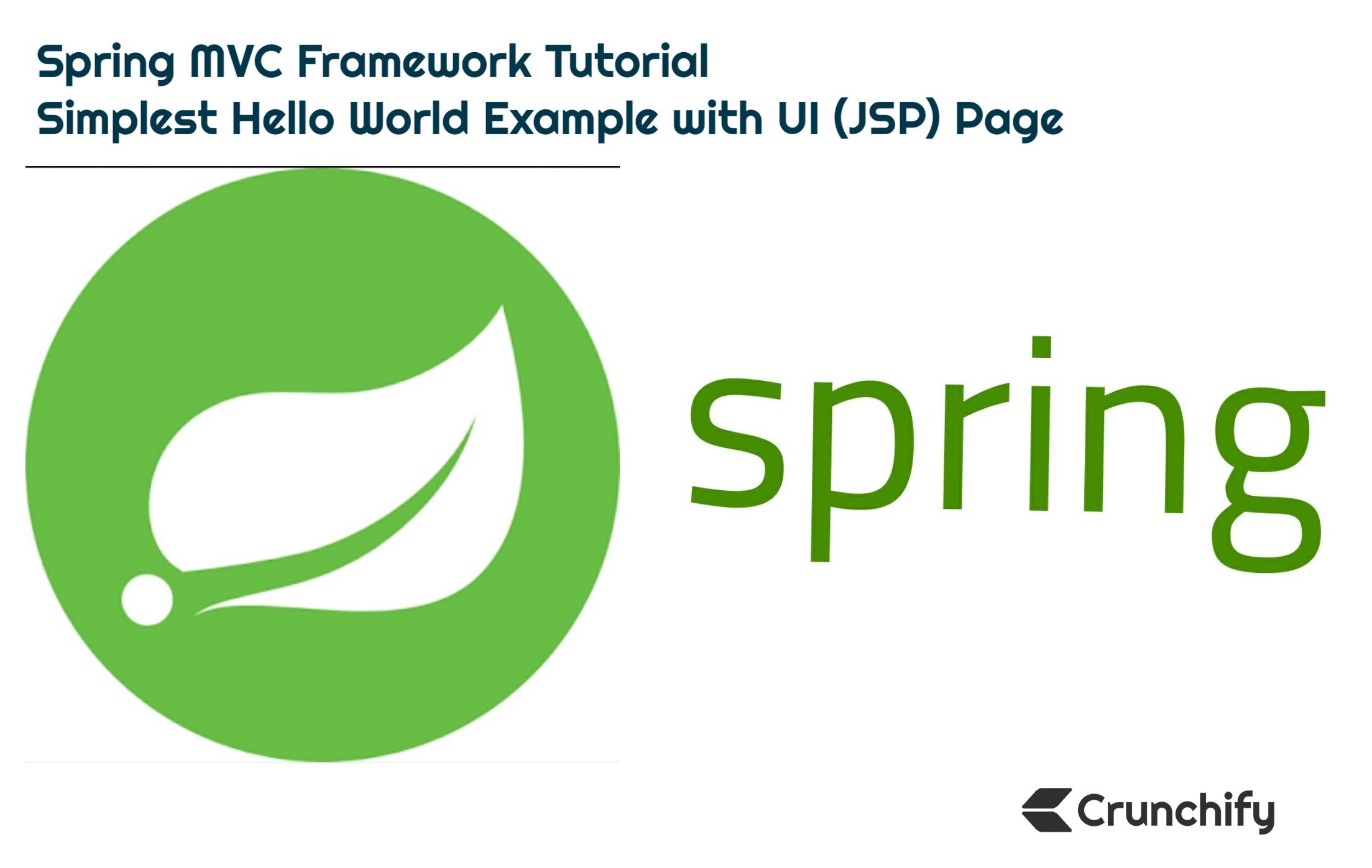
# Spring MVC Framework Tutorial Simplest Hello World Example with UI (JSP) Page

Last Updated on March 26th, 2019 by   App Shah



Do you have any one of below question?

* Fundamentals of Spring MVC 5.1.3
* Developing a [Spring Framework MVC](https://crunchify.com/category/spring-mvc/) 5 application step-by-step..
* [java](https://crunchify.com/memcached-java-client-tutorial-using-com-whalin-memcached-java-client-and-com-googlecode-xmemcached-libraries/) – Spring MVC tutorial from the scratch
* Spring MVC Fast Tutorial
* Spring MVC Framework Tutorial
* First Spring MVC application tutorial
* Spring 5.x MVC Tutorials, [AJAX Demo](https://crunchify.com/how-to-use-ajax-jquery-in-spring-web-mvc-jsp-example/), [jQuery Demo](https://crunchify.com/jquery-very-simple-showhide-panel-on-mouse-click-event/), [JavaScript Demo](https://crunchify.com/paypal-java-sdk-tutorial-authorization-call/), Tips & Tricks Spring 5 MVC

Then you are at right place. Here I’ll demonstrate simple Spring MVC framework for building web applications.

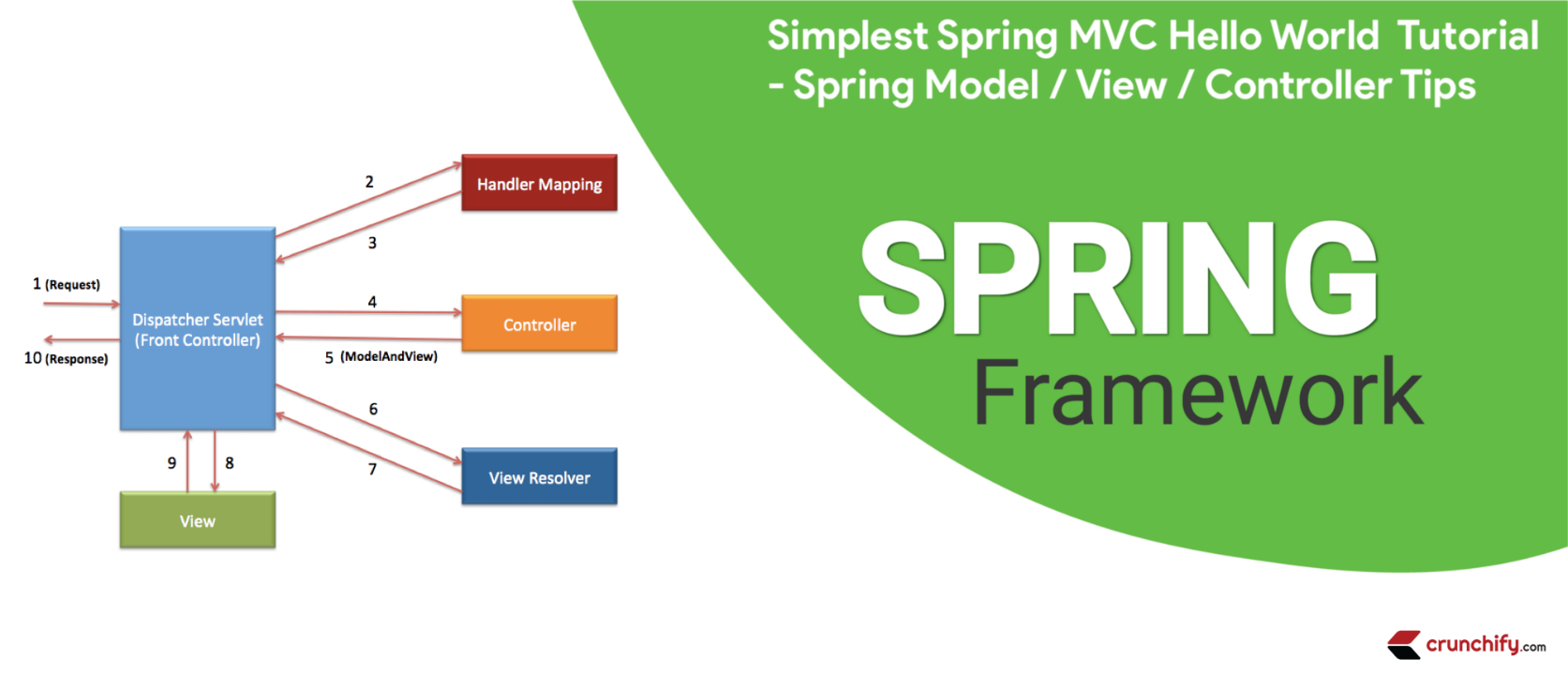
## 1.Setup Environment

First thing first – Let’s Setup Environment

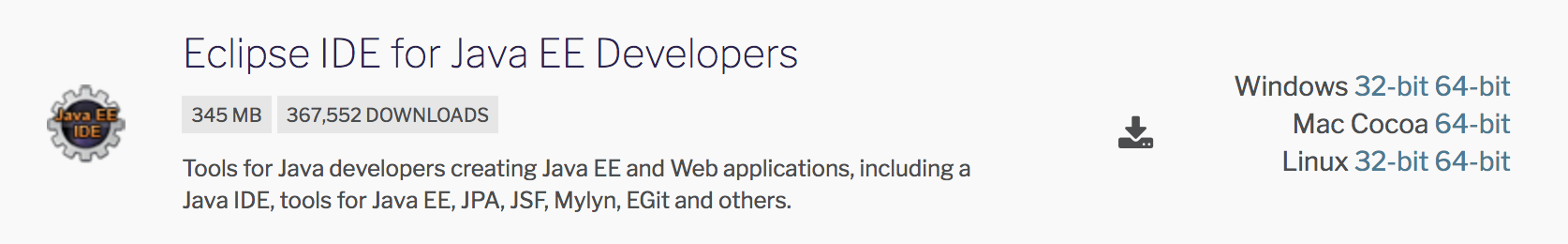
Tutorial last updated: 12/13/2018 with below Tool versions.

