"Spring MVC – Básico parte I"

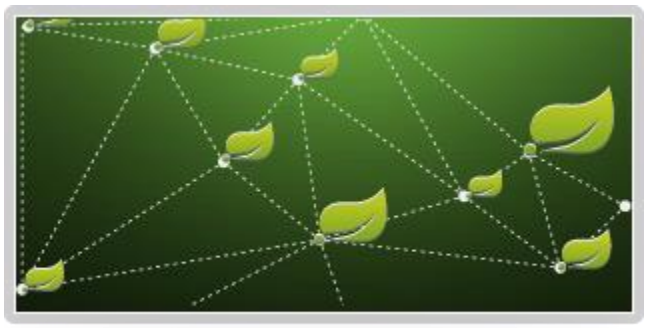
Módulo 3 / 1

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

Introducción

Spring MVC es uno de los componentes del Framework de Spring, y como su propio nombre nos indica implementa una arquitectura Modelo - Vista - Controlador que utilizaremos como base para desarrollar nuestras aplicaciones web.

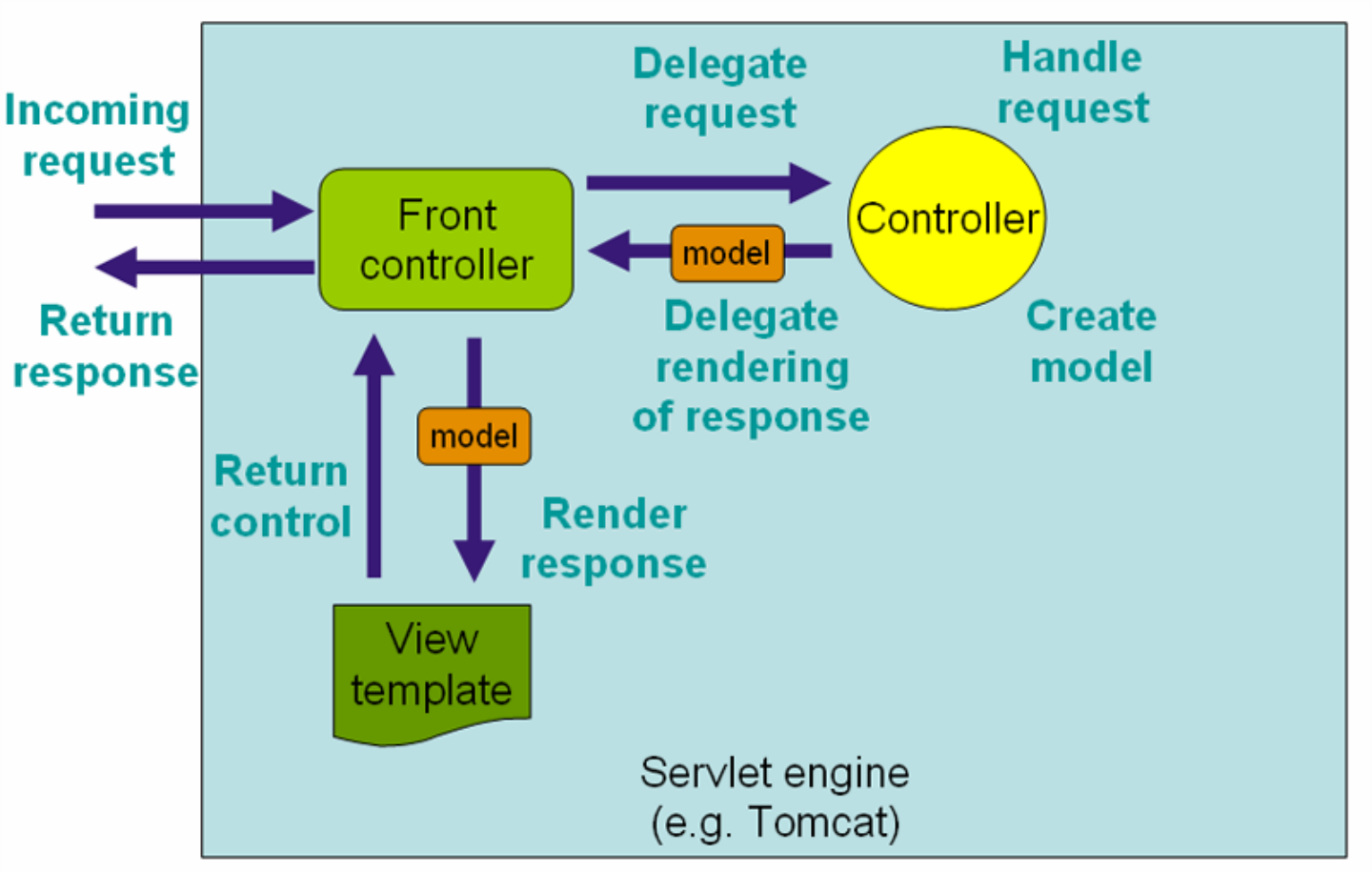
Spring MVC es considerado uno de los más poderosos y a la vez flexible Framework Modelo-Vista-Controlador (MVC) para aplicaciones basadas en Web. Se encarga de mapear las peticiones (o request) hacia los controladores, y de los controladores cargar las vistas. Cuenta con excelentes capacidades de manejo de formularios y validaciones de datos, se integra con las tecnologías o motores de vistas más populares, tales como JSP, Thymeleaf, Velocity, FreeMarker, JasperReports, Excel y PDF.



AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

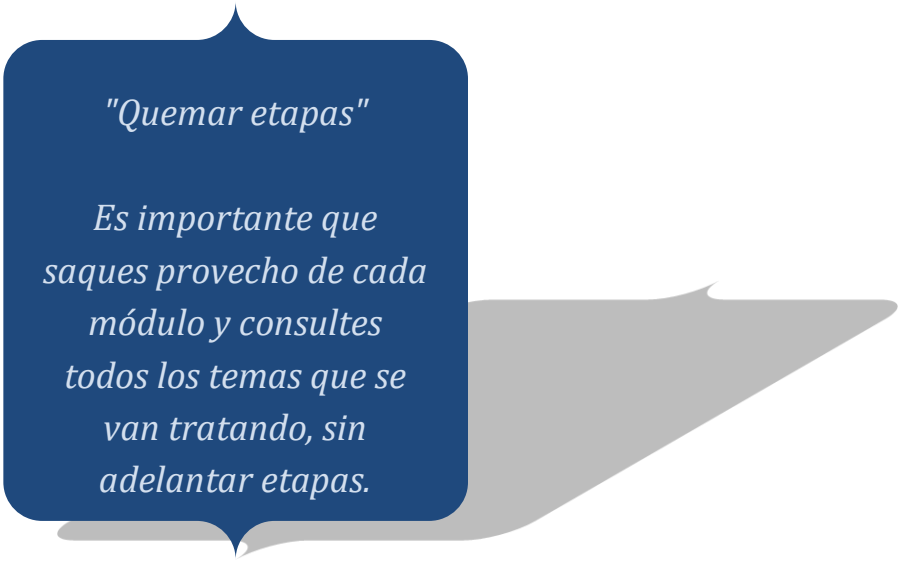
Ciclo de vida de una petición HTTP en Spring Web MVC

1. DispatchServlet recibe una petición HTTP
2. DispatchServlet selecciona un Controlador (Controller) basado en la configuración de mapeos de URL
3. Controller ejecuta un algoritmo de lógica de negocio business logic (y asigna los datos al objeto de vista)
4. Controller retorna un objeto de vista ModelAndView
5. ViewResolver selecciona una vista
6. Una vista seleccionada puedo obtener e imprimir los valores asignados en el controlador.



Objetivos

En este laboratorio, vamos a explorar las características básicas de Spring MVC, especialmente en el uso de anotaciones en los controladores.

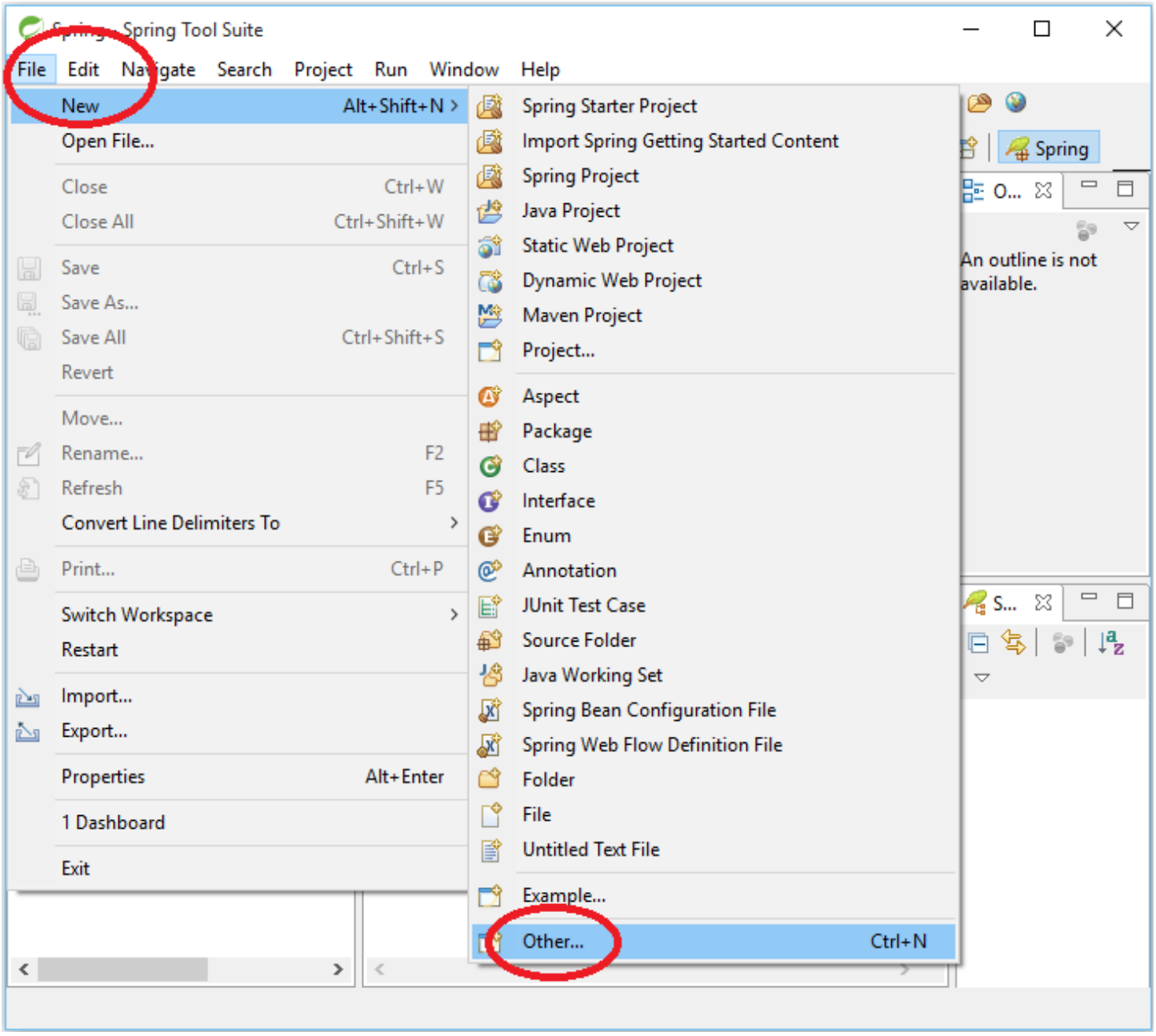


AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

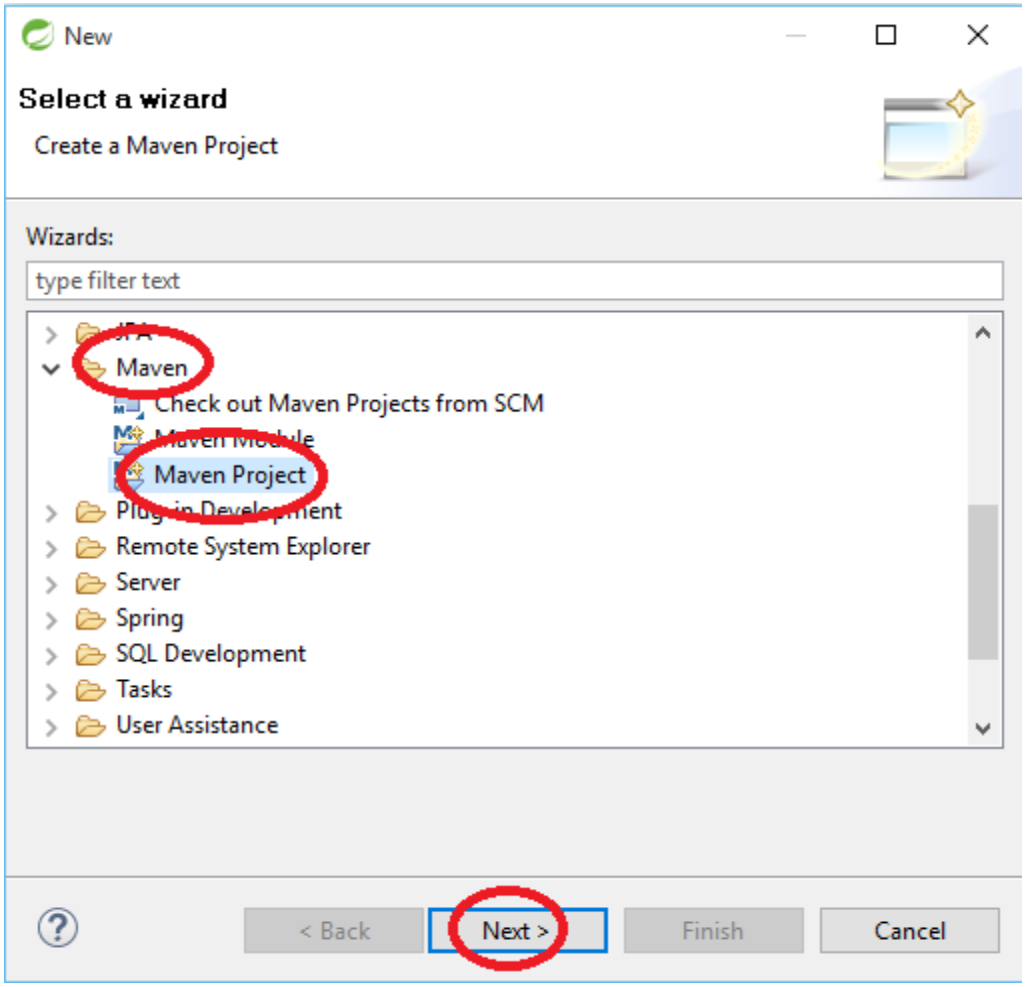
Ejercicio 1: Construir y ejecutar una aplicación "holamundo" usando maven

En este ejercicio, vamos a construir desde cero paso a paso una aplicación Hola Mundo usando Maven arquetipo webapp.

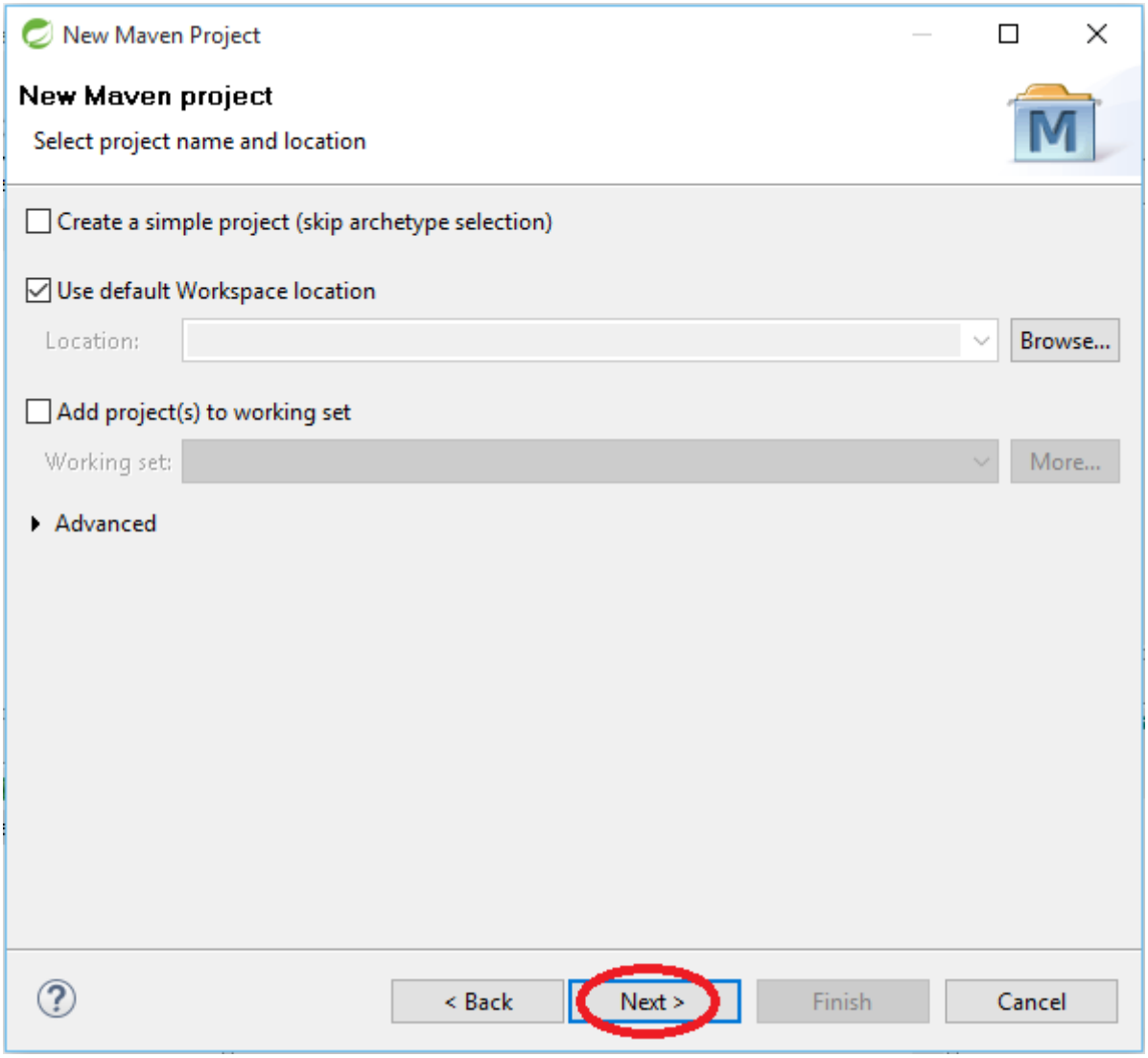
1. Creamos un proyecto maven del tipo "**maven-archetype-webapp**"
2. Creamos un nuevo proyecto Maven: **Seleccionar File->New->Other**.



