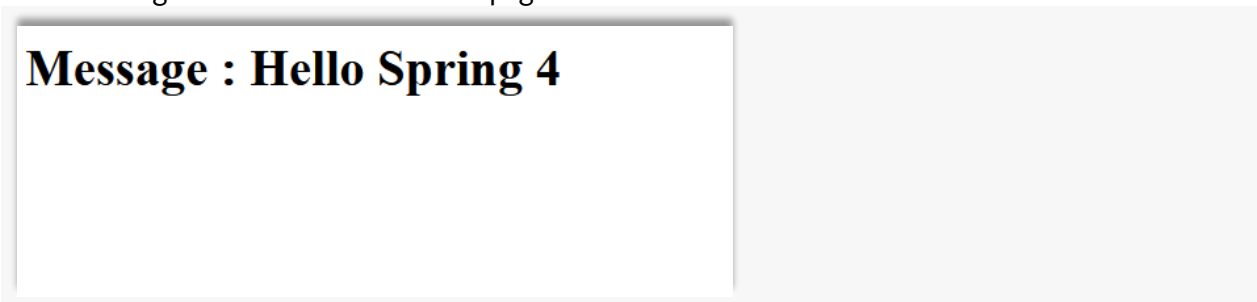
Article summary

Summary	Discusses how to create a Spring MVC 4.x project by using Maven and IntelliJ.
Audience	Developer (intermediate)
Required Skills	Java, Maven

Introduction

You can create a Spring MVC project by using Maven and IntelliJ IDE. In this development article, Spring version 4.3 and IntelliJ 2019 are used. This development article guides you through how to create a Spring MVC project, including how to configure the Spring DispatcherServlet. This servlet dispatches client requests to handlers. The default handler is a controller interface that lets you work with a `ModelMap` instance

The following illustration shows the web page that is created in this article.