I’m using below tools which you may need to download if you don’t have already.

1. Tomcat 9.0.10 – Download latest Apache Tomcat from this [link](https://tomcat.apache.org/download-90.cgi).
2. Make sure you download Eclipse IDE for Java EE Developers (Photon v4.8.0) – Download [link](http://www.eclipse.org/downloads/eclipse-packages/). (diagram below)
3. Spring 5.1.3 (No download required) – we will use [Maven dependency](https://crunchify.com/how-to-import-all-spring-mvc-dependencies-to-your-maven-project/).
4. JDK 10.0.2 – Download [link](https://www.oracle.com/technetwork/java/javase/downloads/index.html).

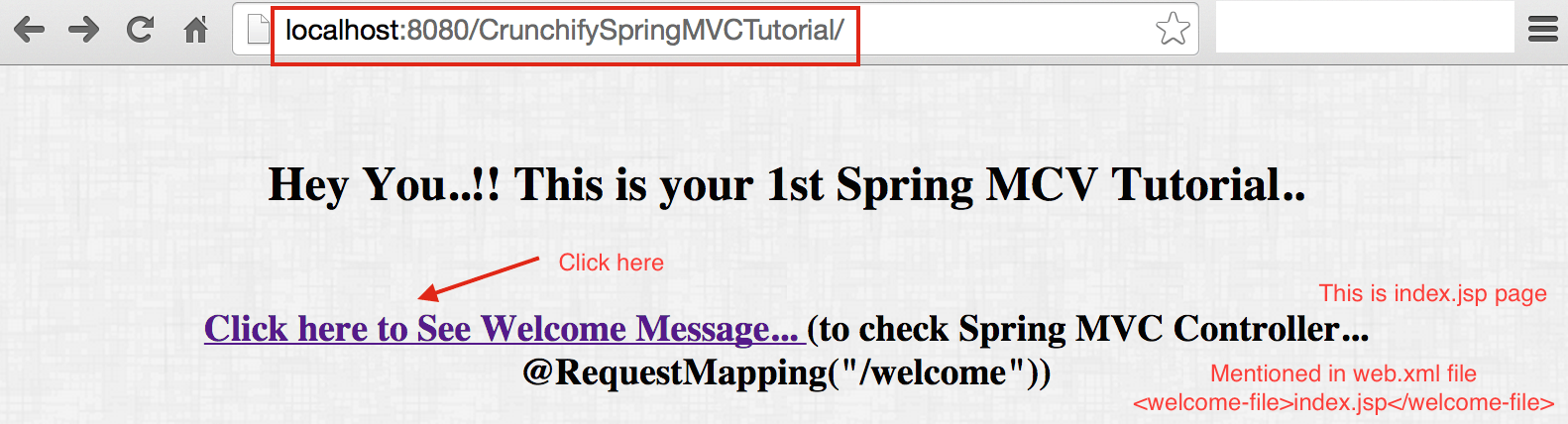


Make sure you download Java EE:

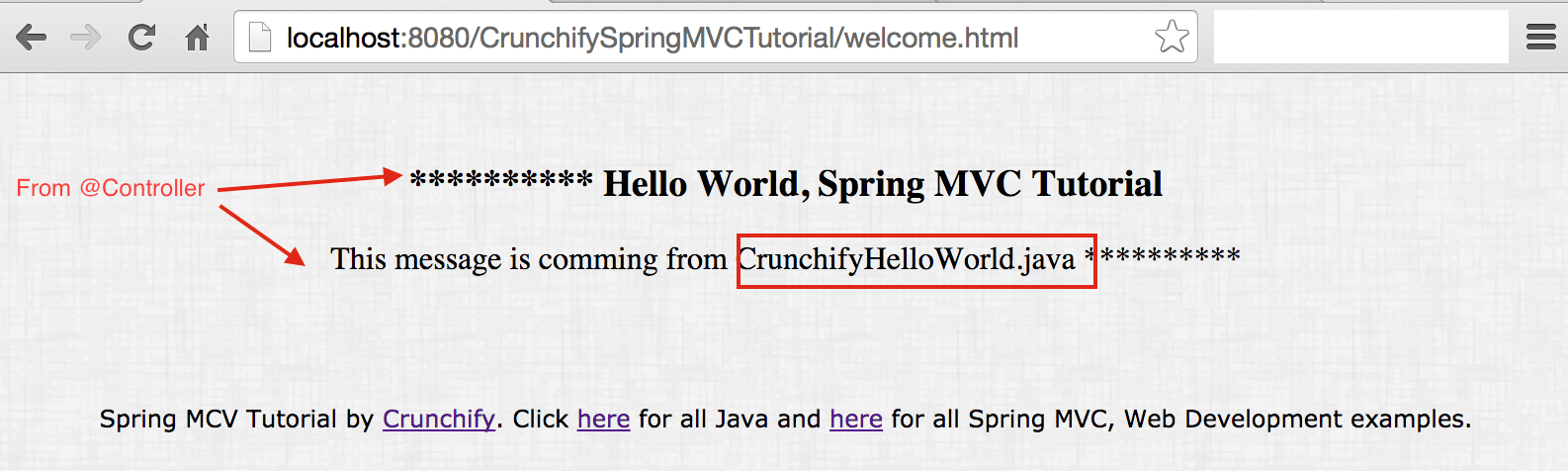


Main goal for this tutorial to create Spring MVC Application in the **simplest way**. This is how our application result will look like. This is a final result once you complete all below steps.

