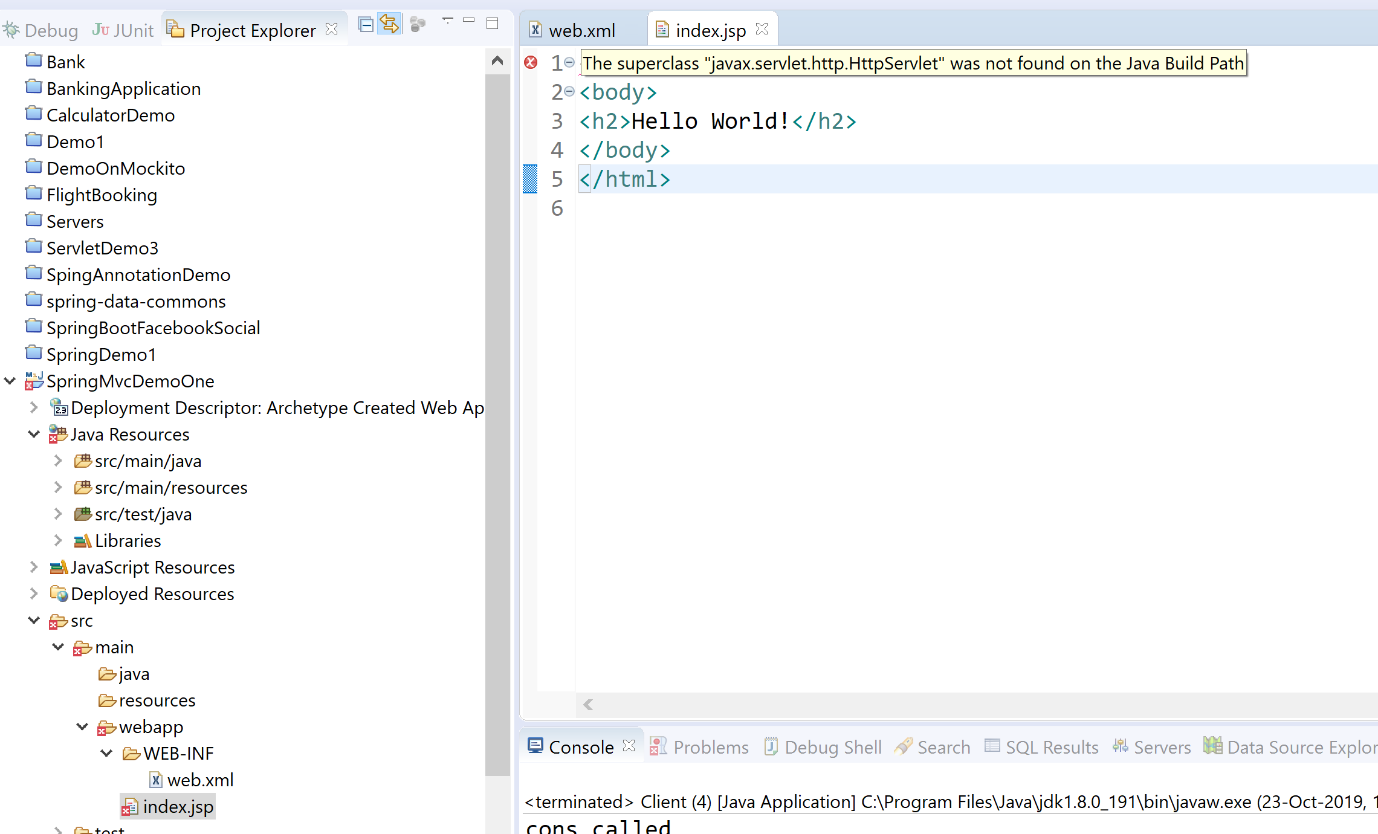












Jar: servlet-api jar/library

If I add tomcat to my project:

Properties -> Server -> add server & Targeted Runtimes -> Add server : not a good approach

Instead, add below dependency to the project:

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

<dependency>

<groupId>javax.servlet</groupId>

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

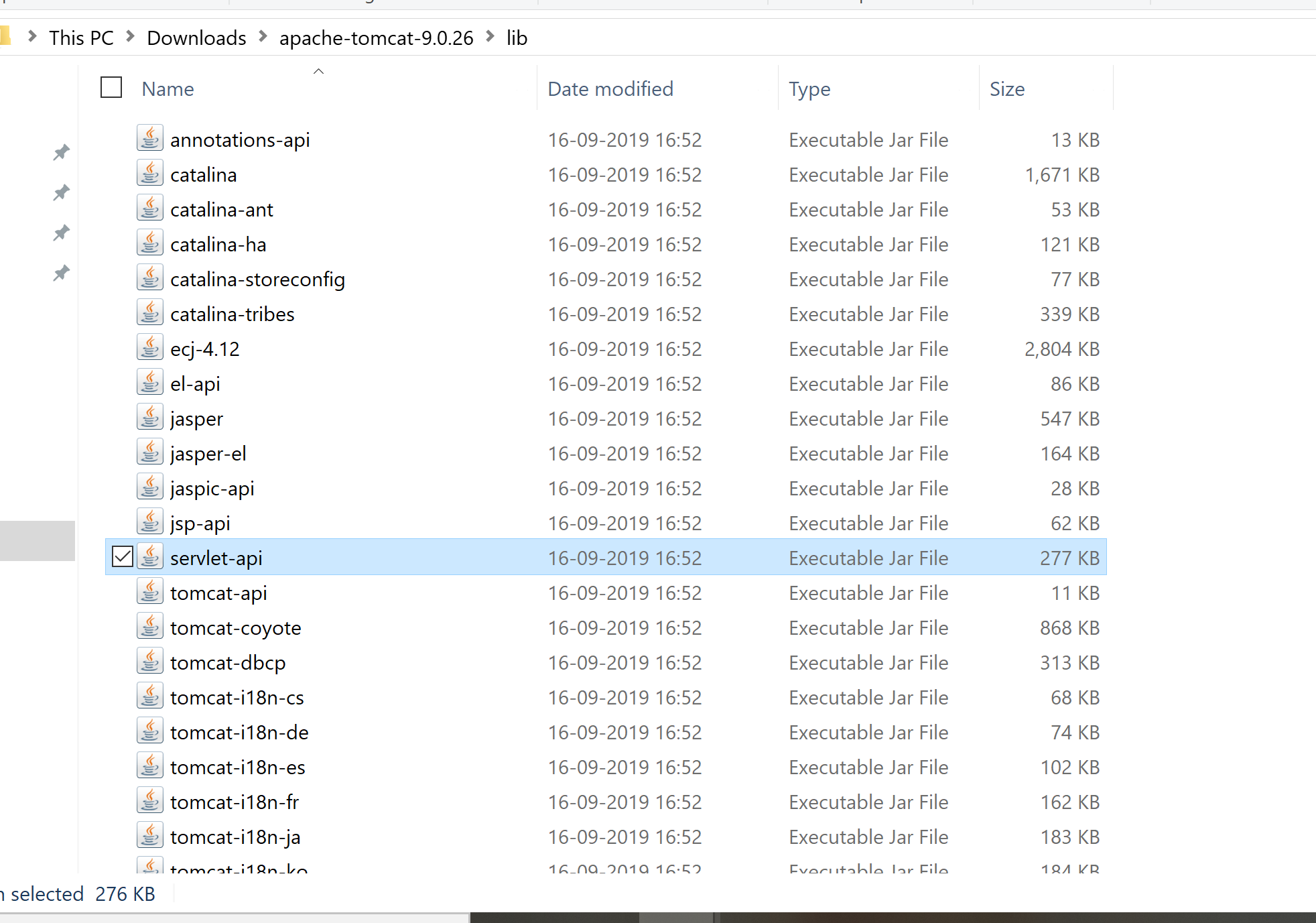
<version>4.0.1</version>

<scope>provided</scope>

</dependency>

**Maven scopes:**

1. Test: library available only for testing
2. Compile: Default scope. Library added at compile time, available during compilation, executing tests, runtime.
3. Runtime: available only at runtime. Ojdbc (driver class): Class.forName:
4. Provided: available at compile time, but not at runtime. Servlet-api



<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.2.0.RELEASE</version>

</dependency>

Web application: Lot of controllers: Common logic

Security

* Request -> front controller -> Ctrl1
  + - * + Ctrl2
        + Ctrl3
        + …
        + Ctlrn

Spring: DispatcherServlet: FrontController

<web-app>

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

<servlet>

<servlet-name>spring</servlet-name>

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

</servlet>

<servlet-mapping>

<servlet-name>spring</servlet-name>

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

</servlet-mapping>

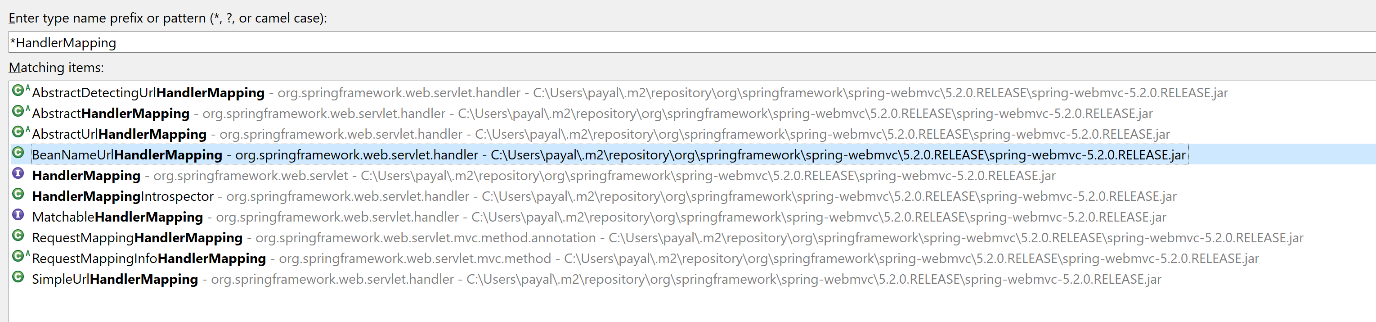
</web-app>

<!-- index.do: spring container.