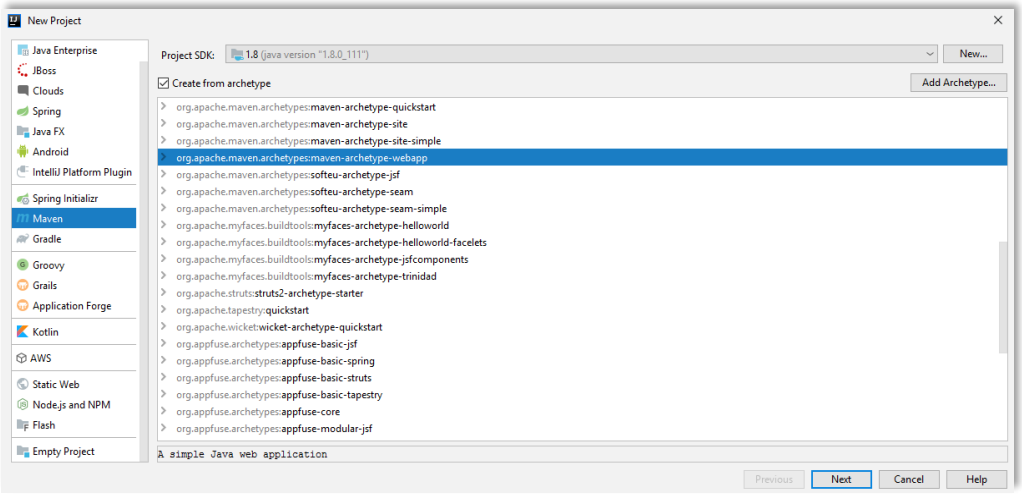


An Spring MVC application

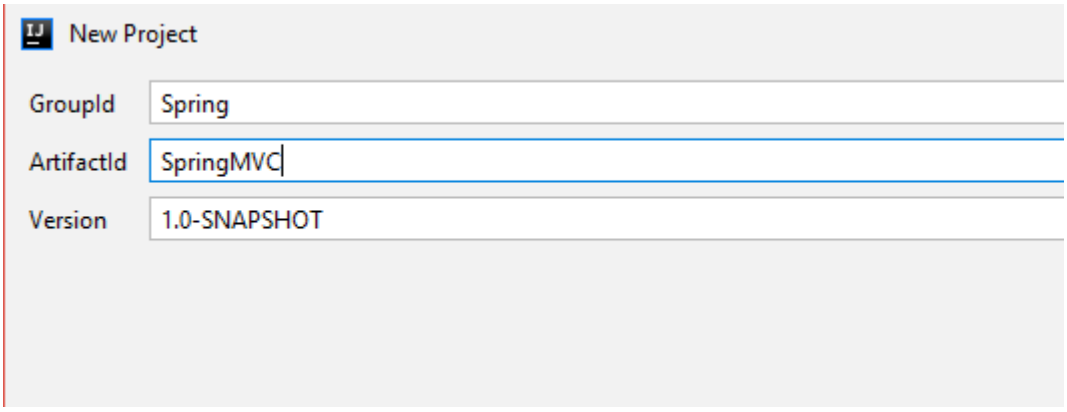
Create the IntelliJ project

Create a new project by using IntelliJ:

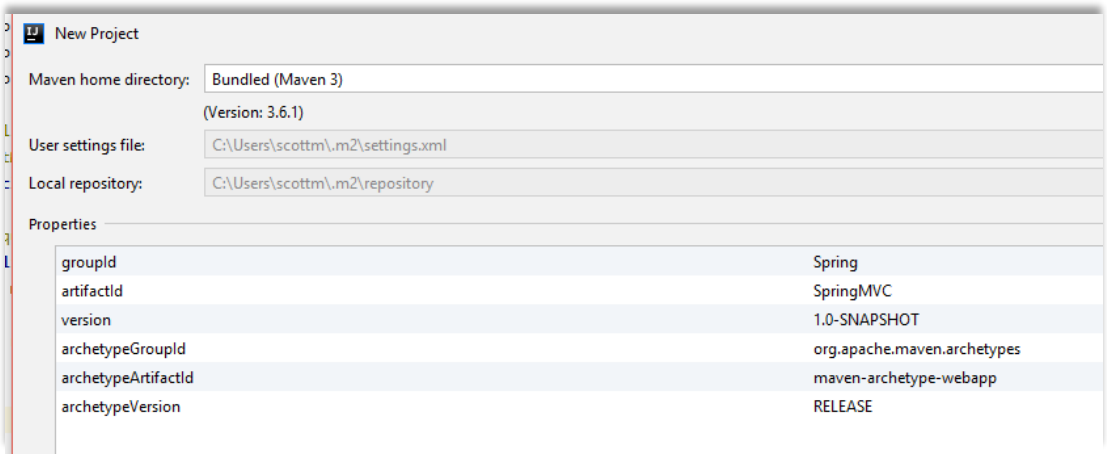
1. Open IntelliJ IDEA and click **File**, **New, Project**.
2. Select **Maven** and click **Create from archetype**.
3. Select **org.apache.maven.archetypes:maven-archetype-webapp**.



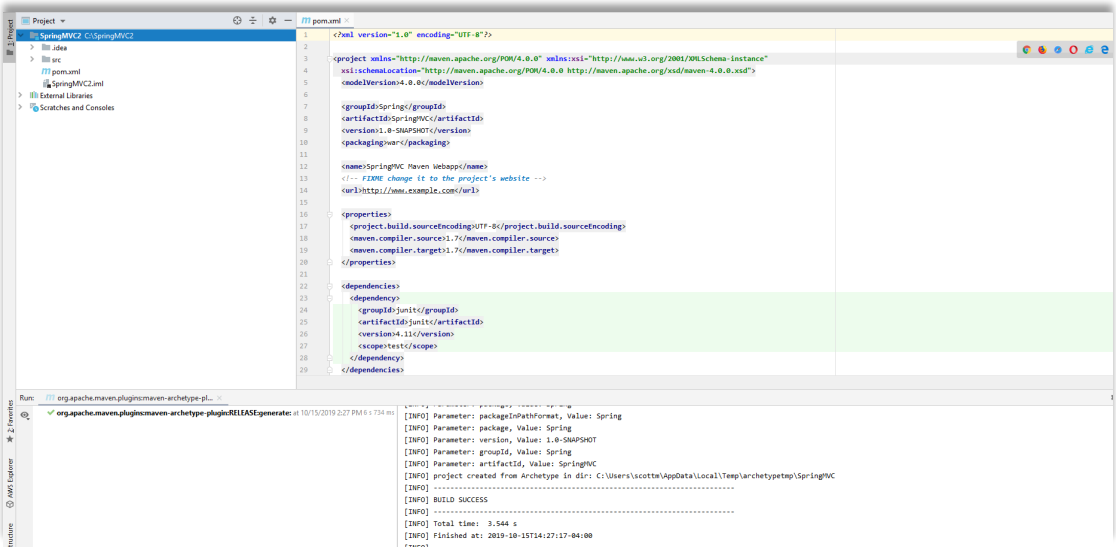
4.