### Here is a final result: Welcome page ==> index.jsp



### Result returns from Controller Class

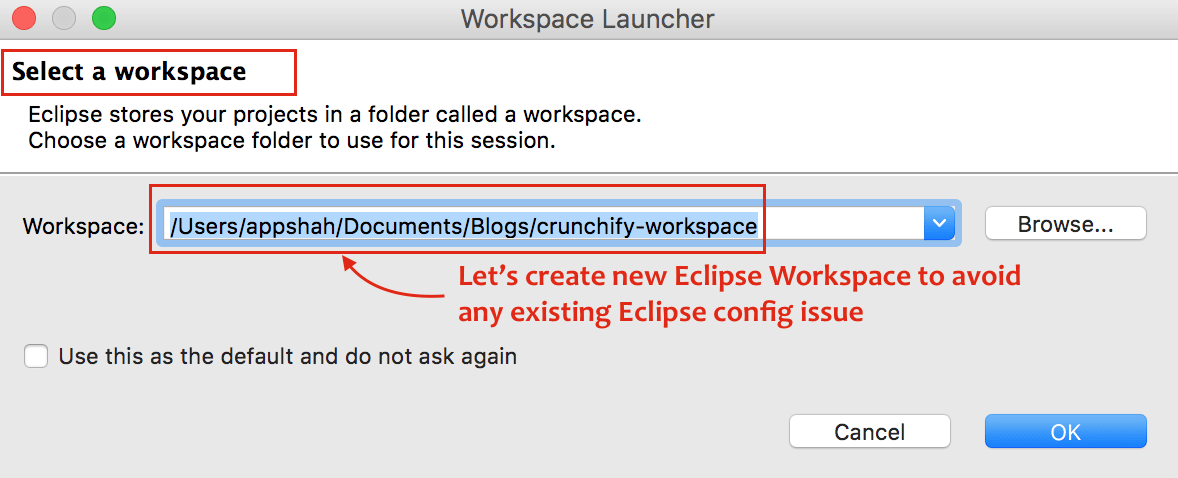


## 2.Started on Tutorial

Now Let’s get started on Tutorial

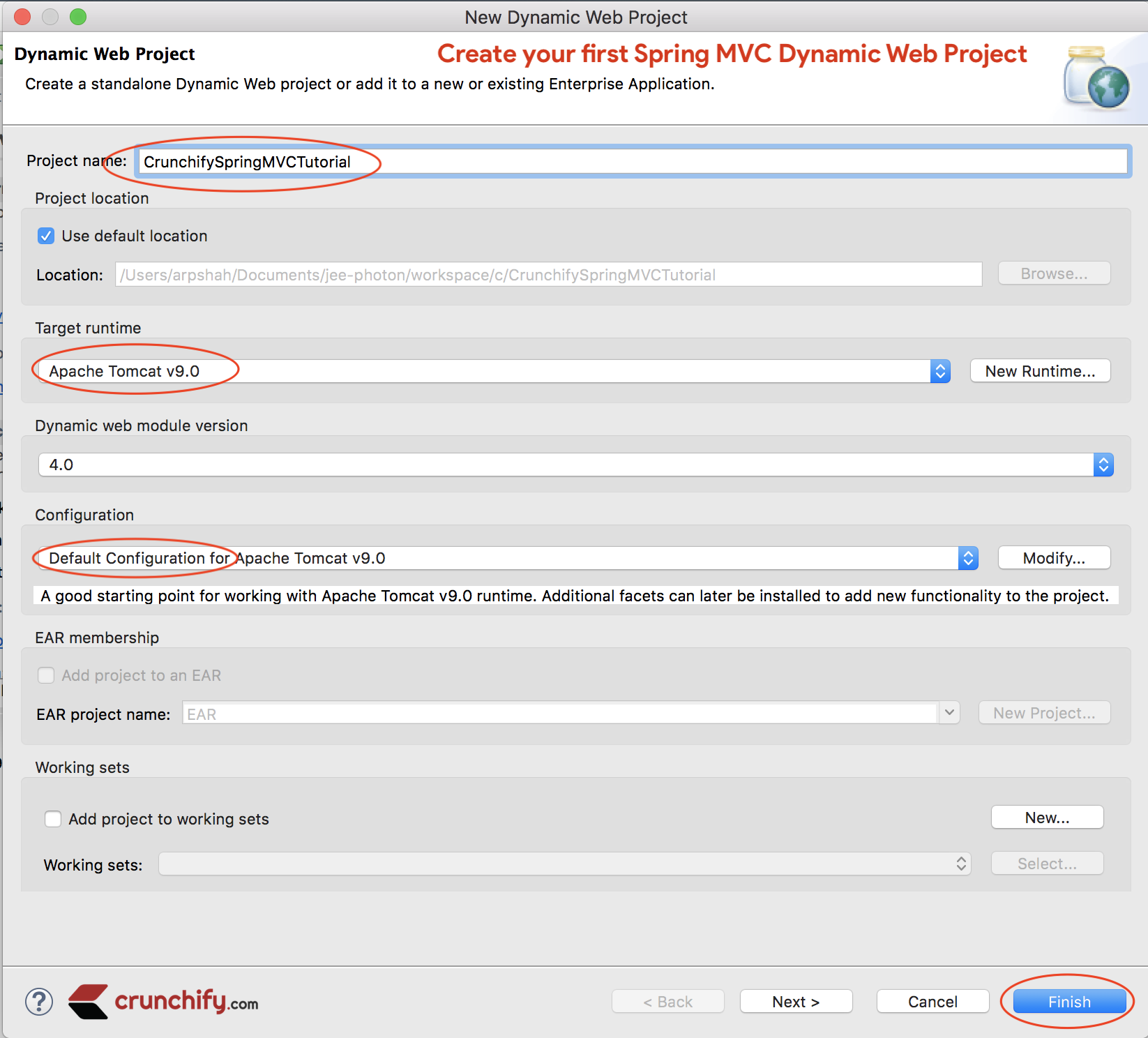
### Step-1 Create Workspace

* Open [Eclipse](https://crunchify.com/missing-maven-settings-xml-file-for-your-eclipse-what-if-you-need-two-settings-xml-file-for-work-personal-workspace/)
* Create New Eclipse Workspace – This is must to avoid any existing workspace config issue.



### Step-2 Create Project

* Click on File
* Click on New
* Choose Dynamic Web Project
* One popup window, Provide Project Name: CrunchifySpringMVCTutorial
* Make sure you use Target Runtime as Apache Tomcat 9.0
  + If you don’t see Target Runtime then [follow these steps](https://crunchify.com/step-by-step-guide-to-setup-and-install-apache-tomcat-server-in-eclipse-development-environment-ide/)
* Choose Configuration as Default Configuration

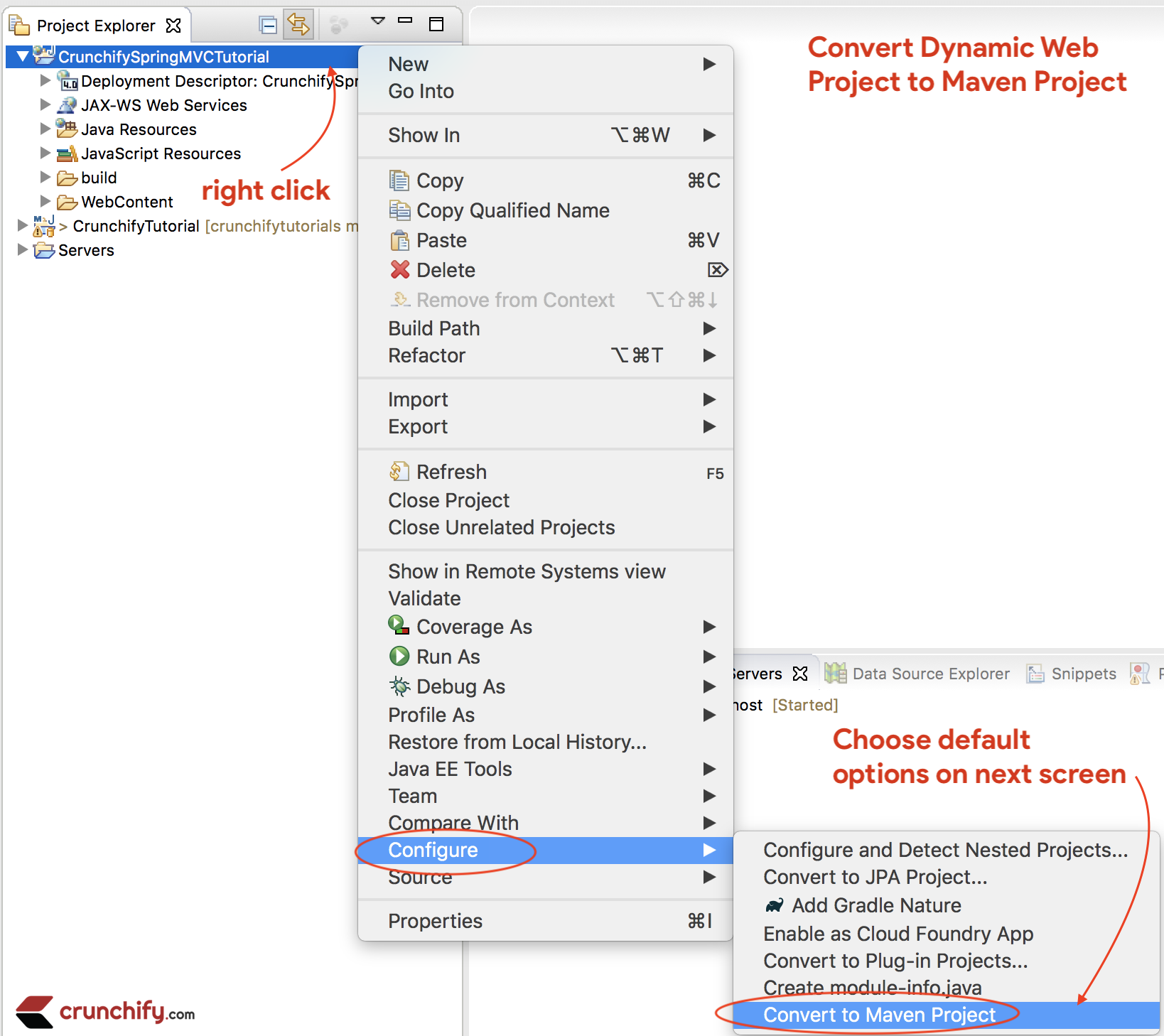


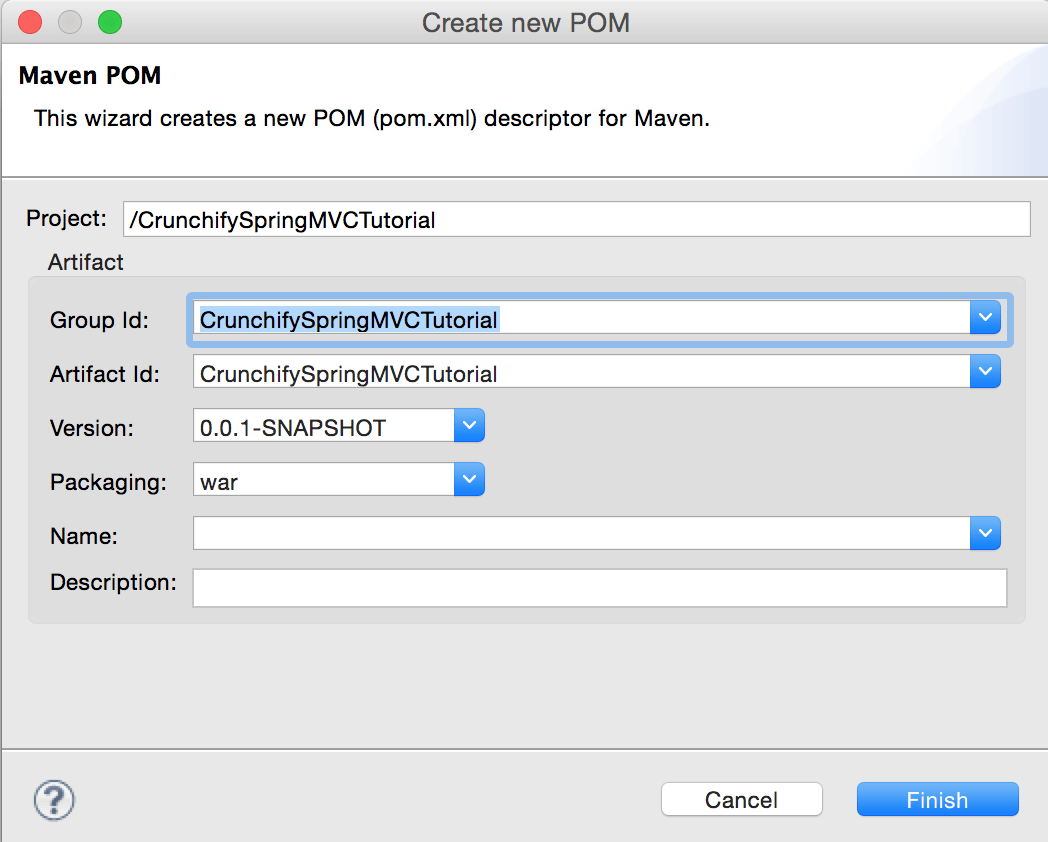
### Step-3 Convert to maven Project

Convert Project to [Maven Project](https://crunchify.com/how-to-create-a-war-file-from-eclipse-using-maven-plugin-apache-maven-war-plugin-usage/) to add all required Spring MVC dependencies to project.