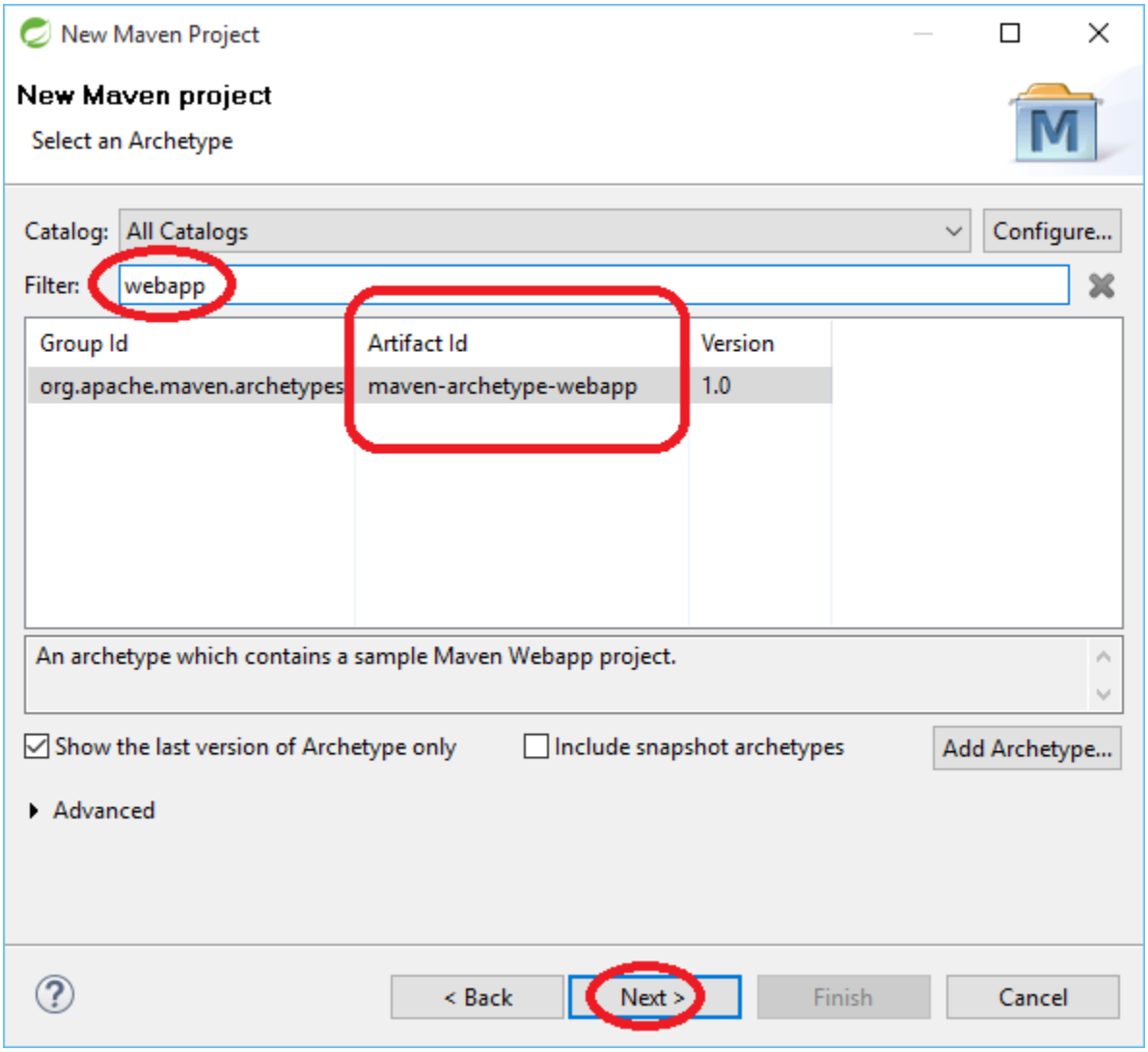
- Expandir **Maven**.
- Seleccionar **Maven Project**
- Clic **Next**.



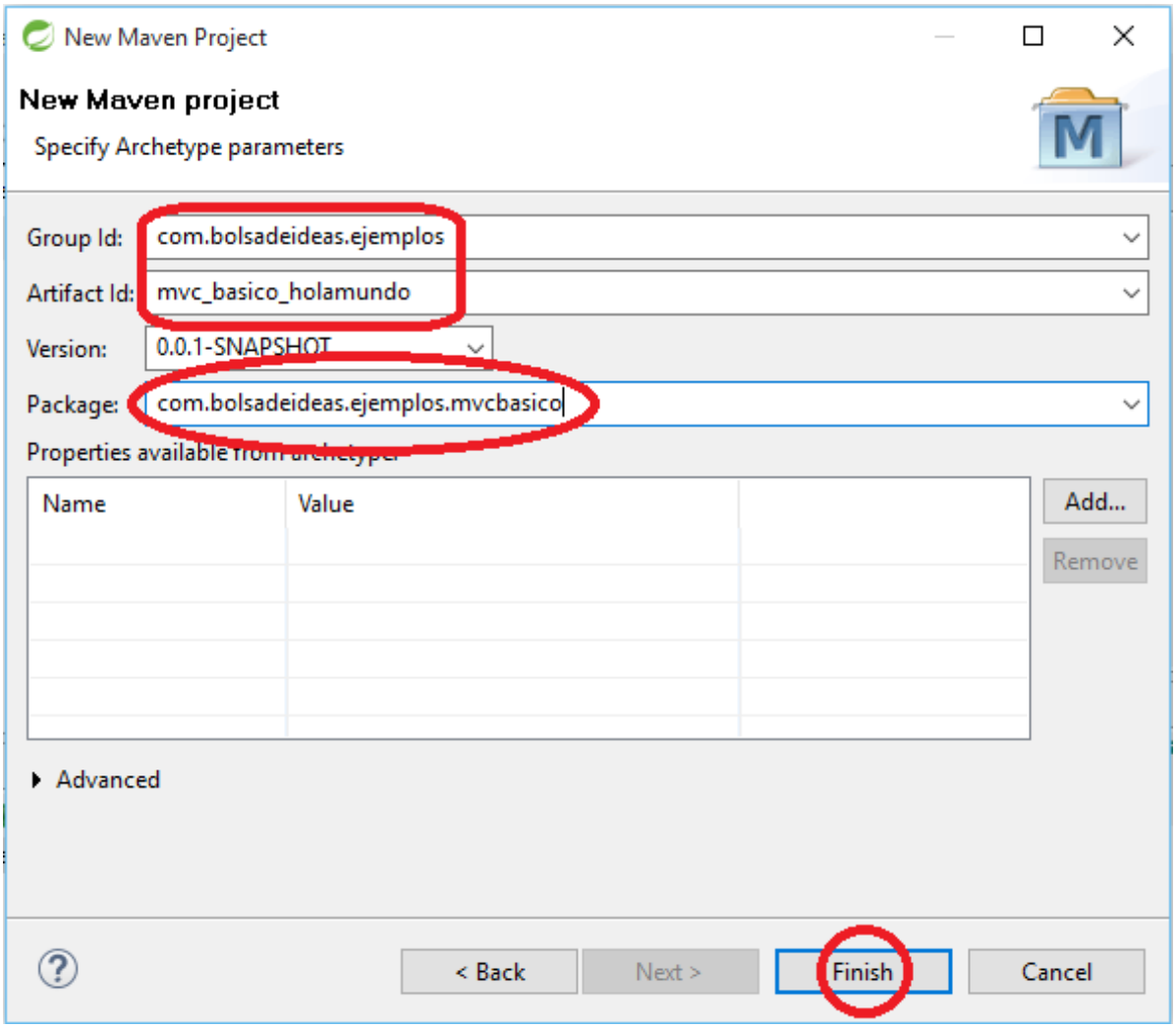
- Clic **Next**


New Maven Project
New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
Browse...
Add project(s) to working set
Working set:
More...
Advanced
< Back
Next >
Finish
Cancel

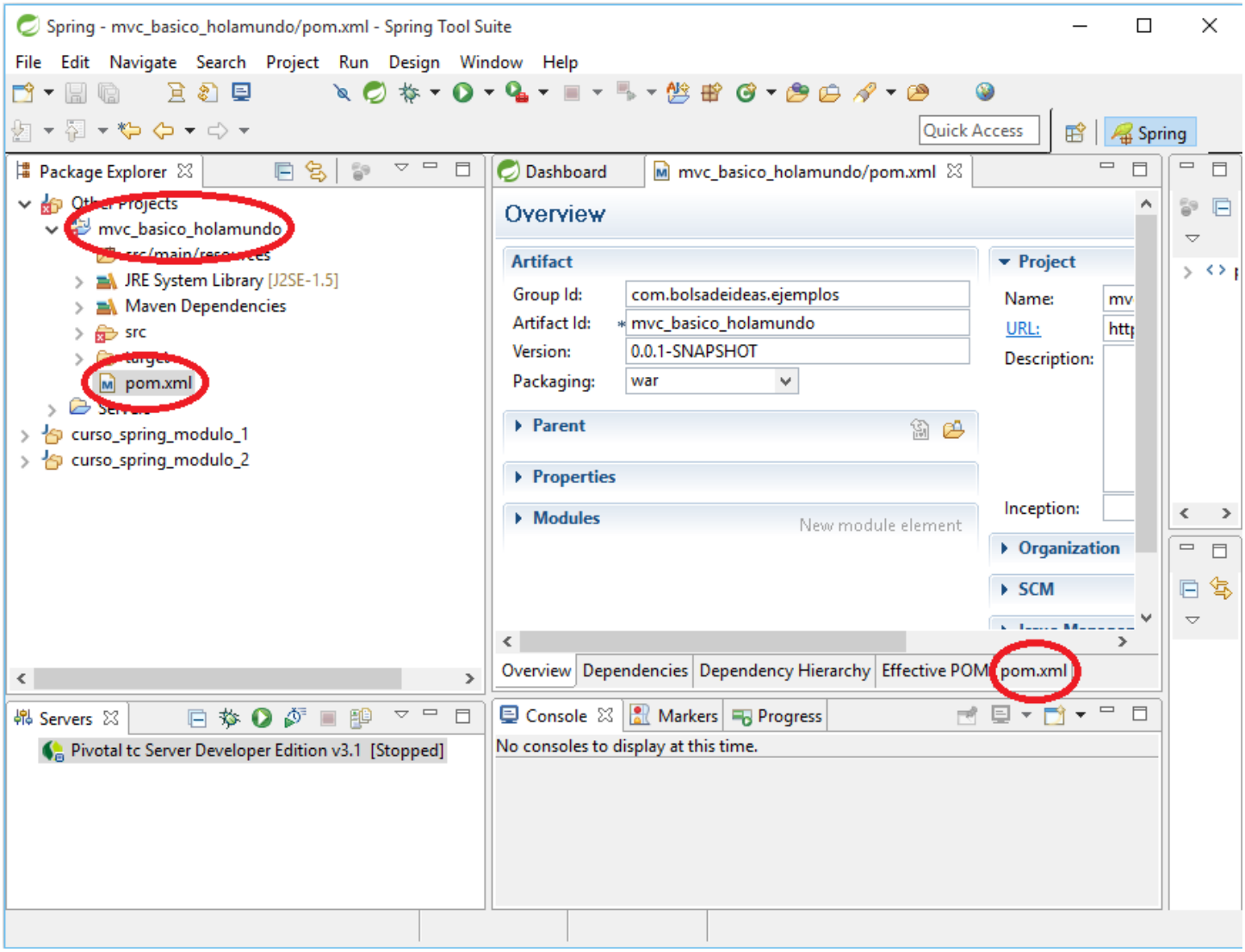
3. Seleccionamos el arquetipo **maven-archetype-webapp**.
    - Observamos que aparece la ventana de dialogo **New Maven Project**.
    - En campo Filter, ingresamos **webapp** para filtrar.
    - Seleccionamos **maven-archetype-webapp**.
    - Clic **Next**



4. Asignamos los parámetros y datos del proyecto maven
    - Para el campo Group ID, introduzca **com.bolsadeideas.ejemplos**.
    - Para el campo Artifact Id, introduzca **mvc_basico_holamundo** (será el nombre del proyecto).
    - Observe que el campo Package es automáticamente completado por el IDE, opcionalmente lo podemos modificar/ajustar.
    - Haga clic en Finish.



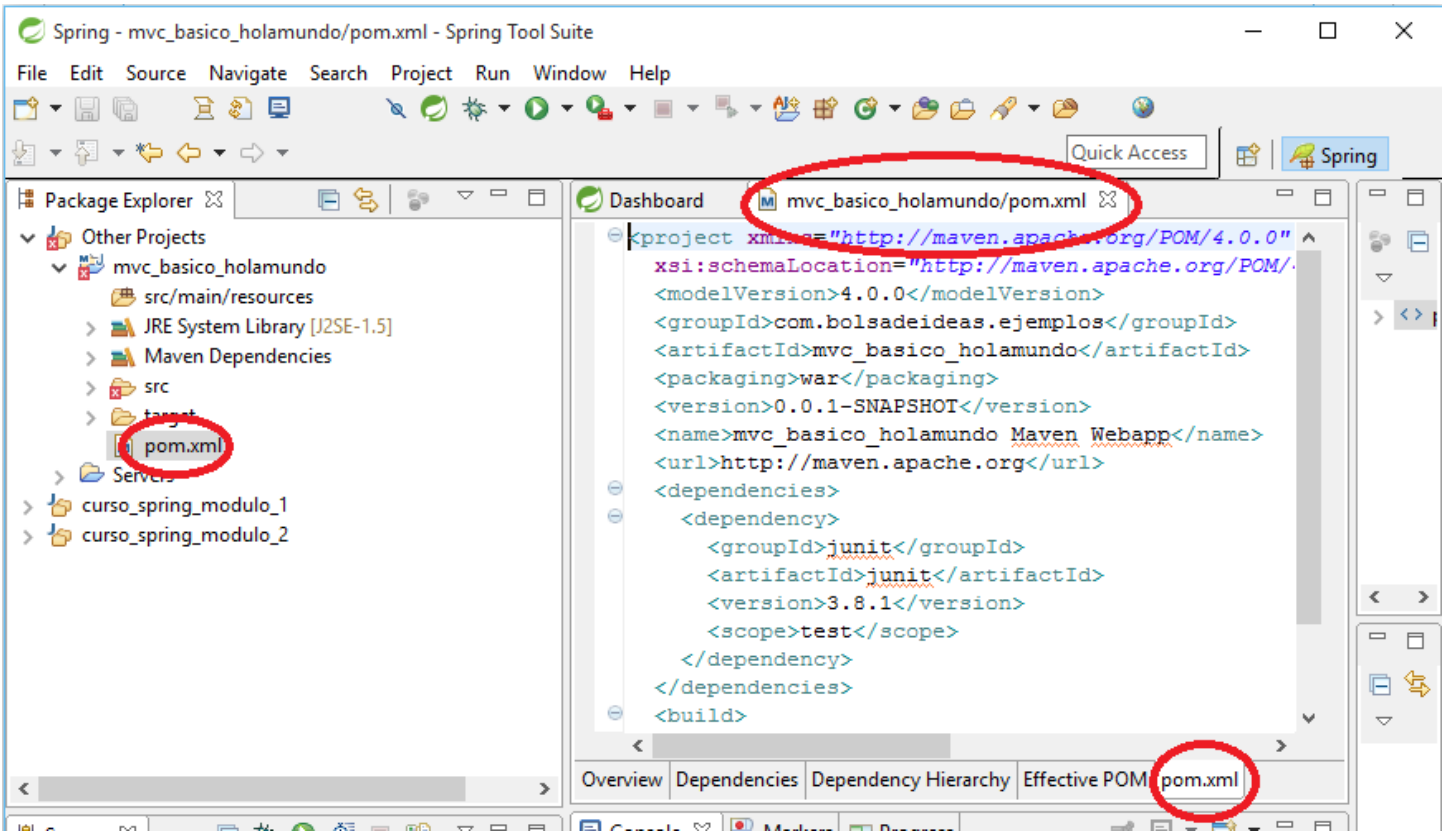
5. Automáticamente se genera el proyecto Maven con su respectivo archivo pom.xml



6. Agregamos las **dependencias actualizadas** de Spring Framework en el pom.xml
    - Modificamos el pom.xml como se muestra a continuación. Esto es para agregar las dependencias de las librerías de Spring MVC. Los fragmentos de código que deben ser agregados se resaltan en **negrita y de color rojo**.