5. Click **Next** and ensure you see the following values.



6. Click **Next** and **Finish.** IntelliJ has created a new project.



Modify the Project POM file

To successfully add Spring library files to your project, you must add the required dependency to your project's POM file. Add the following code to your POM file.

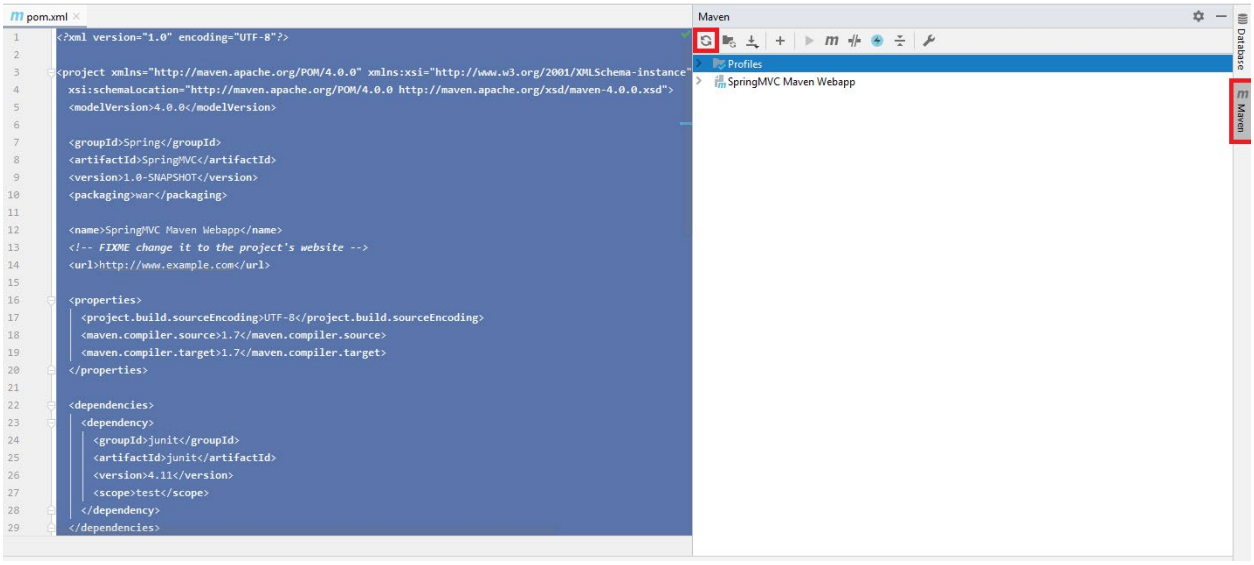
```
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
<version>4.3.3.RELEASE</version>
</dependency>
```