#### Steps:

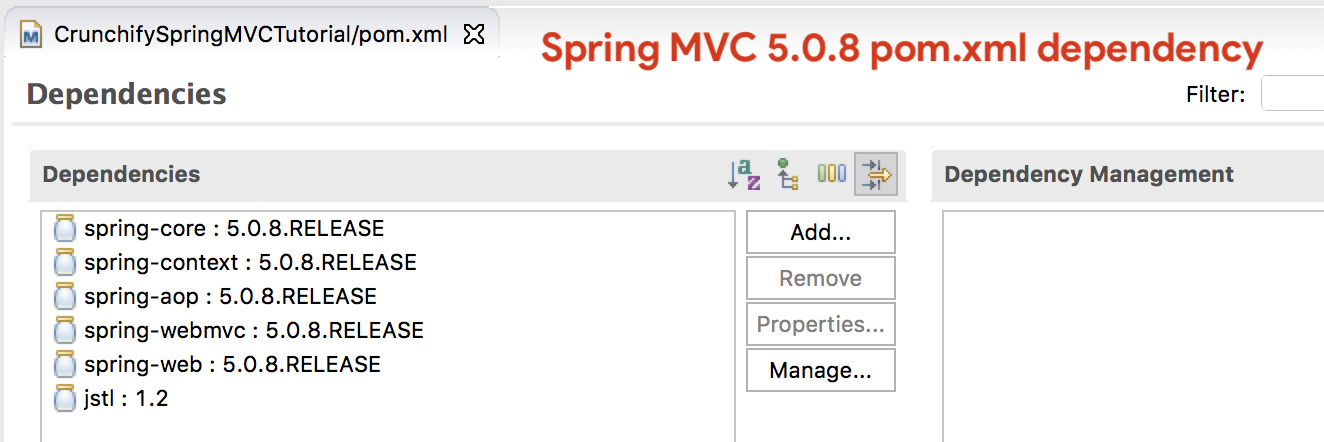
* Right click on project
* Configure
* Convert to Maven project





### Step-4 Add dependencies to project

Open pom.xml file and add below jar dependencies to project.



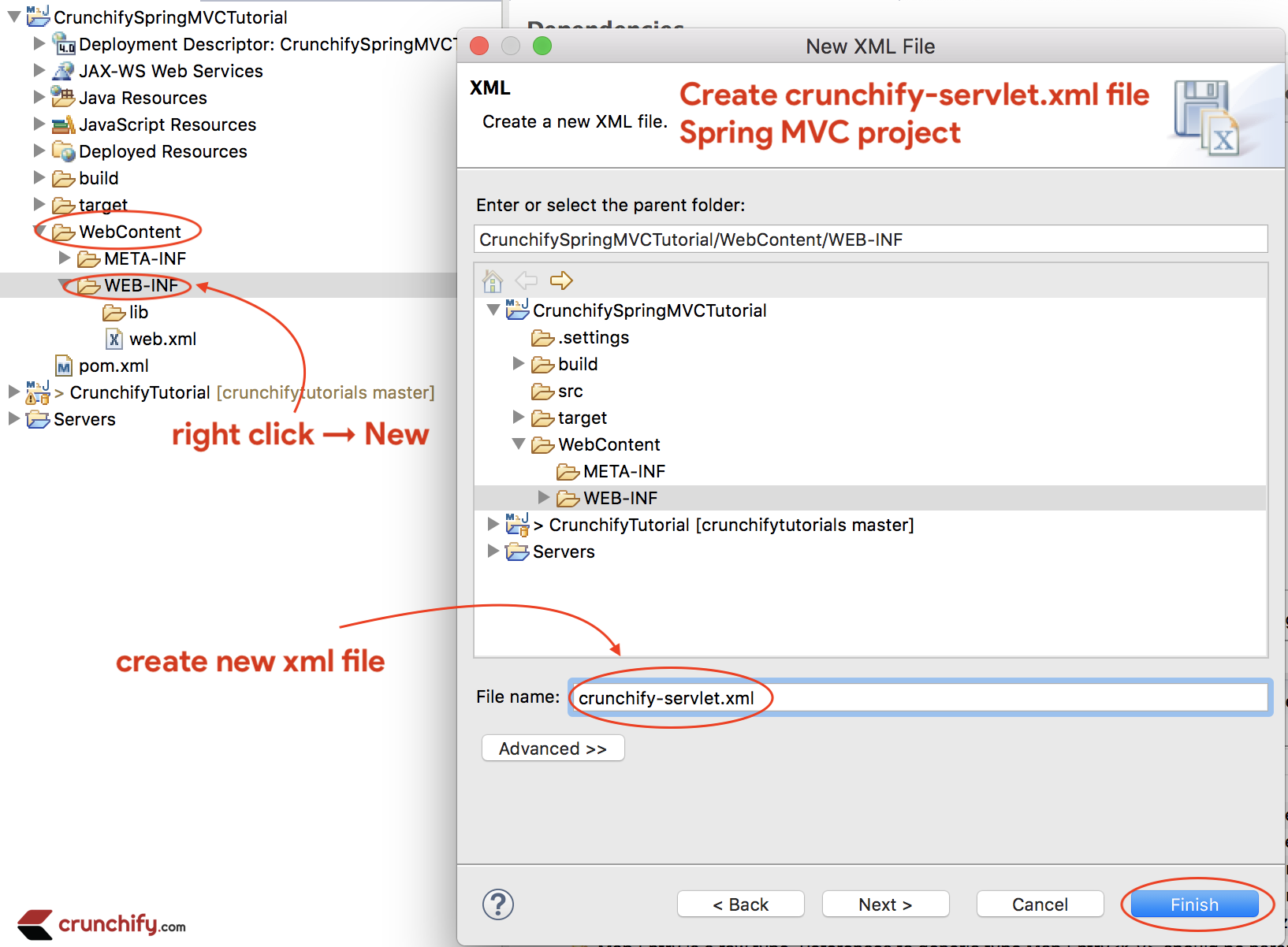
NOTE: Here is my pom.xml file. Make sure you update Java version to 10 if you haven’t yet moved to JDK 10. We will keep updating this tutorial to latest Spring MVC version.

So below pom.xml file may have different (latest) version of Spring MVC dependencies than above image

|  |
| --- |
| <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>CrunchifySpringMVCTutorial</groupId>  <artifactId>CrunchifySpringMVCTutorial</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>war</packaging>  <build>  <sourceDirectory>src</sourceDirectory>  <plugins>  <plugin>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.8.0</version>  <configuration>  <release>10</release>  </configuration>  </plugin>  <plugin>  <artifactId>maven-war-plugin</artifactId>  <version>3.2.1</version>  <configuration>  <warSourceDirectory>WebContent</warSourceDirectory>  </configuration>  </plugin>  </plugins>  </build>  <dependencies>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-core</artifactId>  <version>5.1.3.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>5.1.3.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aop</artifactId>  <version>5.1.3.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>5.1.3.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-web</artifactId>  <version>5.1.3.RELEASE</version>  </dependency>    <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  </dependencies>  </project>  如果下载jar包出错，可以试着更改.m2/settings.xml  <?xml version="1.0" encoding="UTF-8"?>  <settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">  <!-- 默认的值是${user.home}/.m2/repository -->  <localRepository>C:/Users/lzh/.m2/repository</localRepository>  <!-- 如果Maven要试图与用户交互来得到输入就设置为true，否则就设置为false，默认为true。 -->  <interactiveMode>true</interactiveMode>  <!-- 如果Maven使用${user.home}/.m2/plugin-registry.xml来管理plugin的版本，就设置为true，默认为false。 -->  <usePluginRegistry>false</usePluginRegistry>  <!-- 如果构建系统要在离线模式下工作，设置为true，默认为false。 如果构建服务器因为网络故障或者安全问题不能与远程仓库相连，那么这个设置是非常有用的。 -->  <offline>false</offline>  <mirrors>  <mirror>  <id>nexus-aliyun</id>  <mirrorOf>central</mirrorOf>  <name>Nexus aliyun</name>  <url>http://maven.aliyun.com/nexus/content/groups/public/</url>  </mirror>  </mirrors>  <!-- settings.xml中的profile是pom.xml中的profile的简洁形式。 它包含了激活(activation)，仓库(repositories)，插件仓库(pluginRepositories)和属性(properties)元素。  profile元素仅包含这四个元素是因为他们涉及到整个的构建系统，而不是个别的POM配置。 如果settings中的profile被激活，那么它的值将重载POM或者profiles.xml中的任何相等ID的profiles。 -->  <profiles>  <profile>  <id>default</id>  <activation>  <activeByDefault>true</activeByDefault>  <jdk>1.8</jdk>  </activation>  <repositories>  <repository>  <id>spring-milestone</id>  <name>Spring Milestone Repository</name>  <url>http://repo.spring.io/milestone</url>  <releases>  <enabled>true</enabled>  </releases>  <snapshots>  <enabled>false</enabled>  </snapshots>  <layout>default</layout>  </repository>  <repository>  <id>spring-snapshot</id>  <name>Spring Snapshot Repository</name>  <url>http://repo.spring.io/snapshot</url>  <releases>  <enabled>false</enabled>  </releases>  <snapshots>  <enabled>true</enabled>  </snapshots>  <layout>default</layout>  </repository>  </repositories>  </profile>  </profiles>  <!-- activations是profile的关键，就像POM中的profiles，profile的能力在于它在特定情况下可以修改一些值。  而这些情况是通过activation来指定的。 -->  <!-- <activeProfiles/> -->  </settings> |

### Step-5 Spring Configuration

Create new Spring Configuration Bean file: /WebContent/WEB-INF/crunchify-servlet.xml



crunchify-servlet.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xmlns:context="http://www.springframework.org/schema/context"  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          http://www.springframework.org/schema/mvc          http://www.springframework.org/schema/mvc/spring-mvc.xsd          http://www.springframework.org/schema/context          http://www.springframework.org/schema/context/spring-context.xsd">    <mvc:annotation-driven />  <context:component-scan  base-package="com.crunchify.controller" />  <mvc:default-servlet-handler />    <bean id="viewResolver"  class="org.springframework.web.servlet.view.UrlBasedViewResolver">  <property name="viewClass"  value="org.springframework.web.servlet.view.JstlView" />  <property name="prefix" value="/WEB-INF/jsp/" />  <property name="suffix" value=".jsp" />  </bean>    </beans> |

In the above crunchify-servlet.xml configuration file, we have defined a tag <context:component-scan> . This will allow Spring to load all the components from package com.crunchify.controller  and all its child packages.