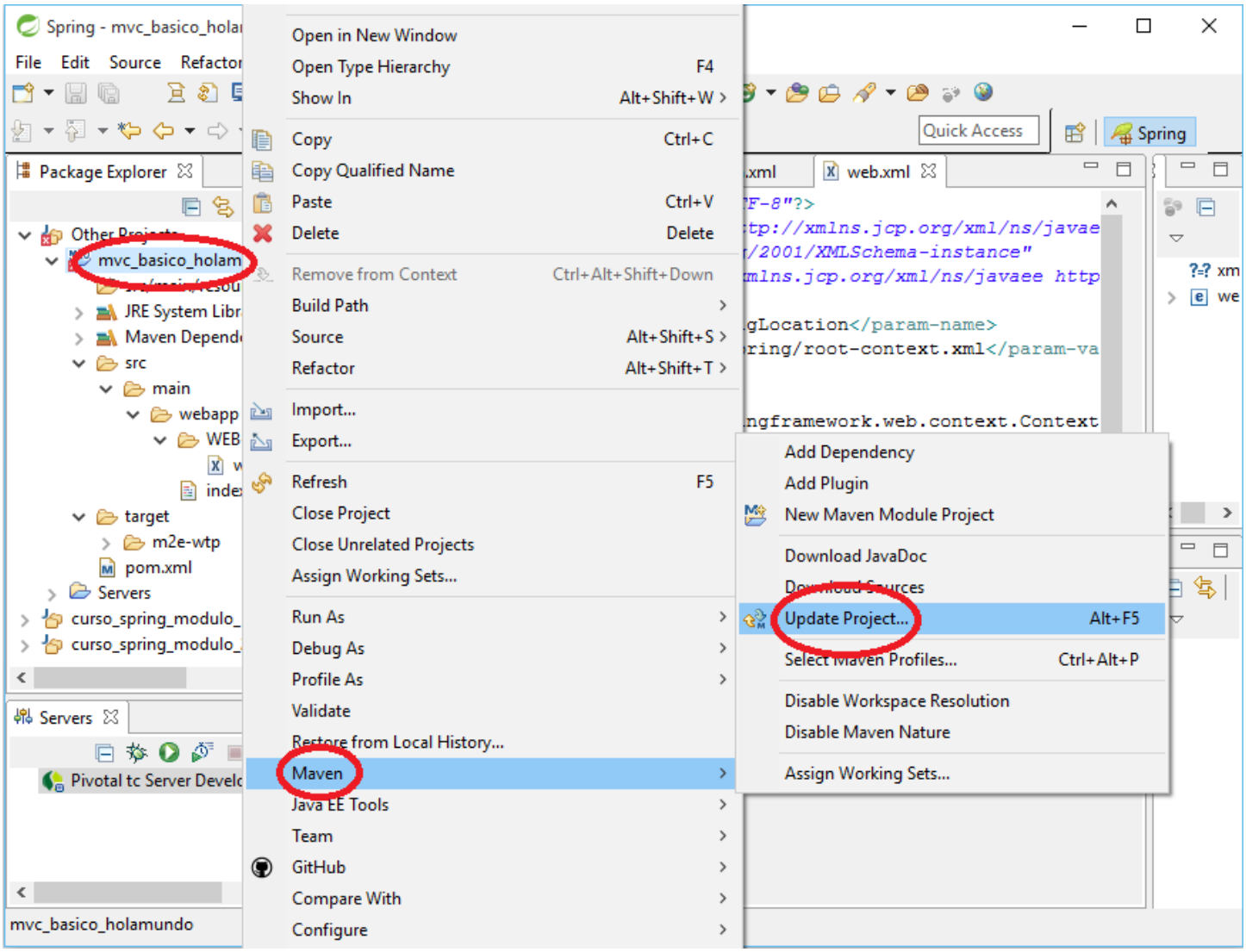
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.bolsadeideas.ejemplos</groupId>
    <artifactId>mvc_basico_holamundo</artifactId>
    <packaging>war</packaging>
    <version>0.0.1-SNAPSHOT</version>
    <name>mvc_basico_holamundo Maven Webapp</name>
    <properties>
        <java-version>1.8</java-version>
        <org.springframework-version>4.2.0.RELEASE</org.springframework-version>
    </properties>
    <dependencies>
        <!-- Spring -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${org.springframework-version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${org.springframework-version}</version>
        </dependency>
        <!-- Servlet -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.2</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp.jstl</groupId>
            <artifactId>javax.servlet.jsp.jstl-api</artifactId>
            <version>1.2.1</version>
        </dependency>

        <!-- Test -->
```

```
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>4.7</version>
                <scope>test</scope>
            </dependency>
        </dependencies>
        <build>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.1</version>
                    <configuration>
                        <source>${java-version}</source>
                        <target>${java-version}</target>
                    </configuration>
                </plugin>
            </plugins>
        </build>
    </project>
```

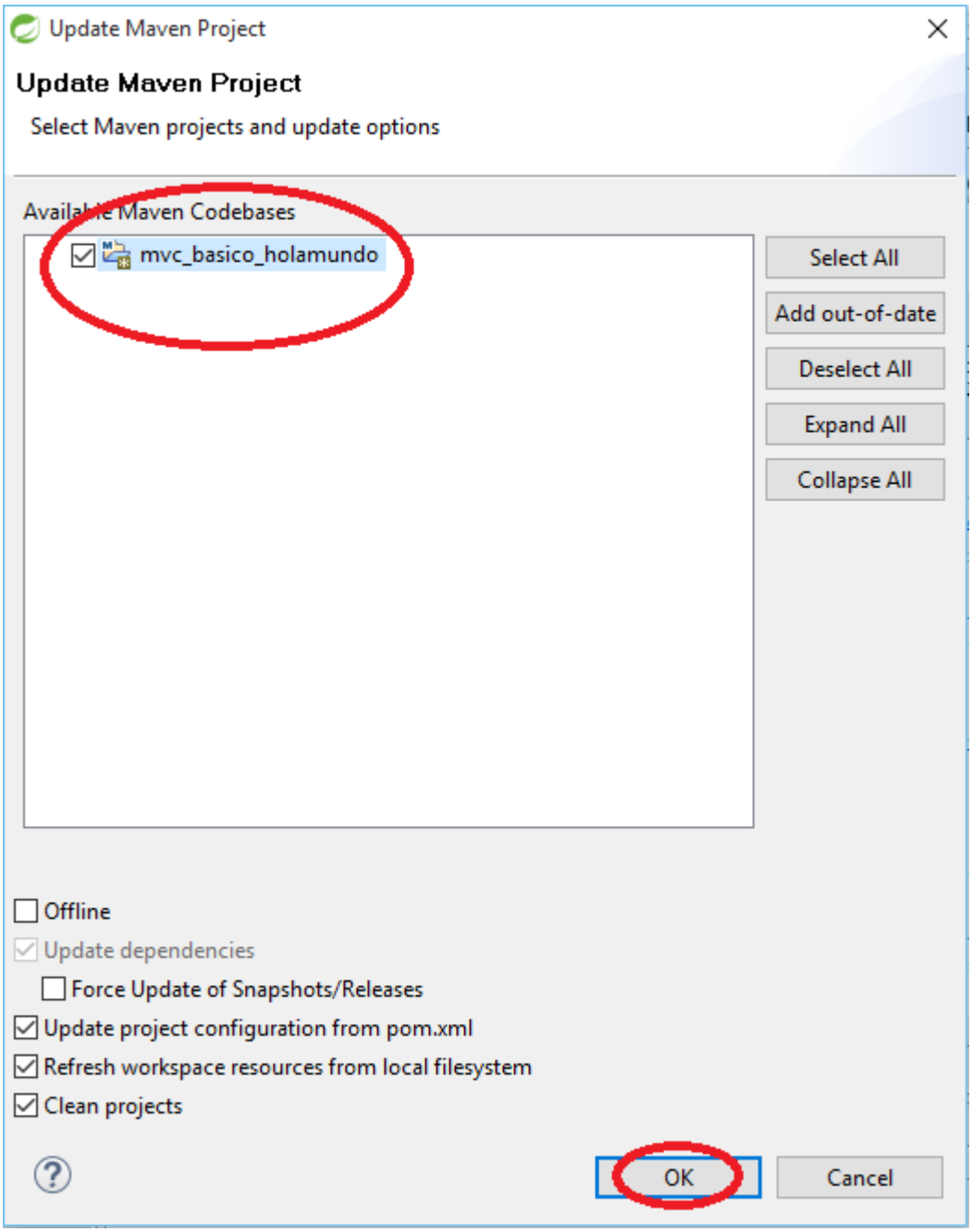


- Guardamos los cambios

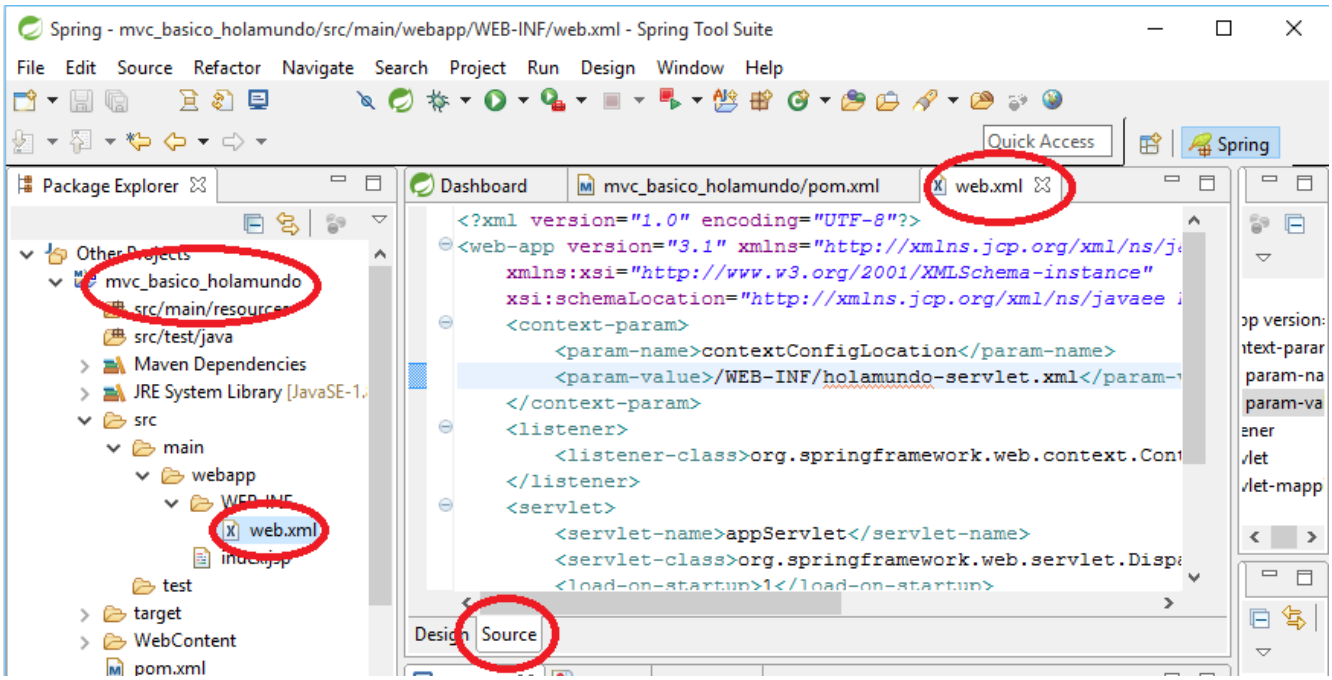
7. Luego hacemos un **Maven->Update Project…** para actualizar las dependencias y configuraciones en el proyecto.



- Clic OK

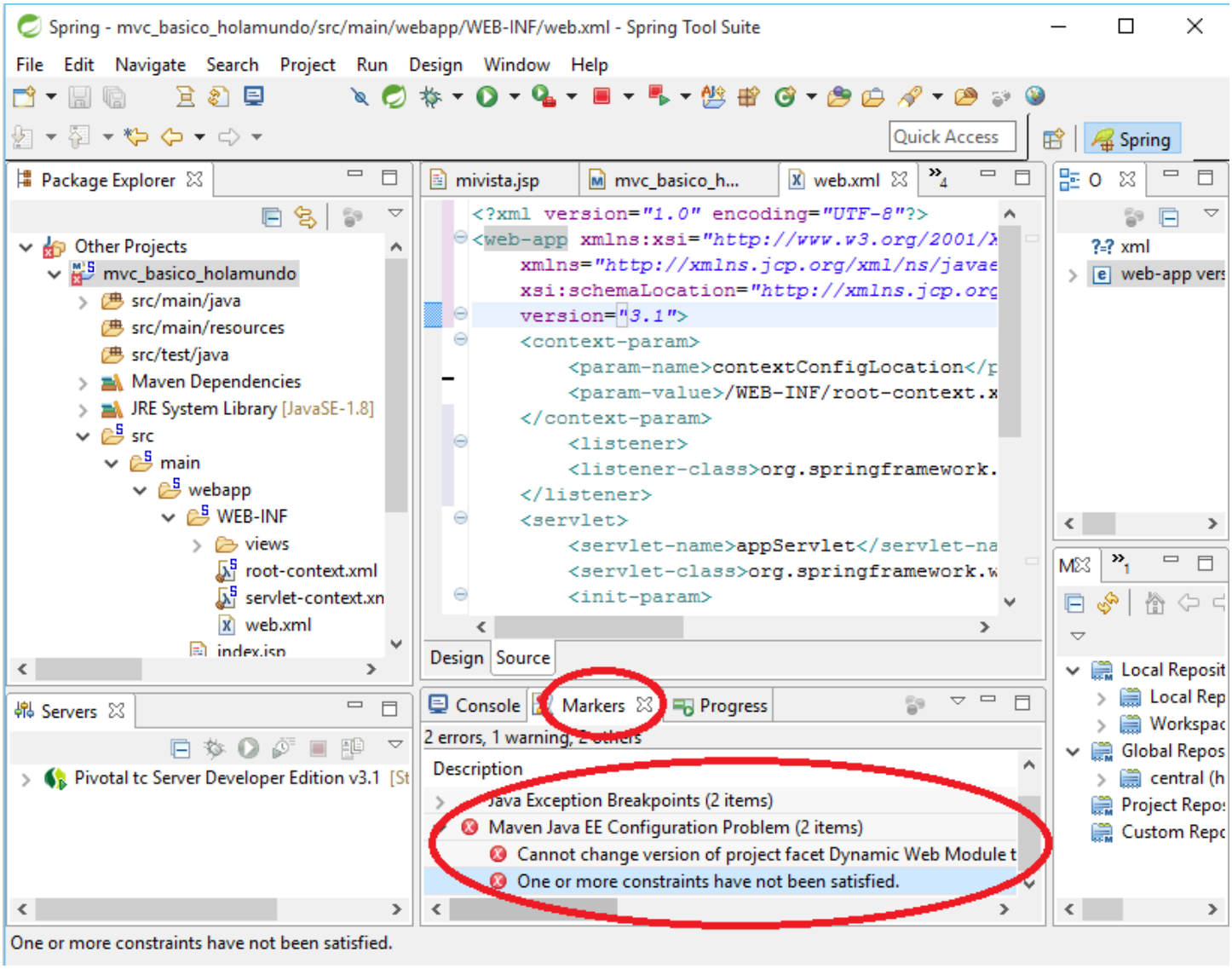

Update Maven Project
Update Maven Project
Select Maven projects and update options
Available Maven Codebases
mvc_basico_holamundo
Select All
Add out-of-date
Deselect All
Expand All
Collapse All
Offline
Update dependencies
Force Update of Snapshots/Releases
Update project configuration from pom.xml
Refresh workspace resources from local filesystem
Clean projects
OK
Cancel

8. Luego Modificamos el web.xml del proyecto para configurar Spring MVC.
    - Modificar nuestro web.xml generado, poner atención en el código en negrita:

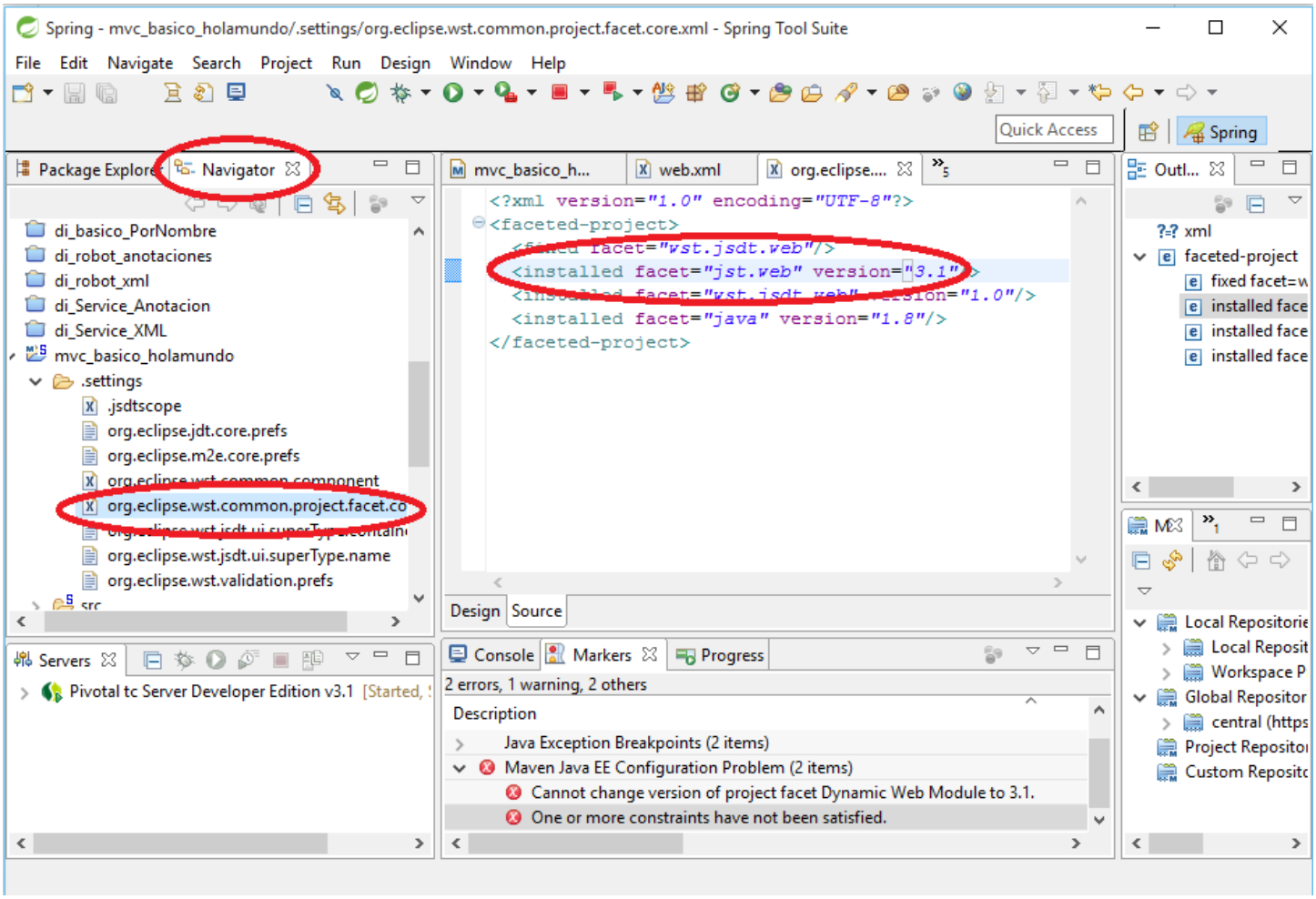