Once you are done, your POM file resembles this file.

```
<?xml version="1.0" encoding="UTF-8"?>

<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>Spring</groupId>
  <artifactId>SpringMVC</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>SpringMVC Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <finalName>SpringMVC</finalName>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults
(may be moved to parent pom) -->
      <plugins>
        <plugin>
          <artifactId>maven-clean-plugin</artifactId>
          <version>3.1.0</version>
```

```
        </plugin>
        <!-- see http://maven.apache.org/ref/current/maven-core/default-
bindings.html#Plugin_bindings_for_war_packaging -->
        <plugin>
          <artifactId>maven-resources-plugin</artifactId>
          <version>3.0.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.0</version>
        </plugin>
        <plugin>
          <artifactId>maven-surefire-plugin</artifactId>
          <version>2.22.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-war-plugin</artifactId>
          <version>3.2.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-install-plugin</artifactId>
          <version>2.5.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>2.8.2</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

Click **Maven** at the right bar of the IDE. Then, click the **sync** button to refresh the dependency packages.

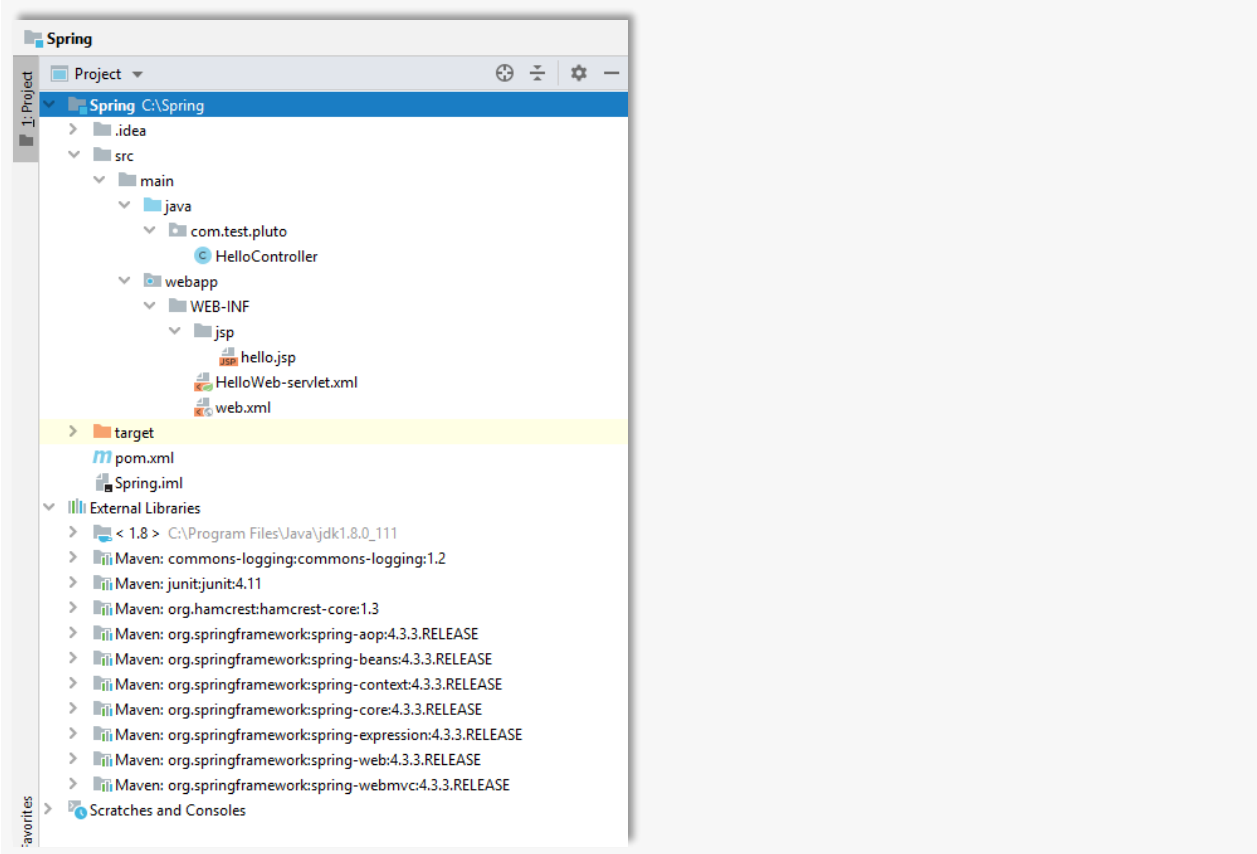


Add project files to your project

Add these project files to your project:

- A Java file that represents the Spring MVC controller
- A JSP file
- XML configuration files

The following illustration shows these files.



The Spring MVC project files

NOTE: Delete the existing *web.xml* and *index.jsp* files.

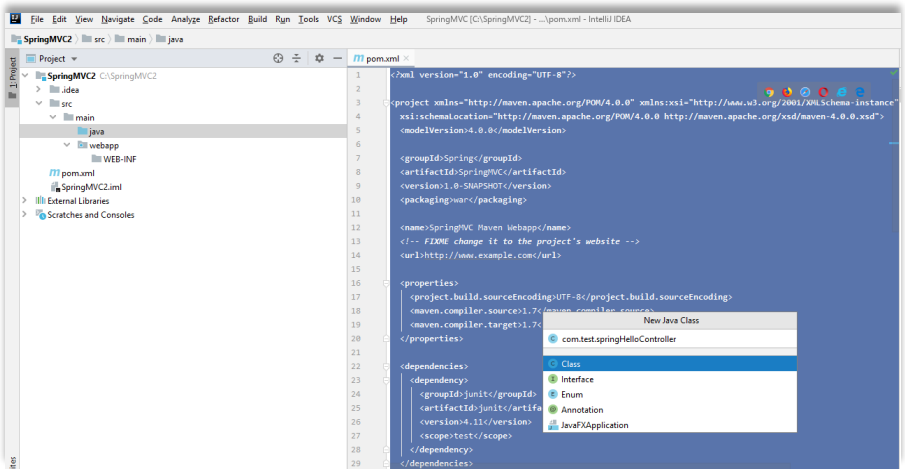
Add the Java Controller class

Perform these steps:

1. In the Project explorer, choose **main**, select **New**, **Directory**.
2. Enter **java**.

3. Select the **java** directory, choose **Make Directory as**, then, choose **Source Root**.

4. Select the **java** directory, choose **New**, **Java Class**. Name the class: `com.test.spring.HelloController`.



5. Add the following code to the `HelloController` class.