This will load our CrunchifyHelloWorld.class . Also we have defined a bean viewResolver. This bean will resolve the view and add prefix string /WEB-INF/jsp/  and suffix .jsp to the view in ModelAndView.

Note that in our CrunchifyHelloWorld class, we have return a ModelAndView object with view name welcome.

This will be resolved to path /WEB-INF/jsp/welcome.jsp .

### Step-6 web.xml

Create new file web.xml if it’s already not there.  Map Spring MVC in /WebContent/WEB-INF/web.xml file.

NOTE: if you don’t see web.xml file in your “dynamic web project” then follow [these steps](https://crunchify.com/eclipse-missing-web-xml-file-how-can-i-create-web-xml-in-eclipse/).

web.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd" id="WebApp\_ID" version="4.0">    <display-name>CrunchifySpringMVCTutorial</display-name>    <welcome-file-list>      <welcome-file>index.html</welcome-file>      <welcome-file>index.htm</welcome-file>      <welcome-file>index.jsp</welcome-file>      <welcome-file>default.html</welcome-file>      <welcome-file>default.htm</welcome-file>      <welcome-file>default.jsp</welcome-file>    </welcome-file-list>          <servlet>          <servlet-name>crunchify</servlet-name>          <servlet-class>              org.springframework.web.servlet.DispatcherServlet          </servlet-class>          <load-on-startup>1</load-on-startup>      </servlet>      <servlet-mapping>          <servlet-name>crunchify</servlet-name>          <url-pattern>/welcome.jsp</url-pattern>          <url-pattern>/index.jsp</url-pattern>          <url-pattern>/welcome.html</url-pattern>          <url-pattern>\*.html</url-pattern>      </servlet-mapping>    </web-app> |

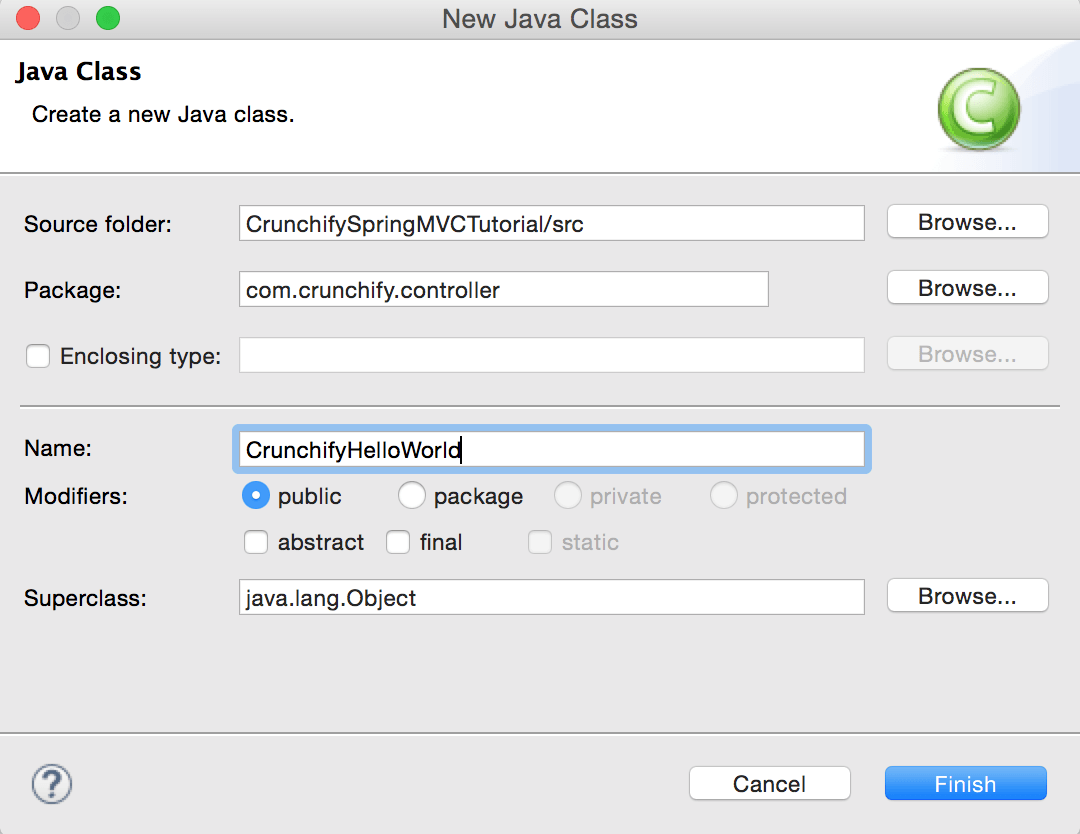
The above code in web.xml will map DispatcherServlet with url pattern /welcome.jsp. Also note that we have define index.jsp as welcome file.

One thing to note here is the name of servlet in <servlet-name> tag in web.xml. Once the DispatcherServlet is initialized, it will looks for a file name [[servlet](https://crunchify.com/how-to-do-java-servlet-session-management-using-cookies/)-name]-servlet.xml  in WEB-INF folder of web application. In this example, the framework will look for file called crunchify-servlet.xml.

### Step-7 create class

Create Controller Class.

* Right click on Java Resources -> src
* Click New -> Class
* Package: com.crunchify.controller
* Filename: CrunchifyHelloWorld.java



CrunchifyHelloWorld.java

|  |
| --- |
| package com.crunchify.controller;    import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.servlet.ModelAndView;    /\*  \* author: Crunchify.com  \*  \*/    @Controller  public class CrunchifyHelloWorld {    @RequestMapping("/welcome")  public ModelAndView helloWorld() {    String message = "<br><div style='text-align:center;'>"  + "<h3>\*\*\*\*\*\*\*\*\*\* Hello World, Spring MVC Tutorial</h3>This message is coming from CrunchifyHelloWorld.java \*\*\*\*\*\*\*\*\*\*</div><br><br>";  return new ModelAndView("welcome", "message", message);  }  } |

Note that we have annotated the CrunchifyHelloWorld class with @Controller and @RequestMapping("/welcome"). When Spring scans our package, it will recognize this bean as being a [Controller bean](https://crunchify.com/working-on-spring-mvc-project-how-to-report-list-of-all-loaded-spring-beans-during-startup/) for processing requests. The @RequestMapping annotation tells [Spring](https://crunchify.com/how-to-import-all-spring-mvc-dependencies-to-your-maven-project/) that this Controller should process all requests beginning with /welcome in the URL path. That includes /welcome/\* and /welcome.html.

The helloWorld() method returns ModelAndView object. The ModelAndView object tries to resolve to a view named “welcome” and the data model is being passed back to the browser so we can access the data within the JSP. The logical view name will resolve to /WEB-INF/jsp/welcome.jsp . Logical name “welcome” which is return in ModelAndView object is mapped to path /WEB-INF/jsp/welcome.jsp.

The ModelAndView object also contains a message with key “message” and Detailed value. This is the data that we are passing to our view. Normally this will be a value object in form of java bean that will contain the data to be displayed on our view. Here we are simply passing a string.

### Step-8 create jsp

**The View –**Create new file /WebContent/index.jsp.

index.jsp

|  |
| --- |
| <html>  <head>  <title>Spring MVC Tutorial Series by Crunchify.com</title>  <style type="text/css">  body {  background-image: url('https://cdn.crunchify.com/bg.png');  }  </style>  </head>  <body>  <br>  <div style="text-align: center">  <h2>  Hey You..!! This is your 1st Spring MCV Tutorial..<br> <br>  </h2>  <h3>  <a href="welcome.html">Click here to See Welcome Message... </a>(to  check Spring MVC Controller... @RequestMapping("/welcome"))  </h3>  </div>  </body>  </html> |