index.html: web container, index.jsp -->

* It will be responsible for sending ur request to appropriate controller
  + Spring config file : HandlerMapping (to find out what controller )
    - **Name**: <servlet-name>-servlet.xml

/index.do??

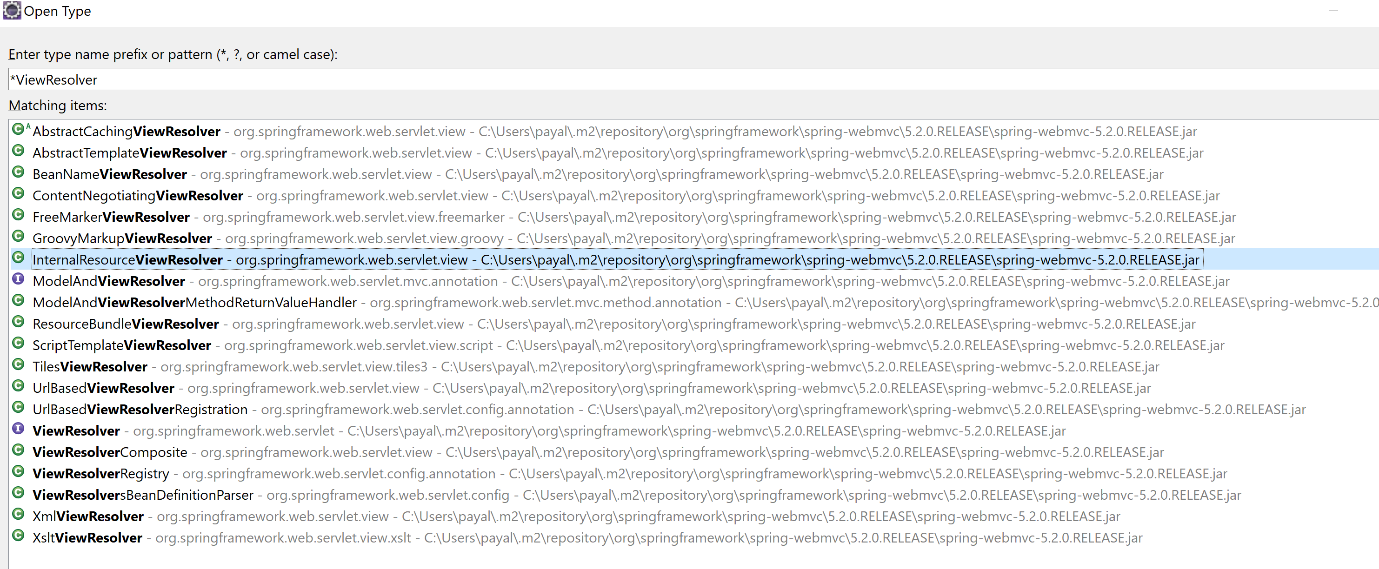
* + Ctrl1
  + Ctrl2
  + Ctrl3
* 

Bydefault, it uses your BeanNameUrlHandlerMapping

<bean name=”/index.do” class=”com.java.Ctrl1”/>

Controller should set the data to be displayed on the view page and should return the name of the view.

DispatcherServlet -> takes the request to find out where is the view page and what is the extension (ViewResolver)



ViewResolver shoud tell where is your view page and what is the extension for the view.

Then DispatcherServlet will forward the request to that view and the view would be rendered by the tomcat container if the url is controlled by webcontainer for the view page.

Controllers

WebApplicationContext.

* Request (index.do)-> web.xml -> mapping for spring container -> dispatcherServlet -> spring config file (<servletname>-servlet.xml) -> HandlerMapping (to find which controller to take the request to ) -> BeanNameUrlHandlerMapping -> Your controller -> set the model object/data which needs to be displayed on the view + return u the viewname) -> DispatcherServlet -> ViewResolver (InternalResourceViewResolver : prefix/suffix) -> /happy.jsp-> display this page
  + DispatcherServlet
  + HandlerMapping (Ctrl)
  + Ctrl-> View name, set ur data
  + ViewResolver : which view to be displayed
* Index.jsp -> web.xml -> web container -> display this page.

Login.jsp -> username, password -> Controller: validate -> valid -> Welcome page, invalid-> error.jsp