1. Cream o aplicatie ca Maeven Archetype, si alegem la Archetype org.apache.maven.archetypes:maven-archetype-webapp. **ASTEPTAM PUTIN SA SE GENEREZE TOT.**
2. Dam click dreapta pe proiect si Adaugam suport la Framewowkuri. Alegem Spring si Spring MVC. **DeBifam WebApp (care se bifeaza automat in principiu! )**
3. Src -> main cream un folder „java”,il marcam ca Sources root, si in el cream un packge cu o clasam unde addaugam @Controller si @RequestMapping
4. In WEB-INF cream un folder view si acolo crem pagini cu extensia .jsp
5. In WEB-INF cream contextApplication.xml. La <mvc:annotation-driven/> lasam spreang singur sa genereze xmlns:mvc, nu il adaugam manual

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xmlns:mvc="http://www.springframework.org/schema/mvc"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/cache http://www.springframework.org/schema/cache/spring-cache.xsd http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd http://www.springframework.org/schema/cache http://www.springframework.org/schema/cache/spring-cache.xsd http://www.springframework.org/schema/mvc https://www.springframework.org/schema/mvc/spring-mvc.xsd">  
  
 <context:component-scan base-package="com.app.test"/>  
 <mvc:annotation-driven/>  
 <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
 <property name="prefix" value="/WEB-INF/view/"/>  
 <property name="suffix" value=".jsp"/>  
 </bean>  
</beans>

1. Cream un web.xml tot acolo

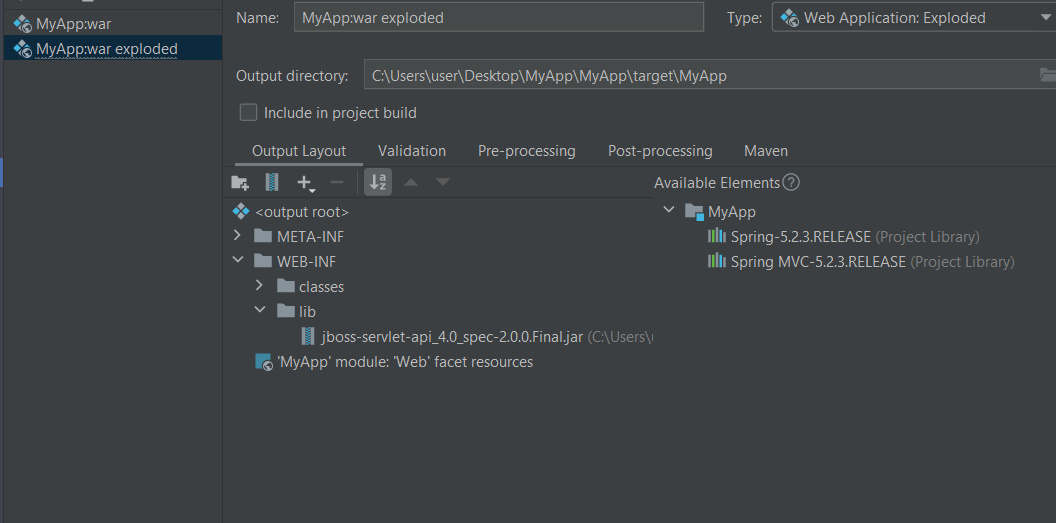
<!DOCTYPE web-app PUBLIC  
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"  
 "http://java.sun.com/dtd/web-app\_2\_3.dtd" >  
  
<web-app>  
 <display-name>Archetype Created Web Application</display-name>  
 <context-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>/WEB-INF/applicationContext.xml</param-value>  
 </context-param>  
 <listener>  
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
 </listener>  
 <servlet>  
 <servlet-name>dispatcher</servlet-name>  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 <init-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>WEB-INF/applicationContext.xml</param-value>  
 </init-param>  
 <load-on-startup>1</load-on-startup>  
 </servlet>  
 <servlet-mapping>  
 <servlet-name>dispatcher</servlet-name>  
 <url-pattern>/</url-pattern>  
 </servlet-mapping>  
</web-app>

1. Cream dispatcher-servlet.xml

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
</beans>

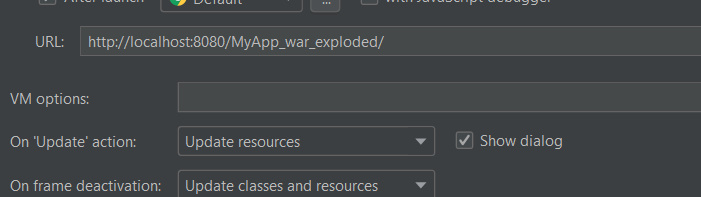
1. Pe proiectul nostru, dam click dreapta si alegem sa rulam ca Tom app, si lasam Intelij sa creeze singur TomCat-ul.

Sau cream noi un TomCat server, Local, si la artefactul exploded adaugam tot ce se afla la Available Elements





Ca TomCat sa faca automat modificarile, si sa nu trebuiasca mereu restart, punem:



Nu trebuie adaugat nimic in pom.xml