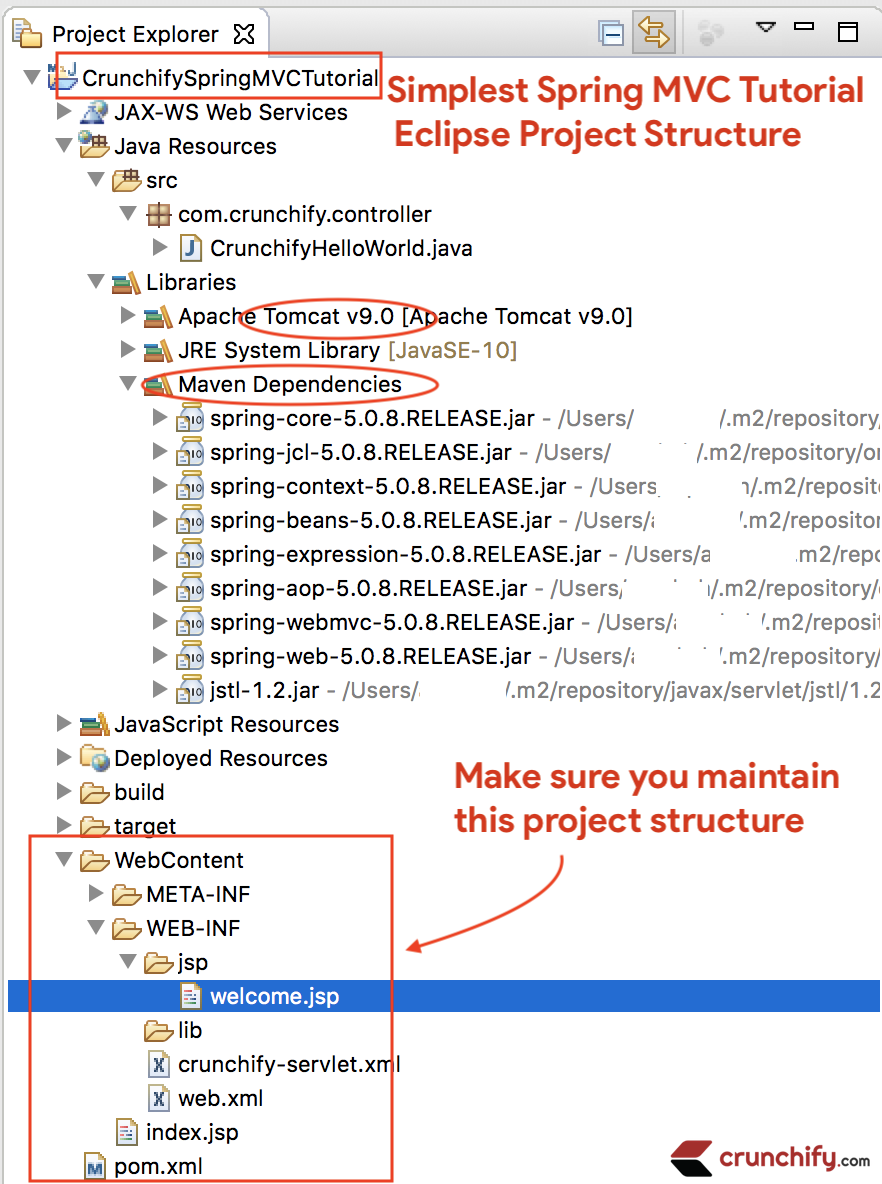
Create another file /WebContent/WEB-INF/jsp/welcome.jsp.

NOTE: Don’t forget to create jsp folder and put welcome.jsp inside that 🙂

welcome.jsp

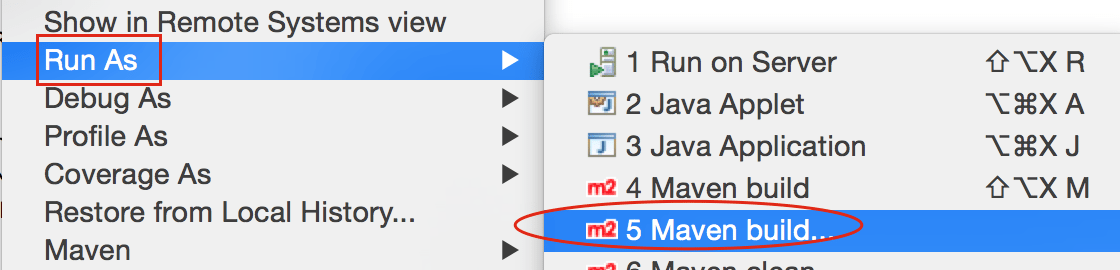
|  |
| --- |
| <html>  <head>  <title>Spring MVC Tutorial by Crunchify - Hello World Spring MVC  Example</title>  <style type="text/css">  body {  background-image: url('https://cdn.crunchify.com/bg.png');  }  </style>  </head>  <body>${message}    <br>  <br>  <div  style="font-family: verdana; padding: 10px; border-radius: 10px; font-size: 12px; text-align: center;">    Spring MCV Tutorial by <a href="https://crunchify.com">Crunchify</a>.  Click <a href="https://crunchify.com/category/java-tutorials/"  target="\_blank">here</a> for all Java and <a  href='https://crunchify.com/category/spring-mvc/' target='\_blank'>here</a>  for all Spring MVC, Web Development examples.<br>  </div>  </body>  </html> |

After everything this is how your workspace should look like.

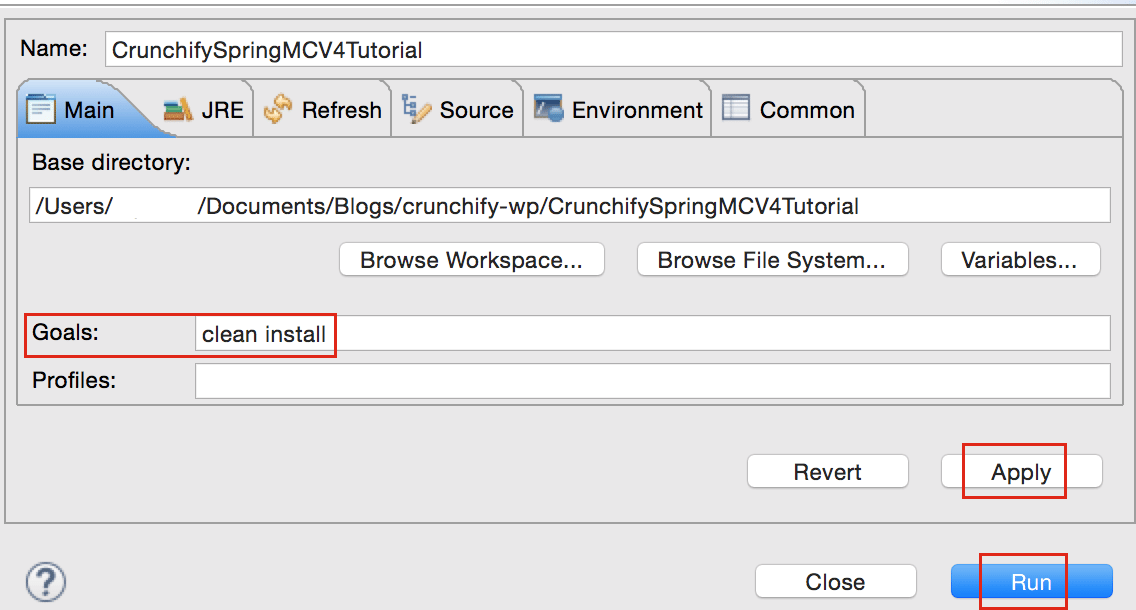


### Step-9 run maven

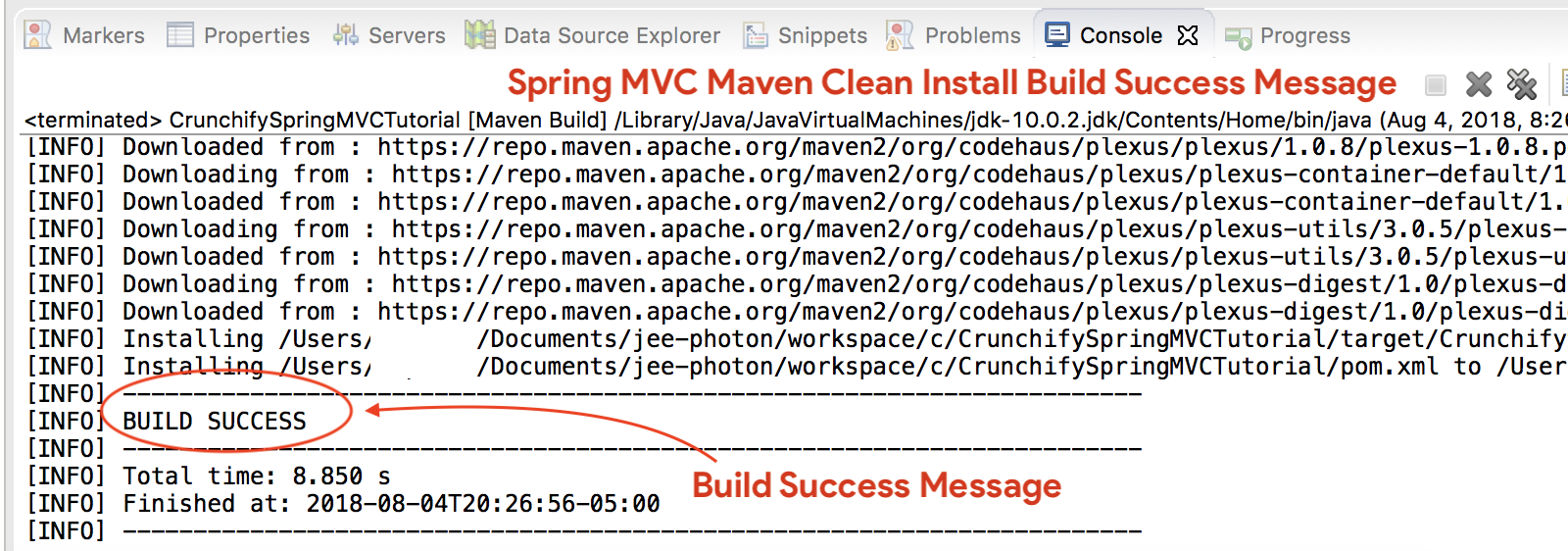
Right Click on Project -> Run As -> Maven Build...



Add Goals: clean install. Click Apply and Run.