```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/root-context.xml</param-value>
    </context-param>
        <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <servlet>
        <servlet-name>appServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/servlet-context.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>appServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

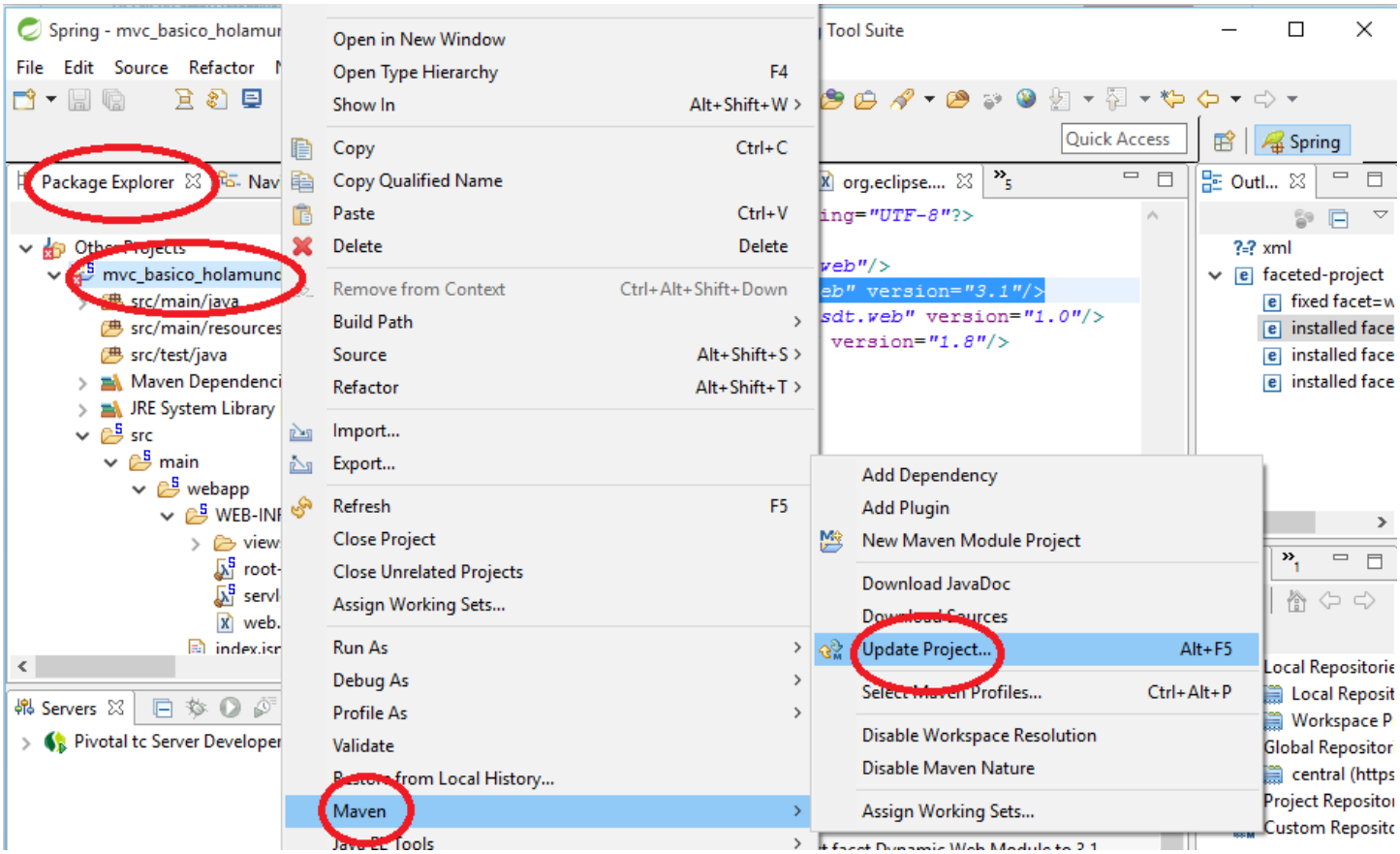
9. Volvemos a realizar un **Maven->Update Project…** para actualizar las configuraciones y facet del proyecto.
10. Luego del **Maven->Update Project…** Notamos que aparecen dos problemas, específicamente en las configuraciones facet del proyecto:
    - Observamos en la pestaña Markers los errores en rojo



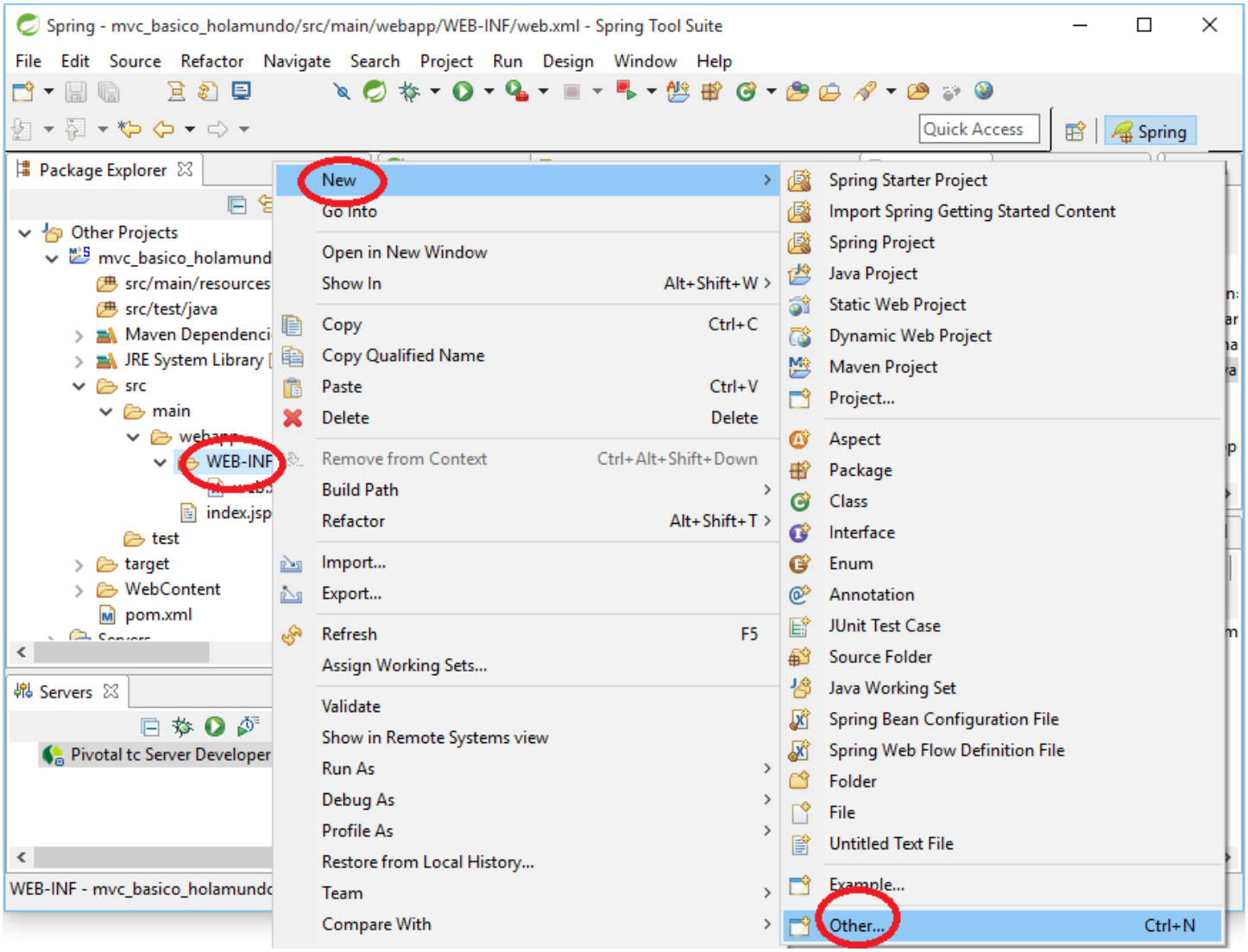
11. Para solucionarlo tenemos que modificar las configuraciones **project facet del proyecto**, entonces:
    - En el menú principal **Window -> Show View -> Other -> General -> Navigator**
    - Nos vamos a la pestaña **Navigator** y buscamos la carpeta **.settings** y expandimos y abrimos el archivo **org.eclipse.wst.common.project.facet.core.xml**.
    - Luego cambiamos la versión del **dynamic web module** a "**3.1**", ejemplo: **<installed facet="jst.web" version="3.1"/>**



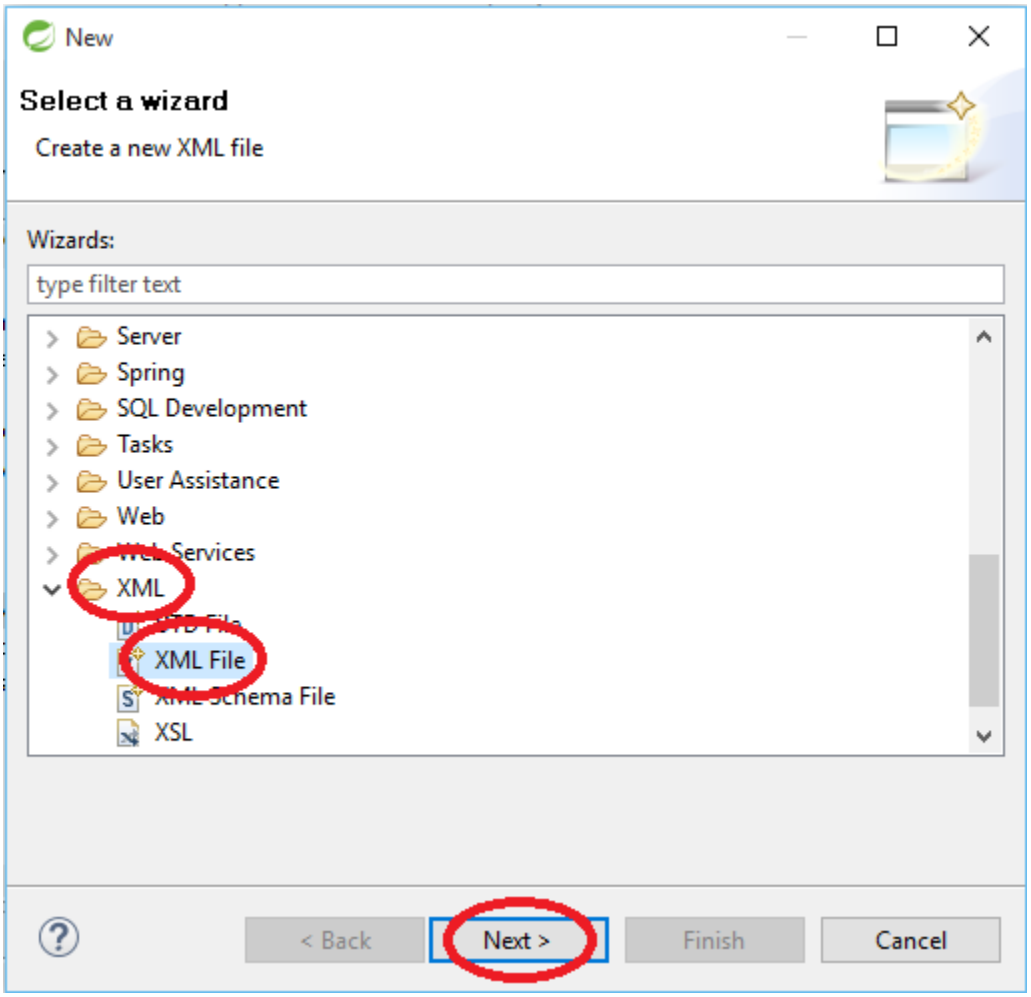
- Volvemos a la pestaña **Package explorer**
- Luego hacemos un **Maven->Update Project…**



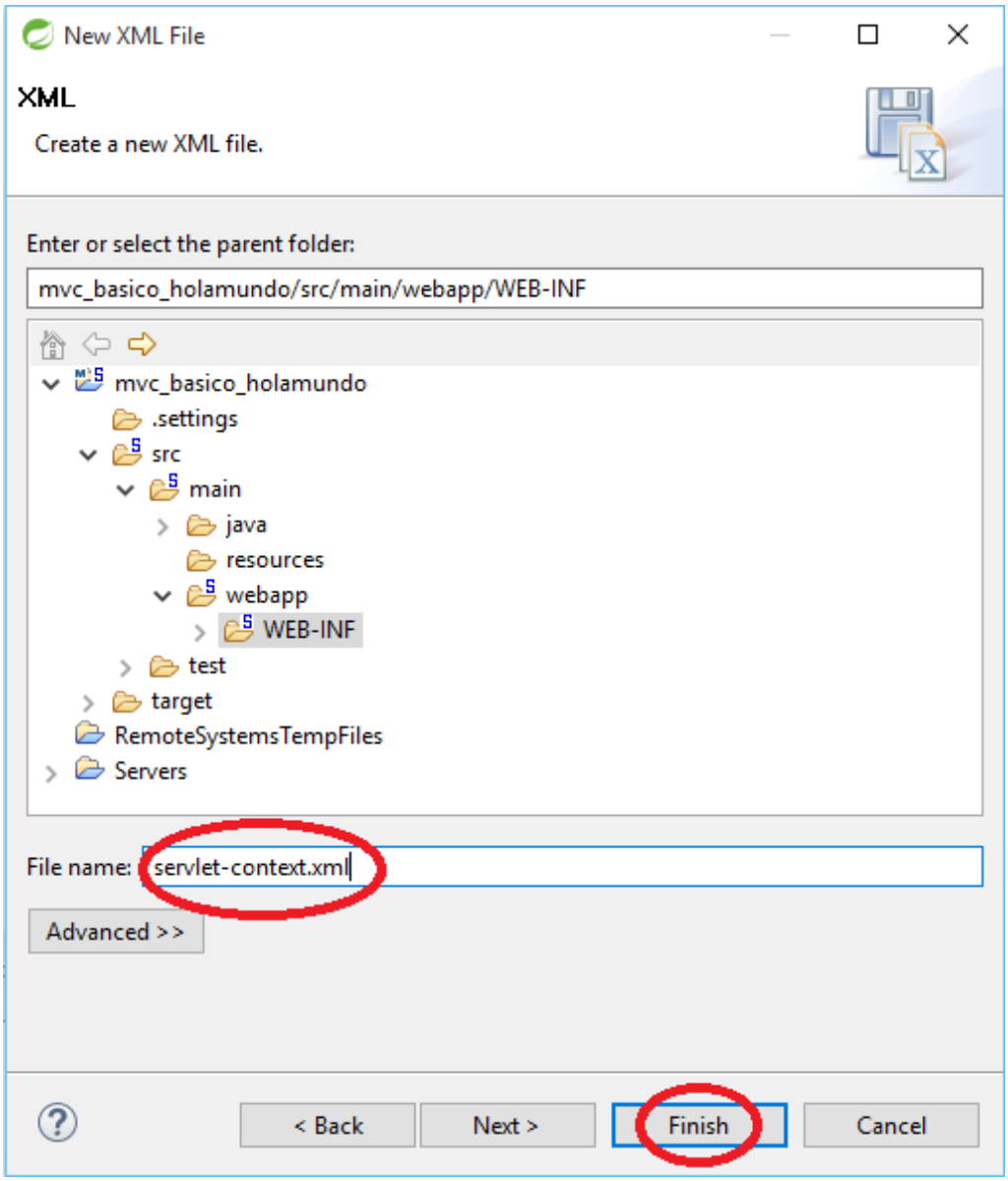
12. Creamos el archivo xml **servlet-context.xml** en el directorio WEB-INF.
    - Clic derecho sobre **/WEB-INF** y seleccionar **New->Other**.



- Expandir XML.
- Seleccionar XML File.
- Clic Next.



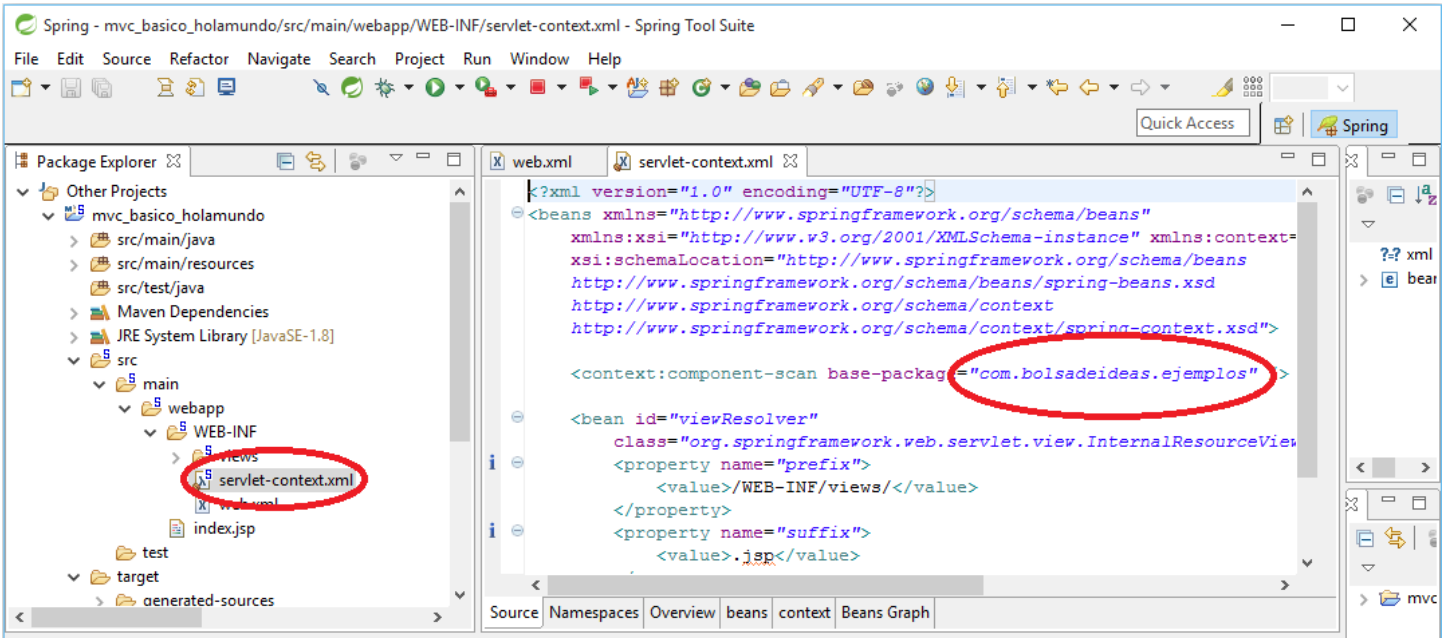
- Para el Campo File name, ingrese: **servlet-context.xml**
- Clic Finish.