```
package com.test.spring;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping("")
public class HelloController {

    @RequestMapping(method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spring 4");

        return "hello";
    }
}
```

Add the JSP file

The next step is to add the JSP file named *hello.jsp* to your project. In this example, place the JSP file in a folder named **jsp** (this is a sub-folder to WEB-INF). All this JSP does display the value of `${messages}` which was populated in the Java controller. Perform these steps:

1. Under the **WEB-INF** folder, create a new folder name **jsp**.

2. Create the JSP file named *hello.jsp* in the **jsp** folder.

3. Add the following code to this JSP file.

```
<html>
<body>
<h1>Message : ${message}</h1>
</body>
</html>
```

Create the web.xml file

The *web.xml* file defines servlets for the web application (after all, a Spring MVC project is a web application). Within the web.xml file, you specify a Spring MVC servlet. A Spring MVC servlet is based on `org.springframework.web.servlet.DispatcherServlet`. In the following example, the name of the Spring dispatcher servlet is **HelloWeb**.

Perform these steps:

1. Under the **WEB-INF** folder, create a new file named *web.xml*.

2. Add the following code to this file.

```
<web-app id="WebApp_ID" version="2.4"
         xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
 http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <display-name>Spring Web MVC Application</display-name>
    <servlet>
        <servlet-name>HelloWeb</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloWeb</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

Create the HelloWeb-servlet.xml file

The *HelloWeb-servlet.xml* informs the Spring framework where to find the controller (in this article, the controller is the **HelloController** class). The dispatcher configuration uses the following element to specify the location of the controller:

```
<context:component-scan base-package="com.test.spring" />
```

Perform these steps:

1. Under the **WEB-INF** folder, create a new file named *HelloWeb-servlet.xml*.

2. Add the following code to this file.

```
<beans xmlns="http://www.springframework.org/schema/beans"
       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-3.0.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.0.xsd">

    <context:component-scan base-package="com.test.spring" />

    <bean
            class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix">
            <value>/WEB-INF/jsp/</value>
        </property>
        <property name="suffix">
            <value>.jsp</value>
        </property>
    </bean>
</beans>
```

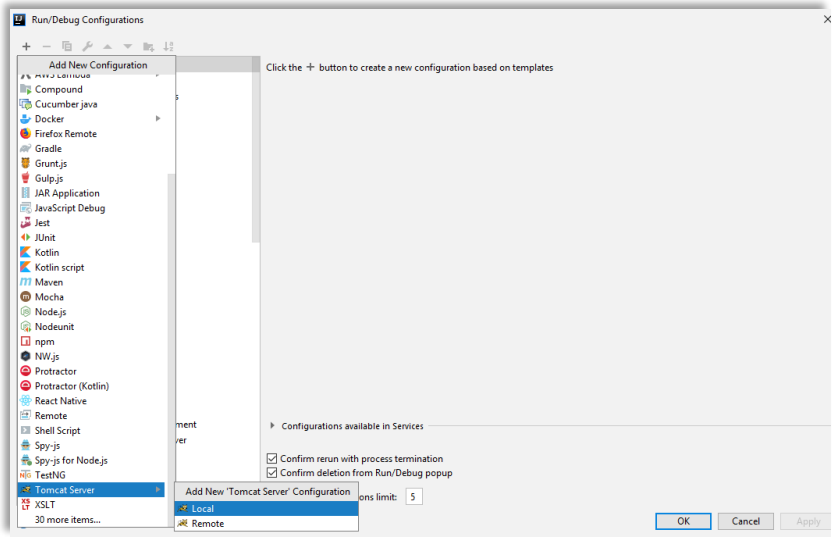
Run the Spring MVC application

To run the application, ensure that you have installed TOMCAT in your environment. Perform these steps:

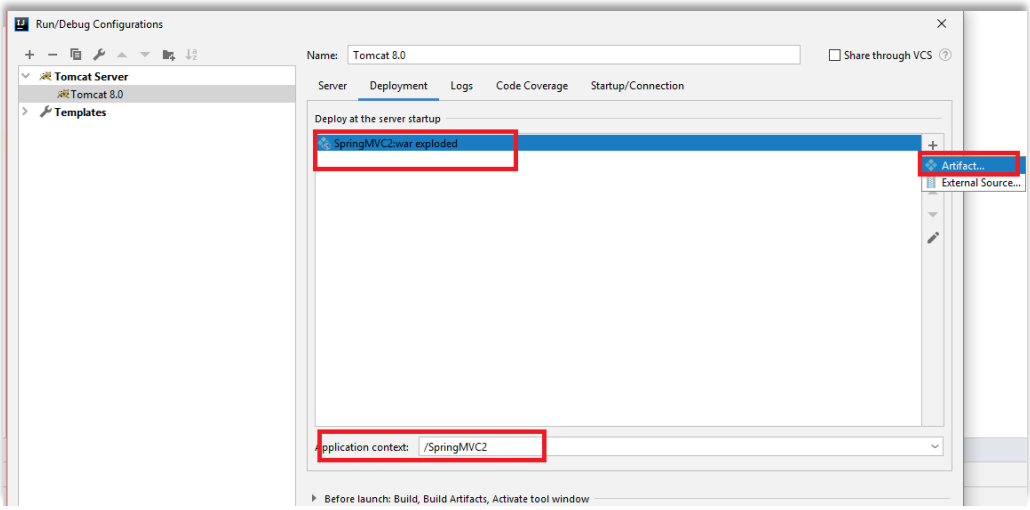
1. Click **Run**, then choose **run**.

2. Click **Edit Configurations**.

3. Click +, **Tomcat Server**, **Local**.



4. Choose the **Deployment** tab, choose "+", select **Artifact**…



5. Choose the second one **springMVC:war exploded**.

6. Click "OK", then click blue button "Run". Now you should see your application running in a web browser. (See the first illustration shown at the start of this article).