You should see build success message:

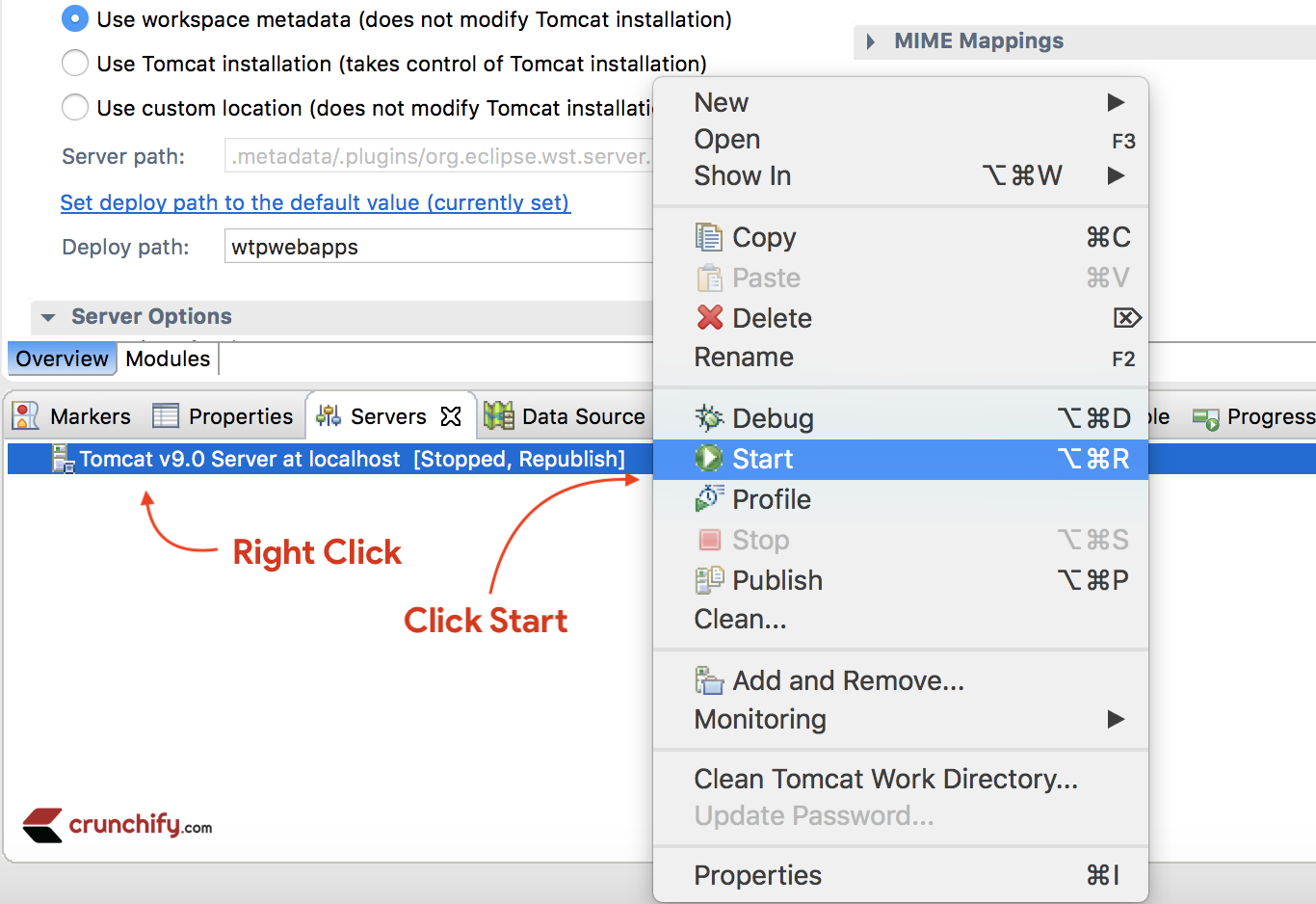


#### Where are all of my .jar files?

You will see all .jar files under /target folder. [Screenshot](https://cdn.crunchify.com/wp-content/uploads/2013/02/Spring-MVC-Hello-World-Target-Folder-Structure.png).

### Step-10 run server

* If you don't see Tomcat Server in Servers tab then follow steps to [add Apache Tomcat to Eclipse](https://crunchify.com/step-by-step-guide-to-setup-and-install-apache-tomcat-server-in-eclipse-development-environment-ide/).
* Deploy project to Apache Tomcat
  + Right click
  + Add and Remove
  + Add Project to server (right side section).
* Click on Start.



Make sure you see below logs. That means your application is successfully deployed on Tomcat Web Server.

|  |
| --- |
| Aug 04, 2018 9:08:10 PM org.apache.tomcat.util.digester.SetPropertiesRule begin  WARNING: [SetPropertiesRule]{Server/Service/Engine/Host/Context} Setting property 'source' to 'org.eclipse.jst.jee.server:CrunchifySpringMVCTutorial' did not find a matching property.  INFO: Server version:        Apache Tomcat/9.0.10  INFO: Server built:          Jun 20 2018 17:32:21 UTC  INFO: Server number:         9.0.10.0  INFO: OS Name:               Mac OS X  INFO: OS Version:            10.13.6  INFO: Architecture:          x86\_64  INFO: Java Home:             /Library/Java/JavaVirtualMachines/jdk-10.0.2.jdk/Contents/Home  INFO: JVM Version:           10.0.2+13  INFO: JVM Vendor:            "Oracle Corporation"  INFO: CATALINA\_BASE:         /Users/appshah/Documents/jee-photon/workspace/c/.metadata/.plugins/org.eclipse.wst.server.core/tmp0  INFO: CATALINA\_HOME:         /Users/appshah/Documents/jee-photon/apache-tomcat-9.0.10  INFO: Command line argument: -Dcatalina.base=/Users/appshah/Documents/jee-photon/workspace/c/.metadata/.plugins/org.eclipse.wst.server.core/tmp0  INFO: Command line argument: -Dcatalina.home=/Users/appshah/Documents/jee-photon/apache-tomcat-9.0.10  INFO: Command line argument: -Dwtp.deploy=/Users/appshah/Documents/jee-photon/workspace/c/.metadata/.plugins/org.eclipse.wst.server.core/tmp0/wtpwebapps  INFO: Command line argument: -Dfile.encoding=UTF-8  INFO: The APR based Apache Tomcat Native library which allows optimal performance in production environments was not found on the java.library.path: [/Users/appshah/Library/Java/Extensions:/Library/Java/Extensions:/Network/Library/Java/Extensions:/System/Library/Java/Extensions:/usr/lib/java:.]  INFO: Initializing ProtocolHandler ["http-nio-8080"]  INFO: Using a shared selector for servlet write/read  INFO: Initializing ProtocolHandler ["ajp-nio-8009"]  INFO: Using a shared selector for servlet write/read  INFO: Initialization processed in 841 ms  INFO: Starting service [Catalina]  INFO: Starting Servlet Engine: Apache Tomcat/9.0.10  INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs during scanning can improve startup time and JSP compilation time.  INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs during scanning can improve startup time and JSP compilation time.  INFO: No Spring WebApplicationInitializer types detected on classpath  INFO: Initializing Spring FrameworkServlet 'crunchify'  INFO: FrameworkServlet 'crunchify': initialization started  INFO: Refreshing WebApplicationContext for namespace 'crunchify-servlet': startup date [Sat Aug 04 21:08:13 CDT 2018]; root of context hierarchy  INFO: Loading XML bean definitions from ServletContext resource [/WEB-INF/crunchify-servlet.xml]  INFO: Mapped "{[/welcome]}" onto public org.springframework.web.servlet.ModelAndView com.crunchify.controller.CrunchifyHelloWorld.helloWorld()  INFO: Looking for @ControllerAdvice: WebApplicationContext for namespace 'crunchify-servlet': startup date [Sat Aug 04 21:08:13 CDT 2018]; root of context hierarchy  INFO: Looking for @ControllerAdvice: WebApplicationContext for namespace 'crunchify-servlet': startup date [Sat Aug 04 21:08:13 CDT 2018]; root of context hierarchy  INFO: Mapped URL path [/\*\*] onto handler 'org.springframework.web.servlet.resource.DefaultServletHttpRequestHandler#0'  INFO: FrameworkServlet 'crunchify': initialization completed in 1607 ms  INFO: Starting ProtocolHandler ["http-nio-8080"]  INFO: Starting ProtocolHandler ["ajp-nio-8009"]  INFO: Server startup in 3579 ms |

### Step-11 show index.jsp

Visit: <http://localhost:8080/CrunchifySpringMVCTutorial/> and you should be all set.

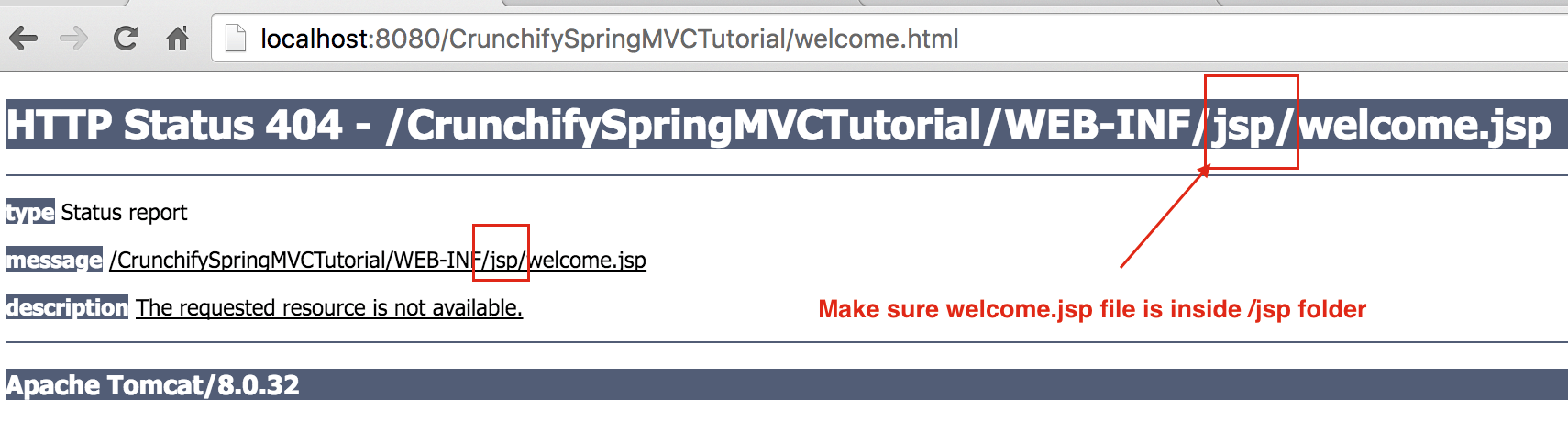
Hurray.. Now you know Hello World Spring MVC 5 Example. Let me know if you encounter any exception while running this. There are lot more example you can find [here](https://crunchify.com/category/java-tutorials/).