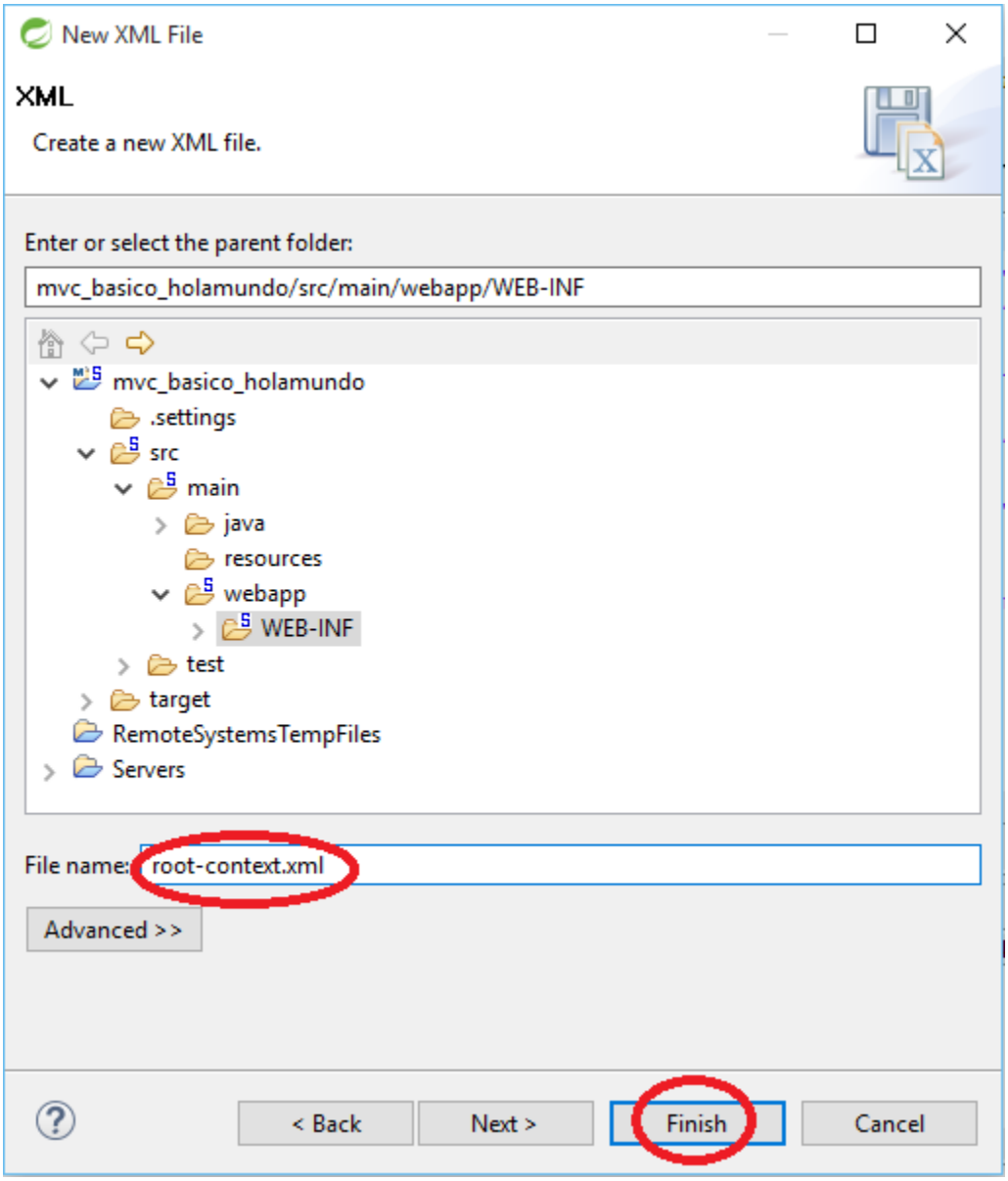
13. Modificamos el archivo **servlet-context.xml** generado por el IDE:

```
<?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"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

   <context:component-scan base-package="com.bolsadeideas.ejemplos" />

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

</beans>
```

- Al especificar el elemento **<context:component-scan ..>** en el archivo de configuración de spring, le indicamos a Spring MVC que busque todos los controladores con la anotación **@Controller** en el package base.
- Con la declaración configura a las clase controladoras marcadas con la anotación @Controller.
- Esta etiqueta registra el DefaultAnnotationHandlerMapping y el beans AnnotationMethodHandlerAdapter que se requieren Spring MVC con anotaciones para despachar las solicitudes hacia los controladores @Controller.



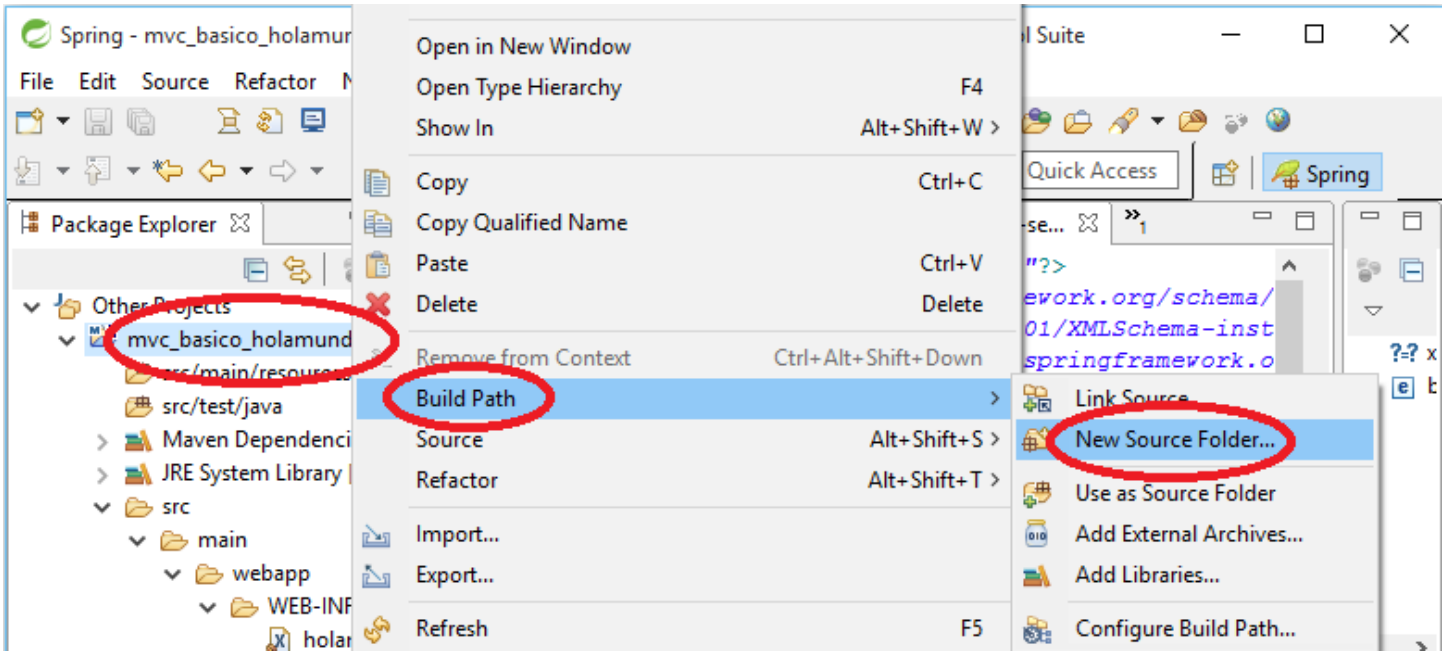
14. Similar a lo anterior, creamos el archivo de contexto xml **root-context.xml** en el directorio WEB-INF.
    - Clic derecho sobre **/WEB-INF** y seleccionar **New->Other**.
    - Expandir **XML-> XML File**.
    - Clic Next.


New XML File
XML
Create a new XML file.
Enter or select the parent folder:
mvc_basico_holamundo/src/main/webapp/WEB-INF
mvc_basico_holamundo
.settings
src
main
java
resources
webapp
WEB-INF
test
target
RemoteSystemsTempFiles
Servers
File name:
root-context.xml
Advanced >>
< Back
Next >
Finish
Cancel

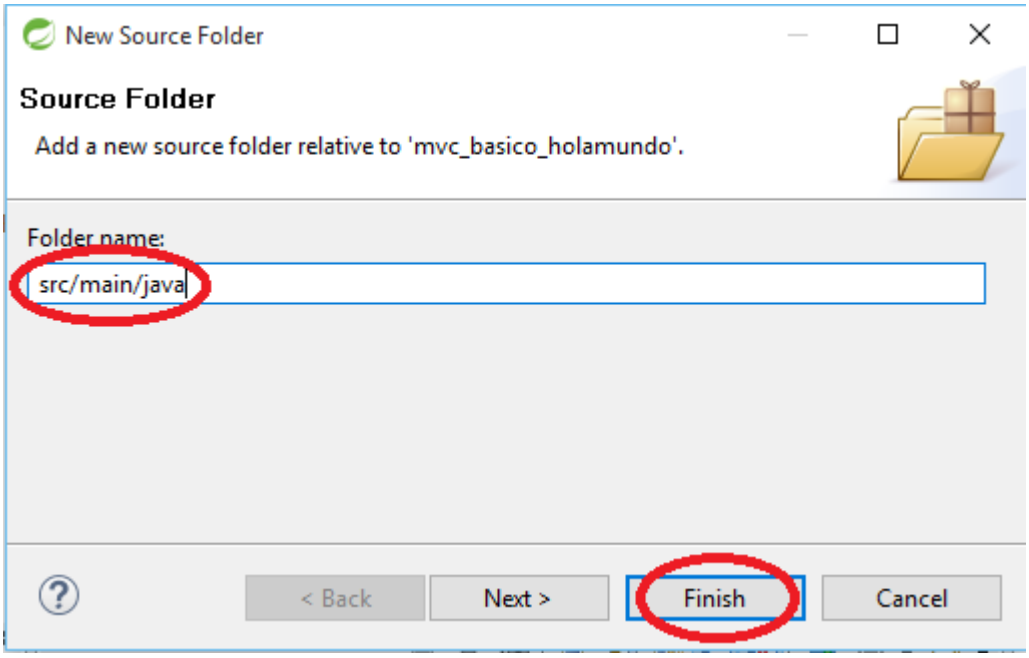
15. Modificamos el archivo **root-context.xml** generado por el IDE:

```
<?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">

    <!-- Root Context: defines shared resources visible to all other web components -->

</beans>
```

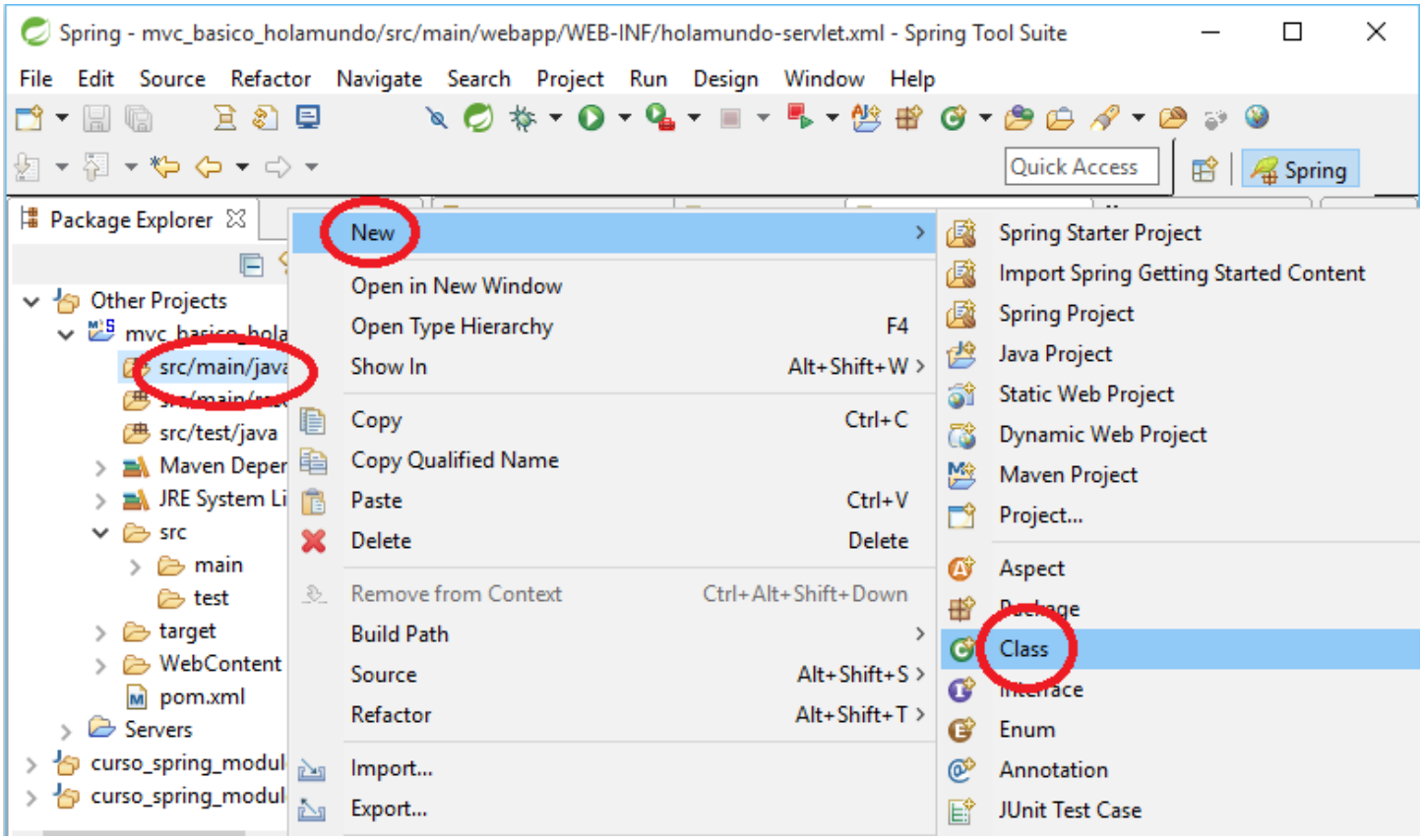
16. Ahora vamos a crear un simple Spring MVC Controller.
    - Vamos a crear el directorio **src/main/java**. Con un clic derecho sobre el proyecto y seleccionar **Build Path->New Source Folder**.



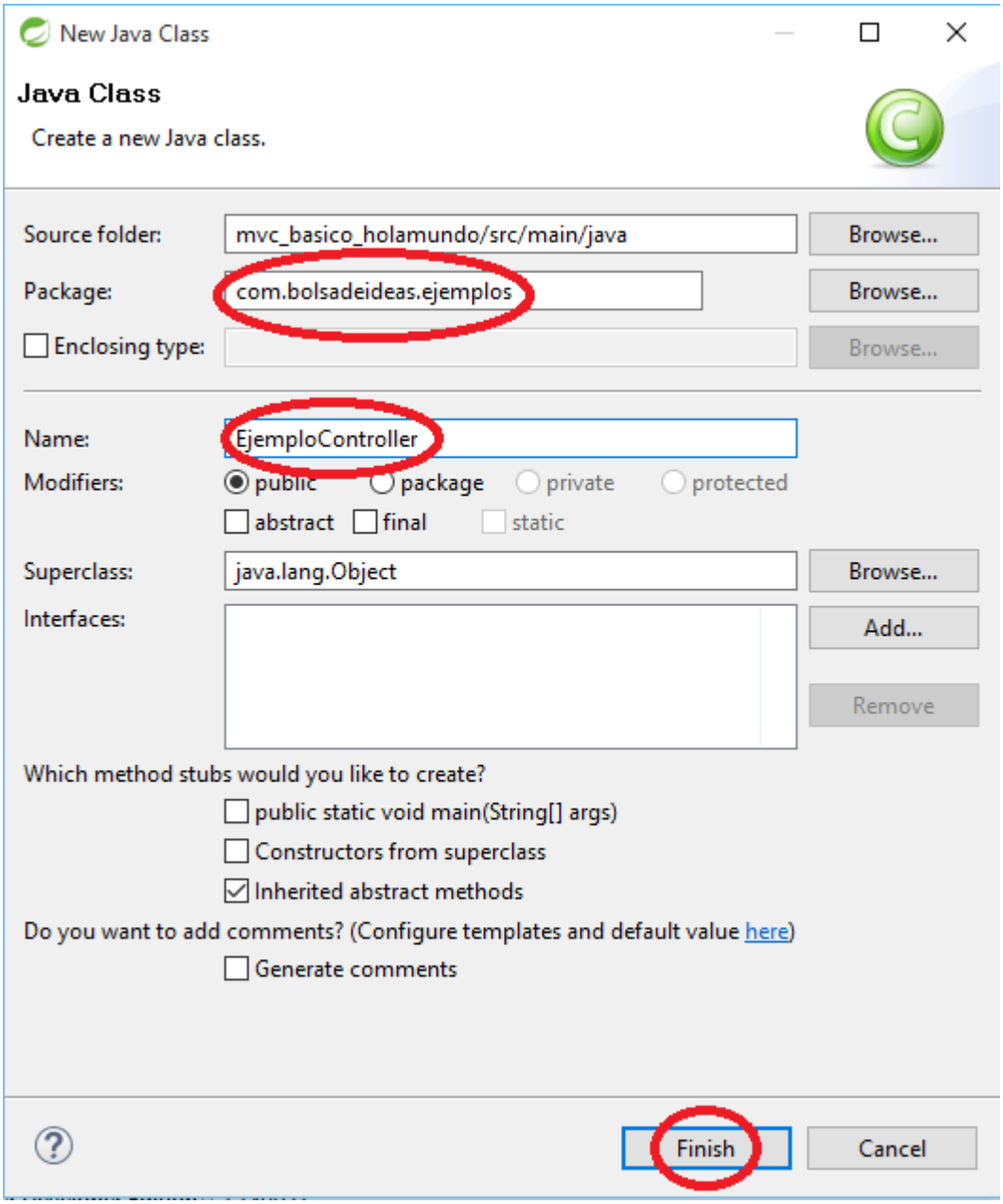
- Para el campo Folder, ingresamos **src/main/java**
- Clic Finish



- Creamos la clase **EjemploController**.java bajo el directorio **src/main/java**. El **EjemploController**.java es nuestra clase controladora.
- Clic derecho **src/main/java** y seleccionar **New->Class**.