Do you want to include JS, CSS and images into JSP file? Follow this tutorial: [Best way to Add/Integrate JS, CSS and images into JSP file using ‘mvc:resources mapping’](https://crunchify.com/spring-mvc-4-2-2-best-way-to-integrate-js-and-css-file-in-jsp-file-using-mvcresources-mapping/).

Having trouble? Any issue?

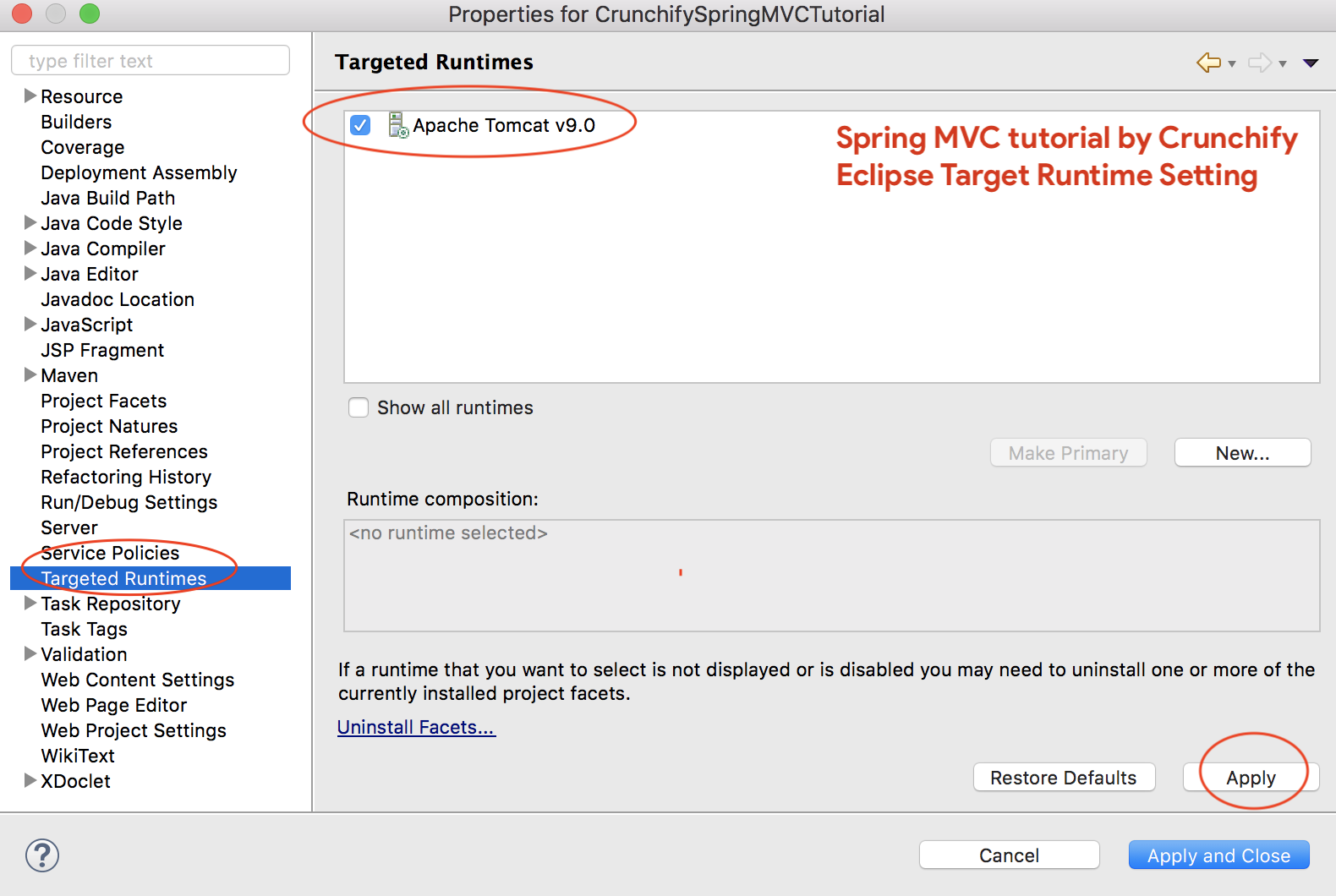
## 3.Triaging errors

### Triaging Step-1 – Having HTTP Status 404 error?

  
Also, follow below tutorial:

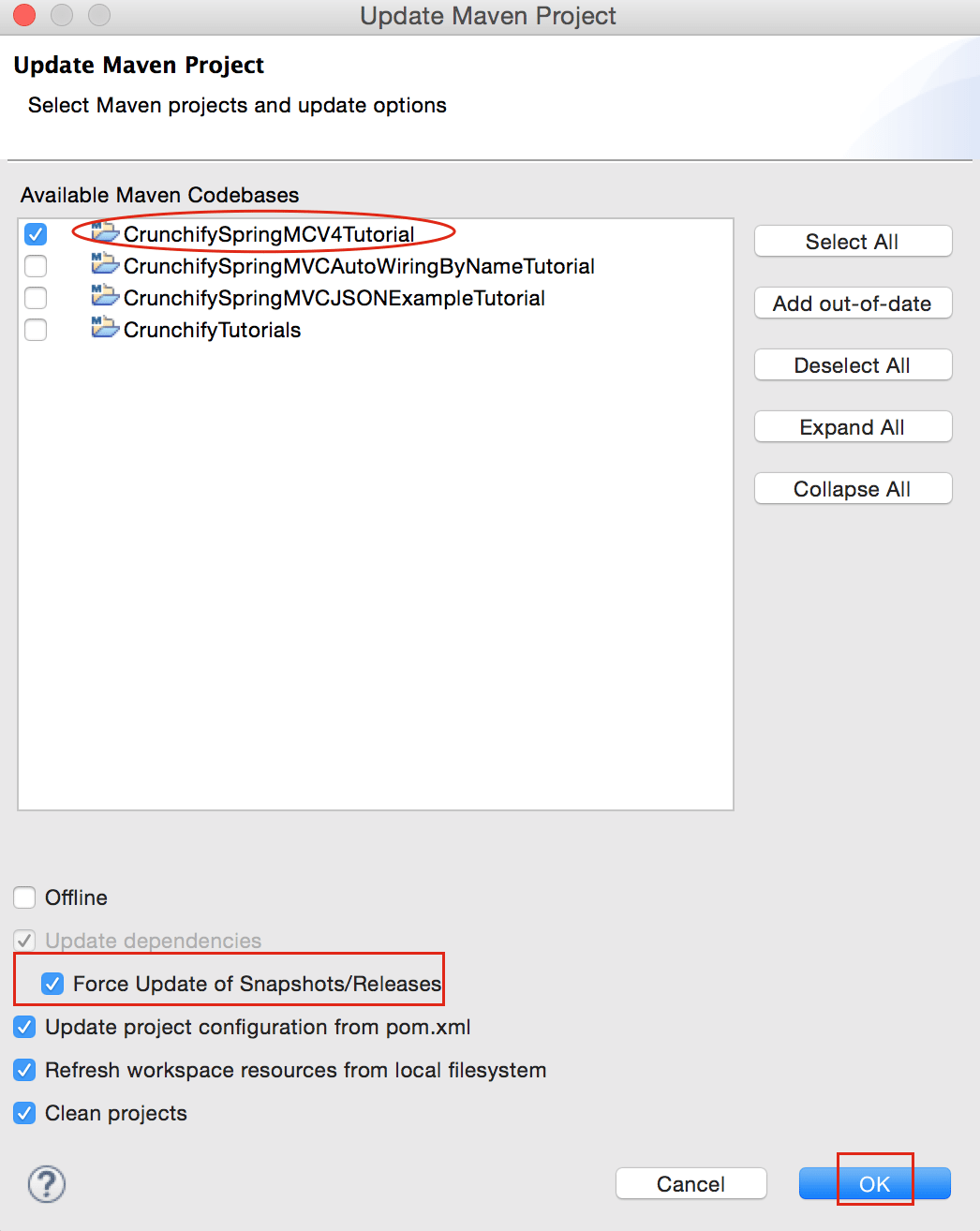
### Triaging step-2 – URL doesn’t work? Tomcat error?

Make sure you add Apache Tomcat Server to Targeted Runtime. Which you may have selected in Step-1. Tomcat 7 or 8 any – server should work.



### Triaging Step-3 – maven errors?

Make sure to update all maven dependencies.



Feel free to email or comment below if you have any problem running above tutorial.