- Para el campo Package, ingrese **com.bolsadeideas.ejemplos**
- Para el campo Name, ingrese **EjemploController**
- Clic Finish

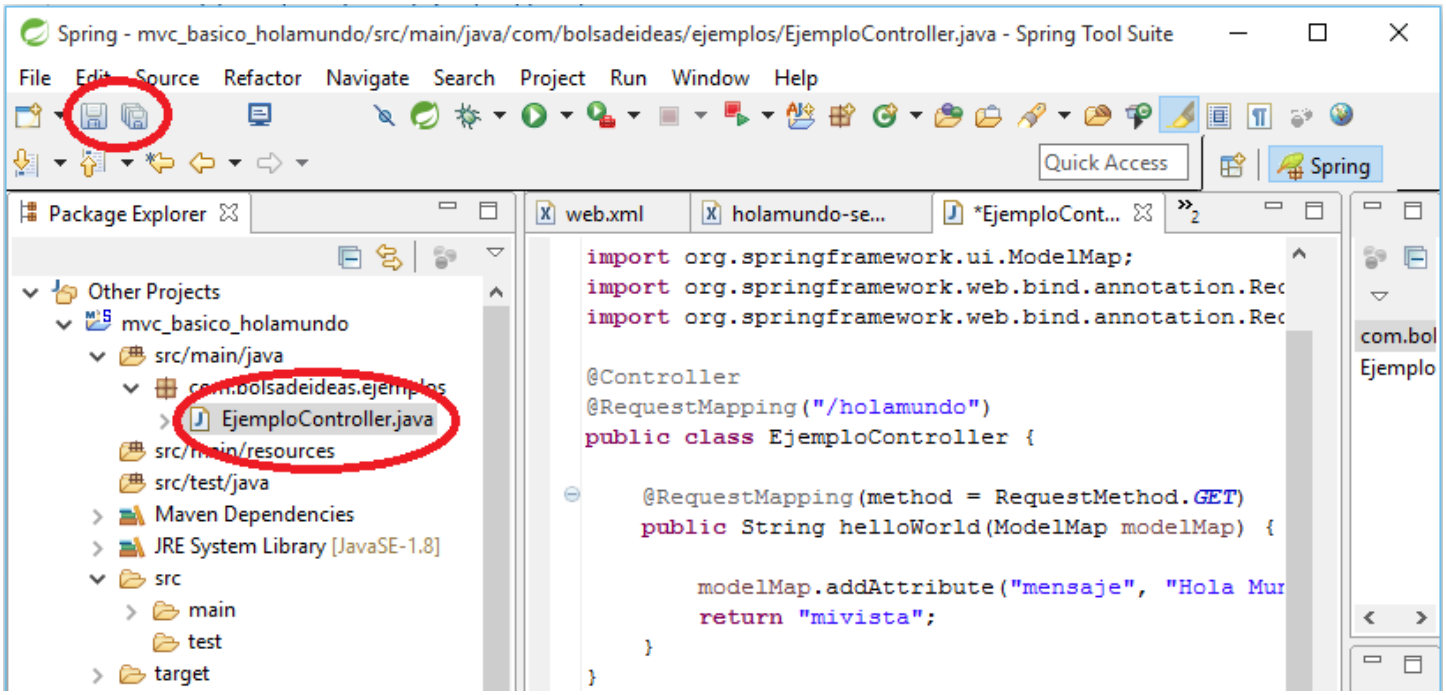

New Java Class
Java Class
Create a new Java class.
Source folder:
mvc_basico_holamundo/src/main/java
Browse...
Package:
com.bolsadeideas.ejemplos
Browse...
Enclosing type:
Browse...
Name:
EjemploController
Modifiers:
public
package
private
protected
abstract
final
static
Superclass:
java.lang.Object
Browse...
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel

- Modificamos la clase generada EjemploController.java:

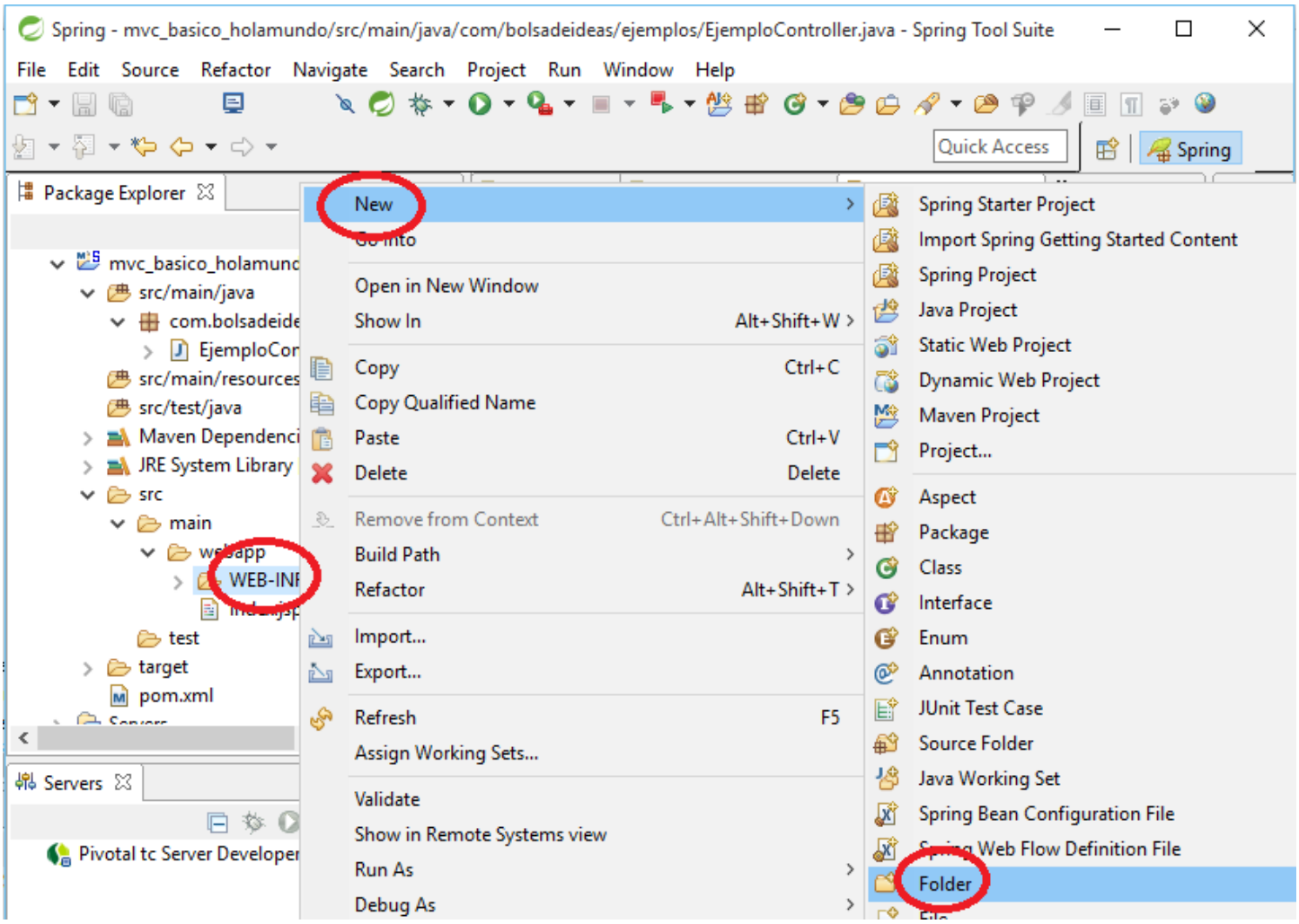
```
package com.bolsadeideas.ejemplos;

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("/holamundo")
public class EjemploController {

    @RequestMapping(method = RequestMethod.GET)
    public String helloWorld(ModelMap modelMap) {

          modelMap.addAttribute("mensaje", "Hola Mundo Spring Framework!");
          return "mivista";
    }
}
```

Hasta este punto podríamos experimentar algunos errores de compilación. Es es debido a que no hemos guardado aun los cambios en el archivo pom.xml, lo que significa que los archivos de Spring Framework (jar) no se han descargado todavía. Tan pronto como pom.xml se guarda, los archivos de dependencia se descargan desde los repositorios de Maven.

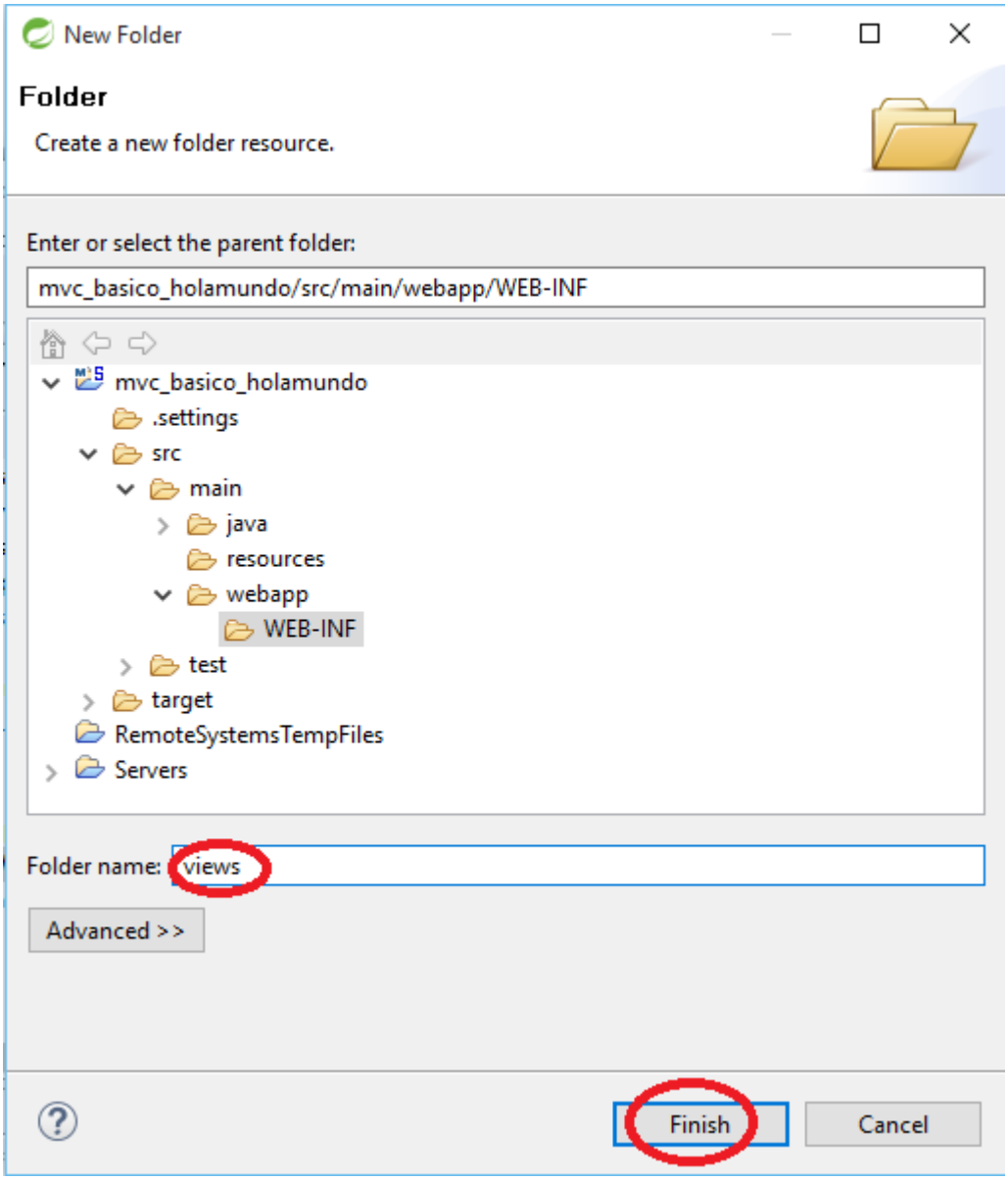
- Guardamos todos los cambios (Save All).



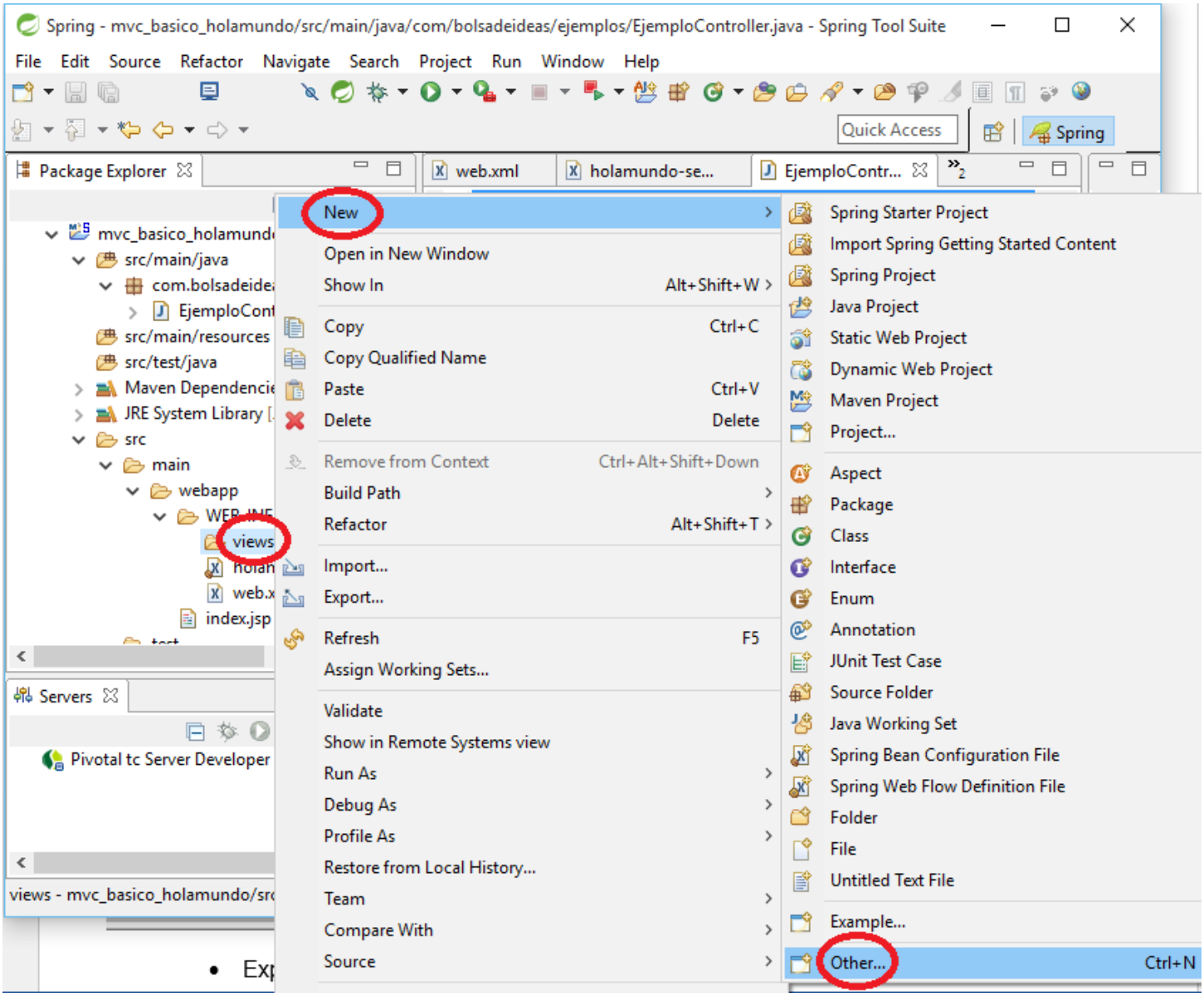
17. Lo que sigue sería crear la vista JSP para nuestro controlador.
    - Lo primero es crear el directorio **views** dentro del **WEB-INF**. En este directorio vamos a guardar todas nuestras vistas jsp del proyecto, esto es porque en el xml de spring que creamos más arriba, le especificamos que el directorio **/WEB-INF/views** será el lugar donde mantendremos nuestras vistas *.jsp.
    - Clic derecho sobre **WEB-INF** y seleccionar **New->Folder**.



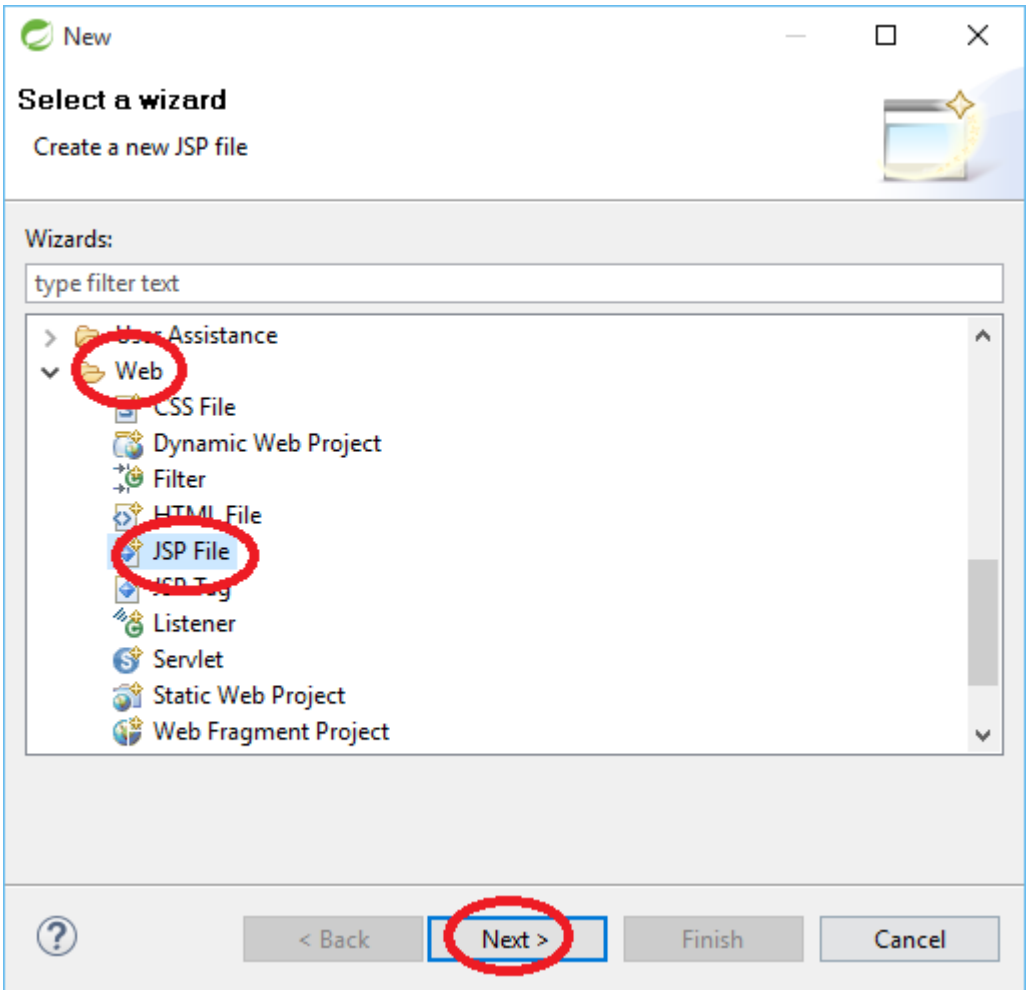
- Para el campo Folder, ingrese **views**
- Clic Finish.


New Folder
Folder
Create a new folder resource.
Enter or select the parent folder:
mvc_basico_holamundo/src/main/webapp/WEB-INF
mvc_basico_holamundo
.settings
src
main
java
resources
webapp
WEB-INF
test
target
RemoteSystemsTempFiles
Servers
Folder name: views
Advanced >>
Finish
Cancel

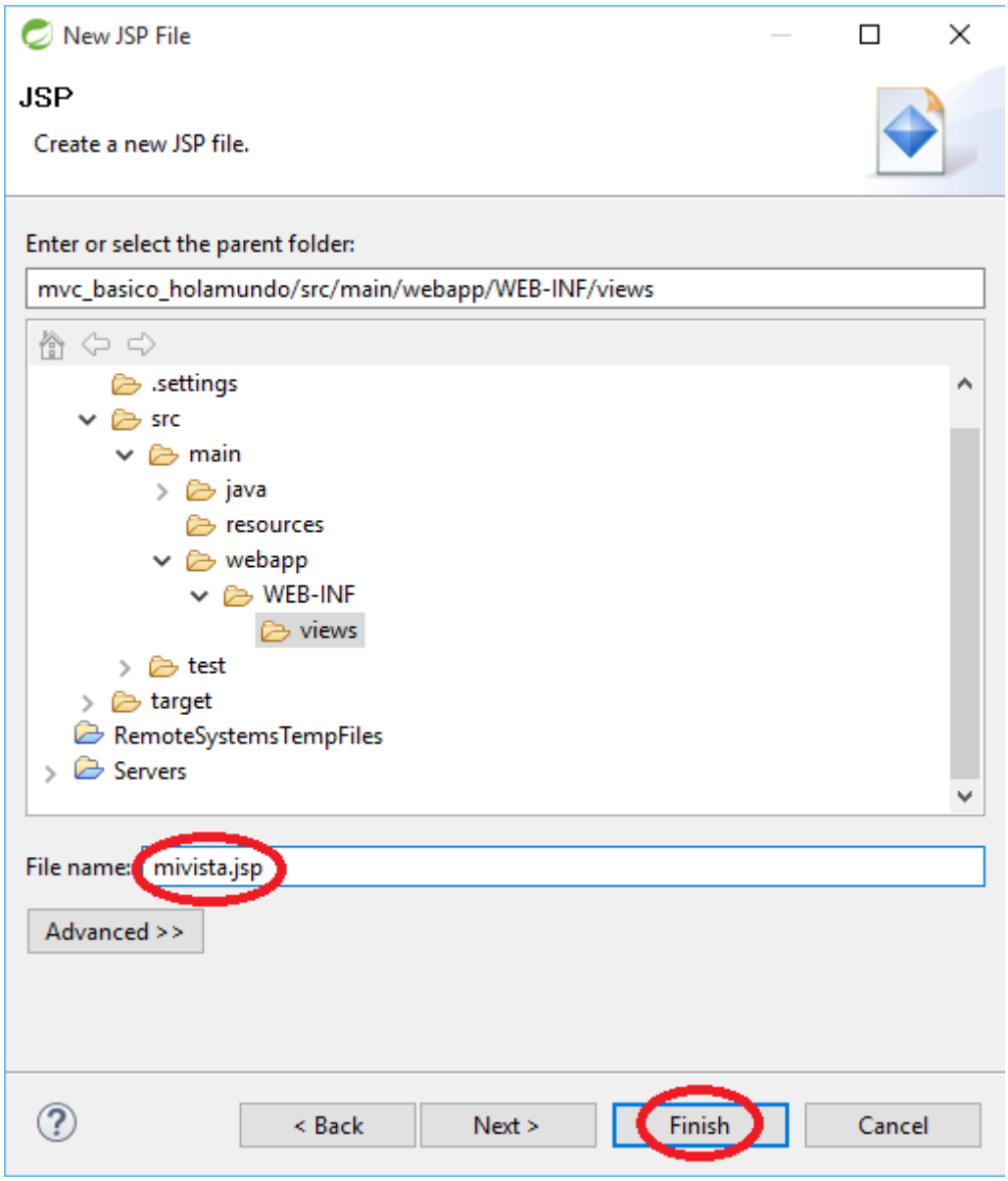
- Creamos la vista jsp **mivista.jsp** bajo el directorio **views**, entonces clic derecho sobre **WEB-INF/views** y seleccionar **New->Other**



- Expandir Web.
- Seleccionar JSP File.
- Clic Next.

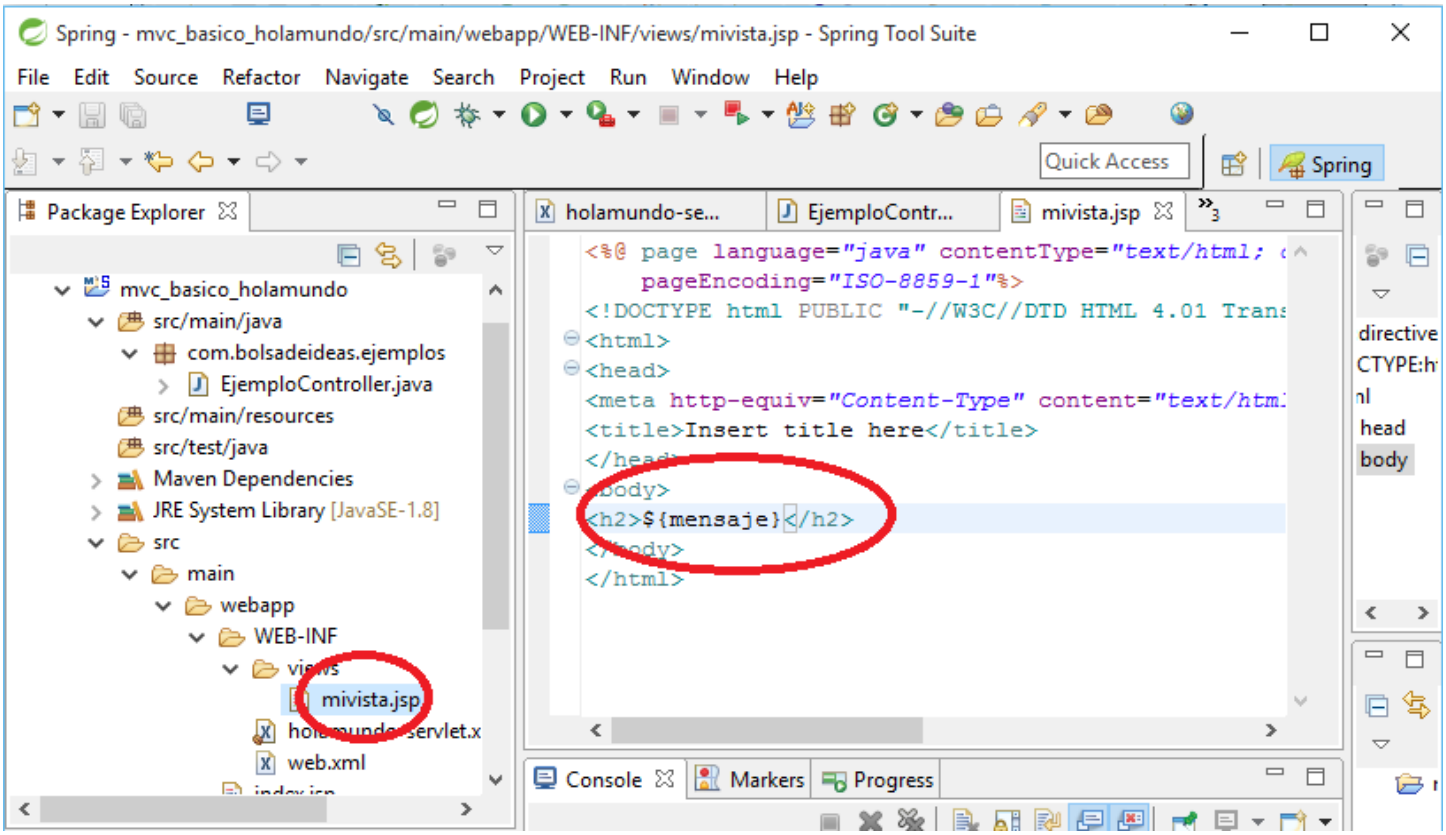


- Para el campo **File name**, ingrese **mivista.jsp**.
- Clic Finish

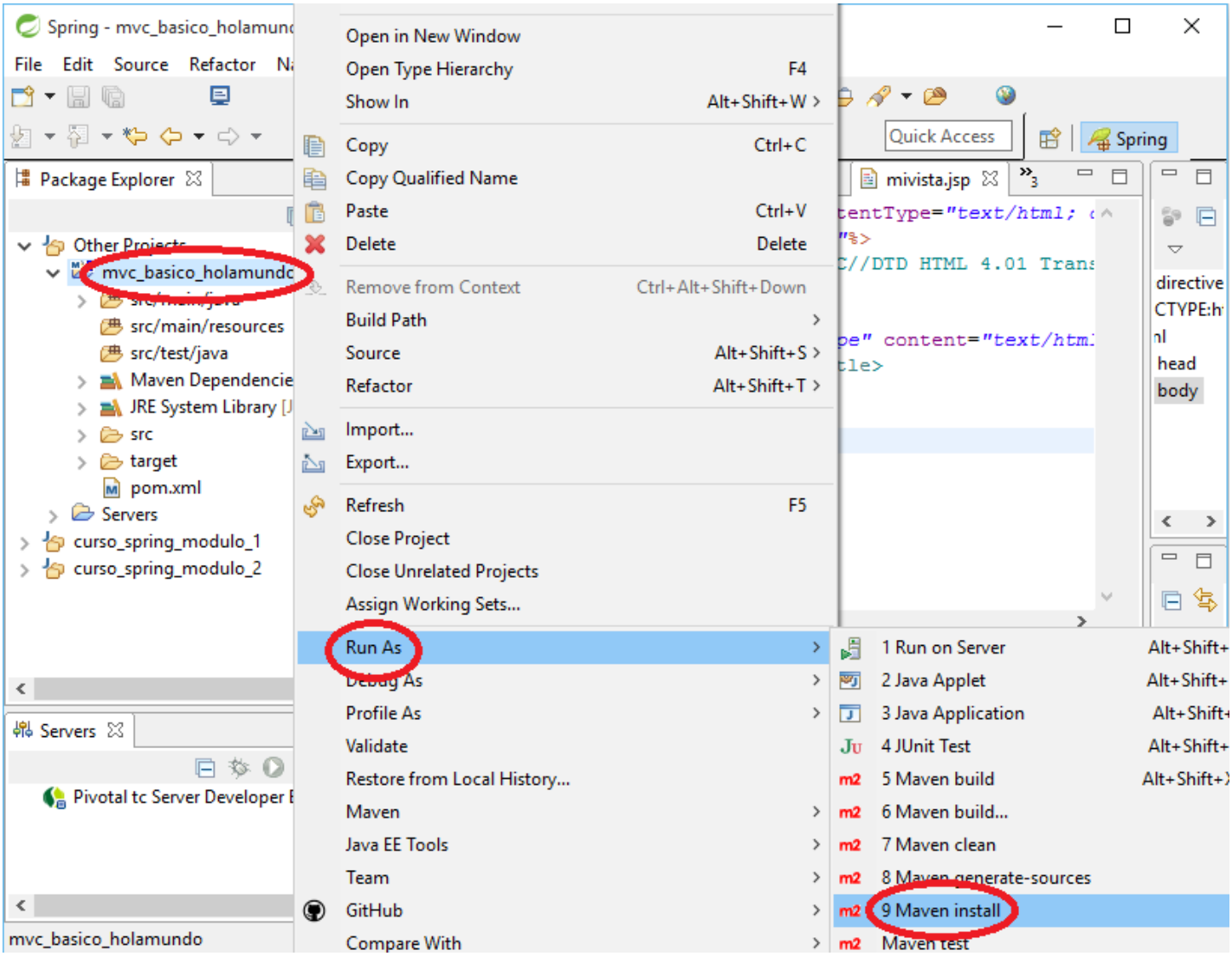


18. Modificamos el código de la **mivista.jsp** por el siguiente:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2>${mensaje}</h2>
</body>
</html>
```

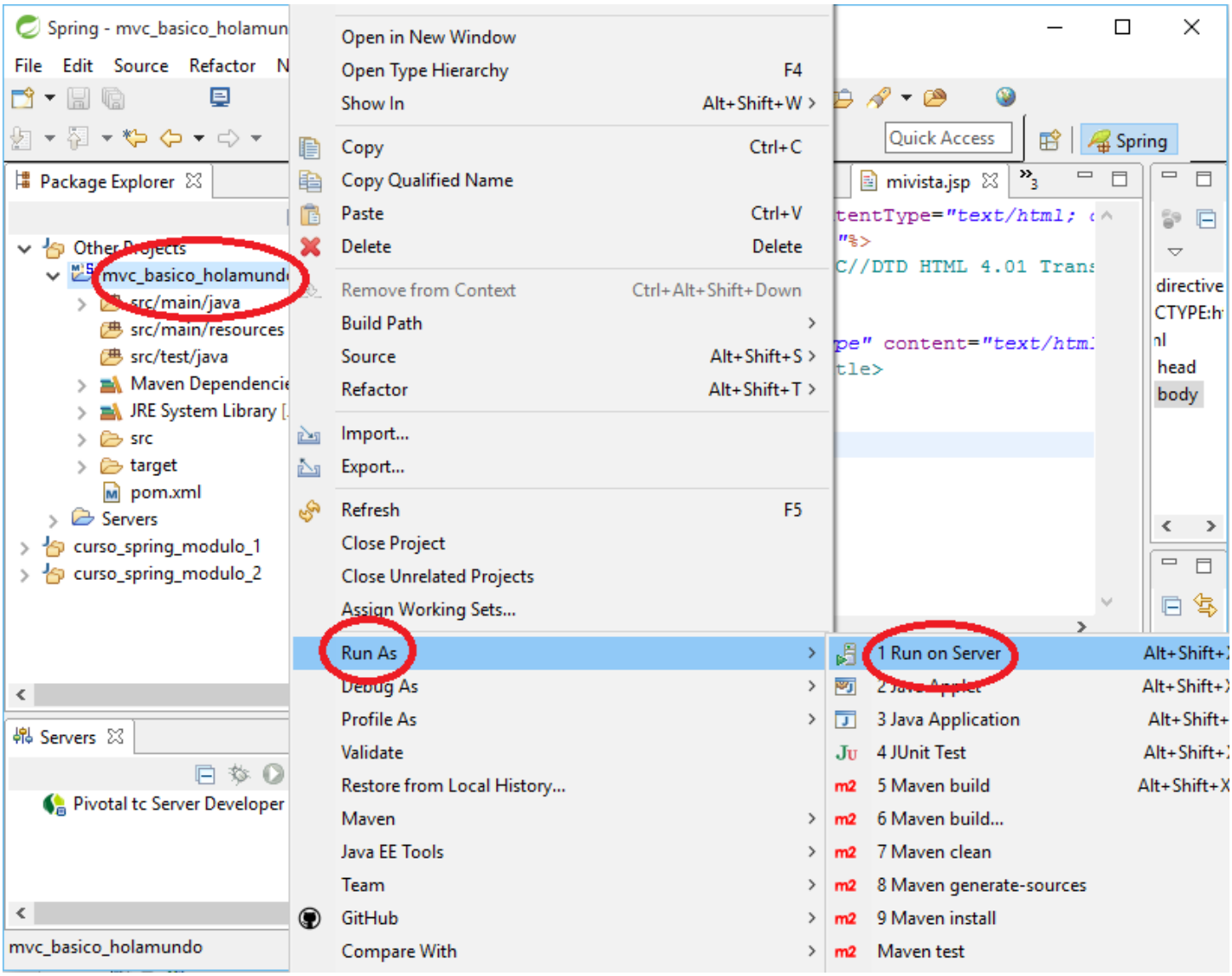


19. Generamos el package de maven.
    - Clic derecho sobre el proyecto y seleccionar **Run As->Maven Install**.

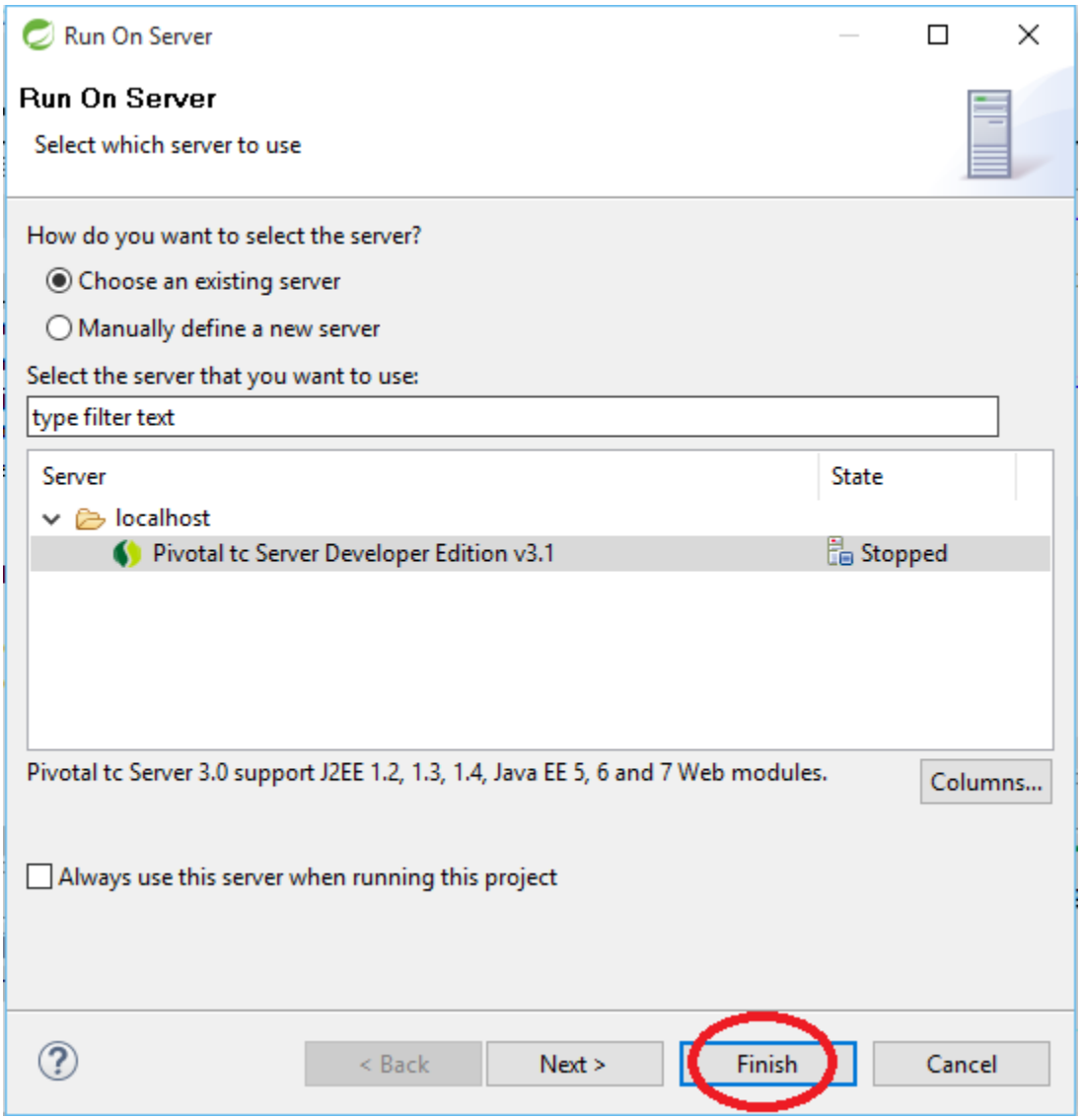


20. Finalmente hacemos un **Maven->Update Project…** para actualizar las dependencias y configuraciones en el proyecto

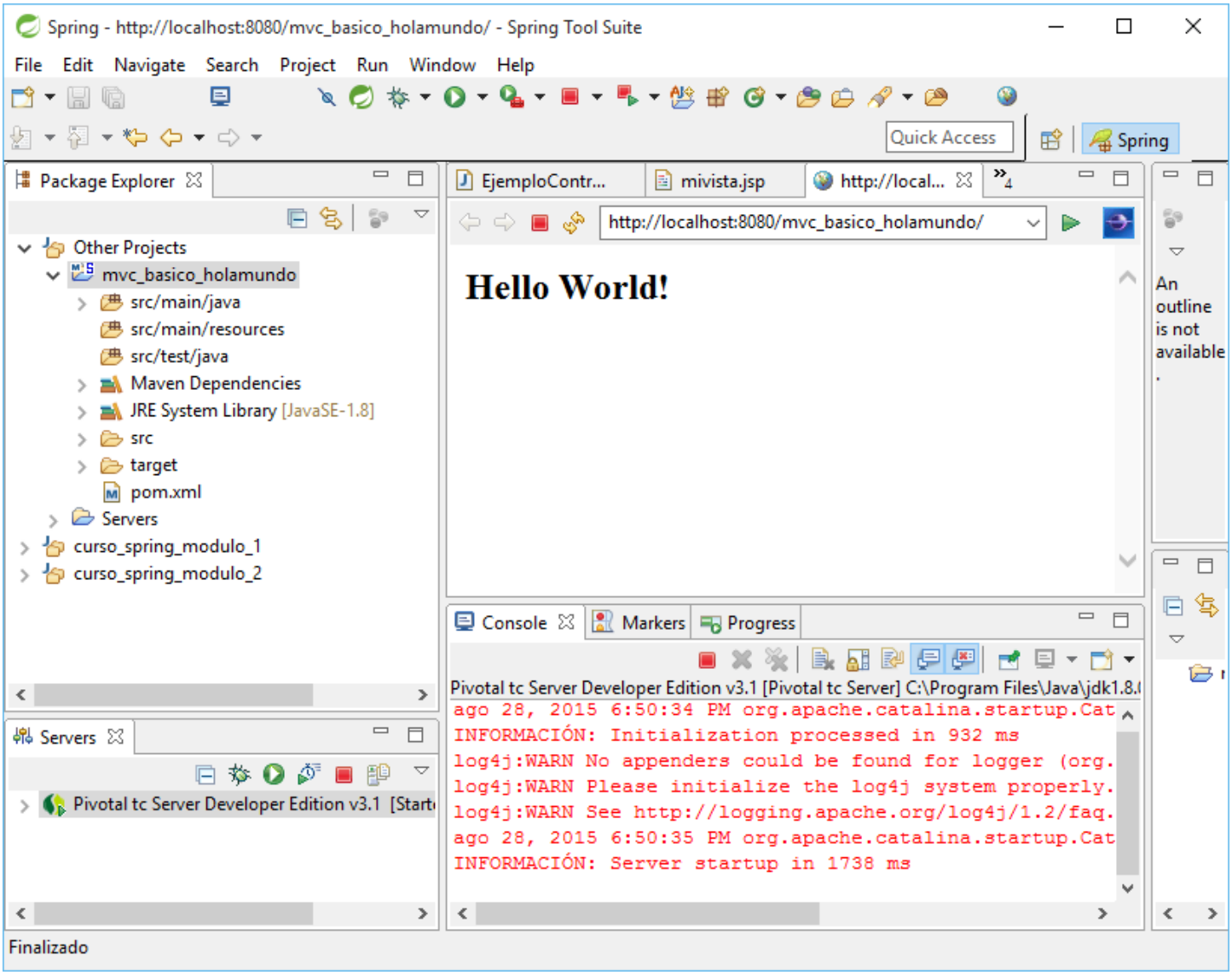
21. Ejecutamos la aplicación en el servidor **Pivotal tc Server Developer** de Spring. (También podemos ejecutar sobre otros servidores como GlassFish, JBoss o Tomcat.)
    - Clic derecho sobre el proyecto y seleccionar **Run As->Run on Server**.



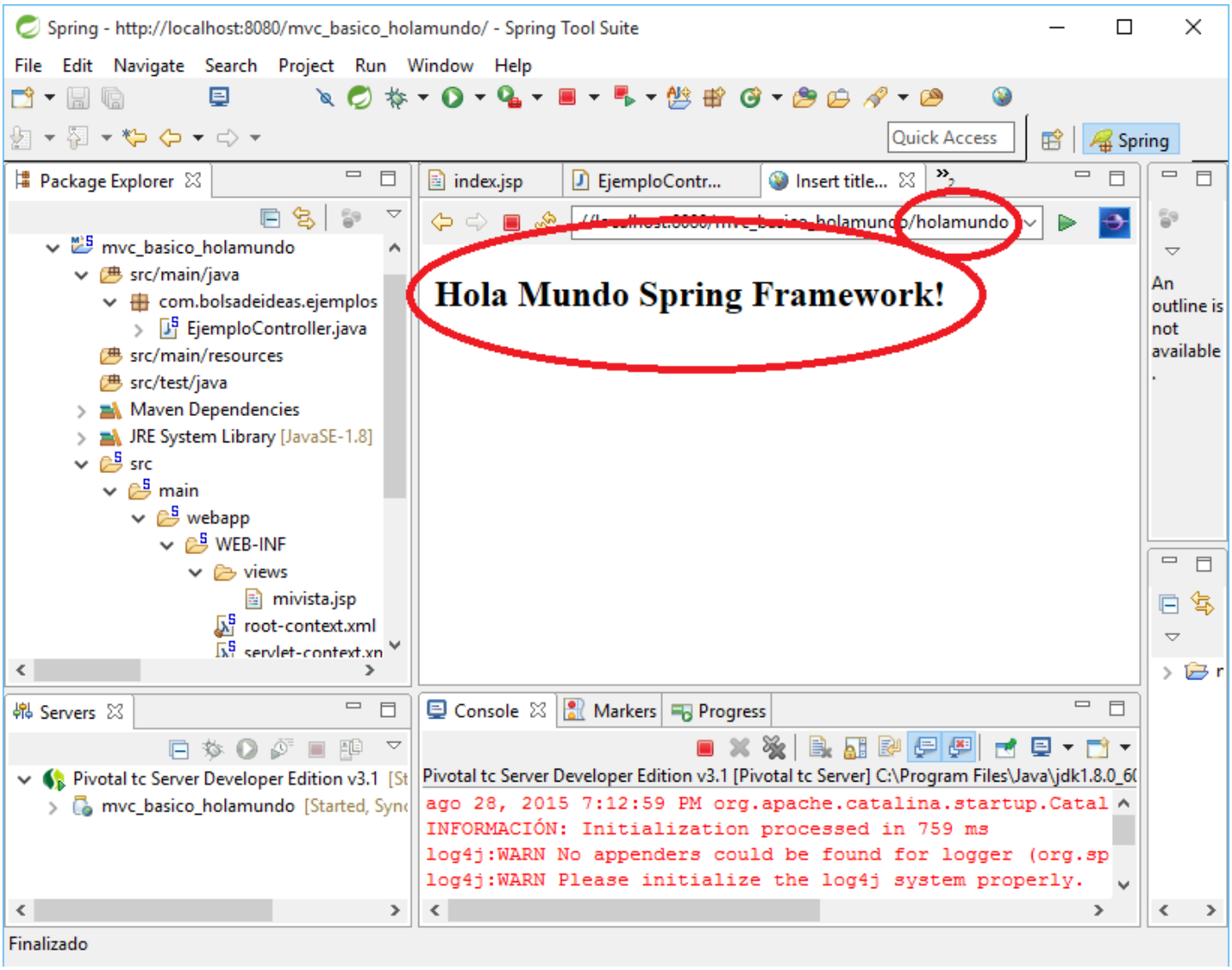
- Seleccionamos **Pivotal tc Server Developer Edition** de Spring
- Clic Finish



- Observe que aparece el saludo Hello World!, ya sea en un navegador integrado o en un navegador externo (dependiendo de cómo haya configurado el IDE.)



- Agregue en la URL, al final: **/holamundo** y refresque el navegador.



Ejercicio 2: Modificar el proyecto Spring MVC "mvc_basico_holamundo"

Podemos agregar nuevo métodos de petición (handlers request) en el controlador usando la anotación @RequestMapping.

La anotación @RequestMapping puede ser usada a nivel de clase y método.

El tipo de retorno de un método handler puede ser uno de los siguientes:

- String
- ModeAndView
- View (No se muestra en este ejercicio)
- Void (No se muestra en este ejercicio)

En la vistas JSP se puede acceder al modelo a tarves de ${..}

1. Modificamos el controlador EjemploController.java como se muestra a continuación. Los fragmentos de código agregados se destacan en **negrita y rojo**.

```
package com.bolsadeideas.ejemplos;

import java.util.Map;

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

@Controller
@RequestMapping("/holamundo")
public class EjemploController {

   @RequestMapping(method = RequestMethod.GET)
   public String helloWorld(ModelMap modelMap) {

         modelMap.addAttribute("mensaje", "Hola Mundo Spring Framework!");
         return "mivista";
   }

   @RequestMapping(value = "/spring", method = RequestMethod.GET)
   public String helloWorld(Map<String, Object> map) {

         map.put("mensaje2", "Spring!");
         return "mivista2";
   }
```

```
    @RequestMapping(value = "/springmvc", method = RequestMethod.GET)
    public ModelAndView helloWorld() {

          ModelAndView modelAndView = new ModelAndView("mivista3");
          modelAndView.addObject("mensaje3", "Spring MVC!");

          return modelAndView;
    }
}
```

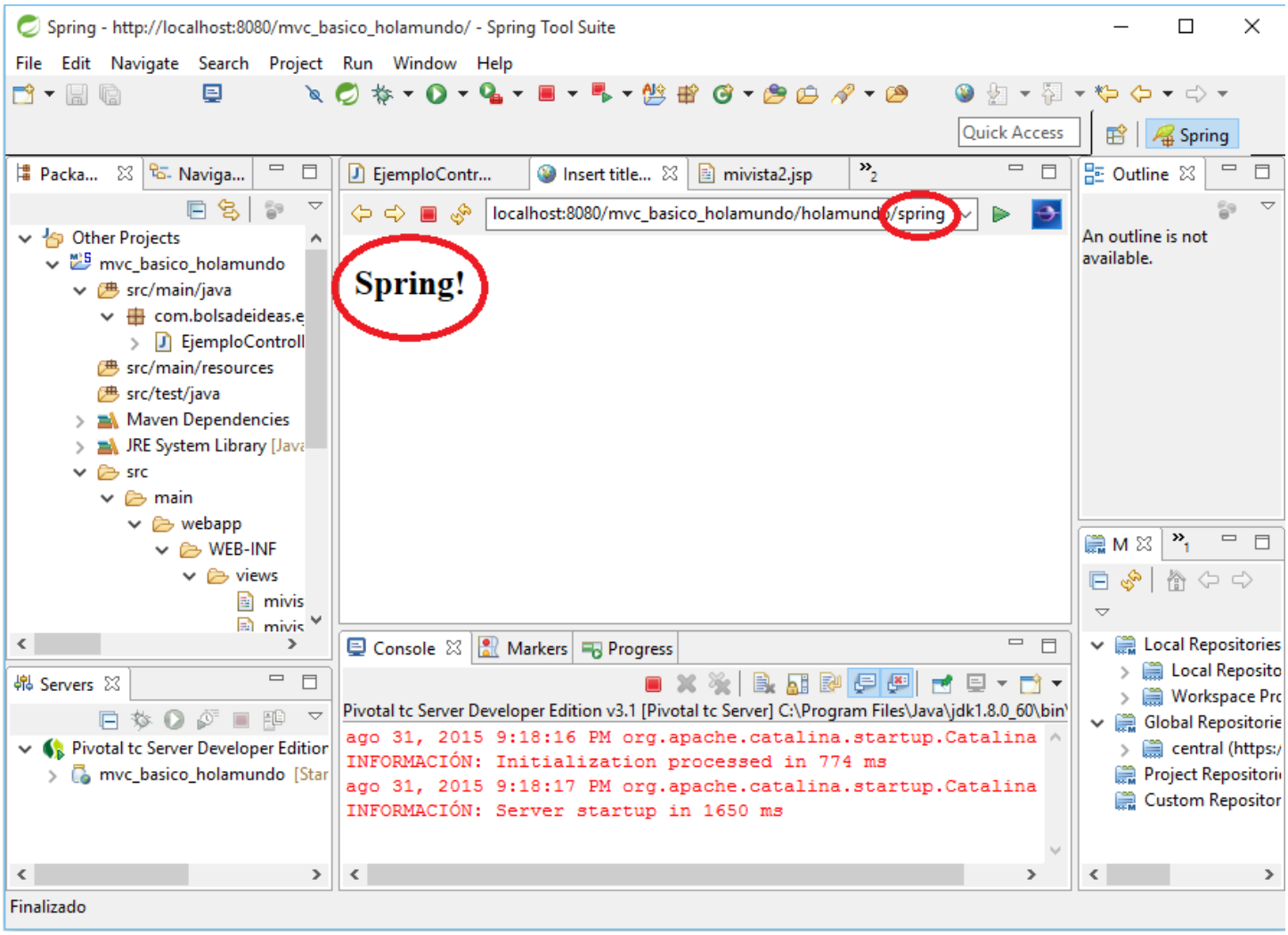
2. Creamos las vistas jsp de ambos métodos handlers:
    - Creamos **mivista2.jsp** bajo el directorio **src->main->webapp->WEB-INF->views**.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2>${mensaje2}</h2>
</body>
</html>
```

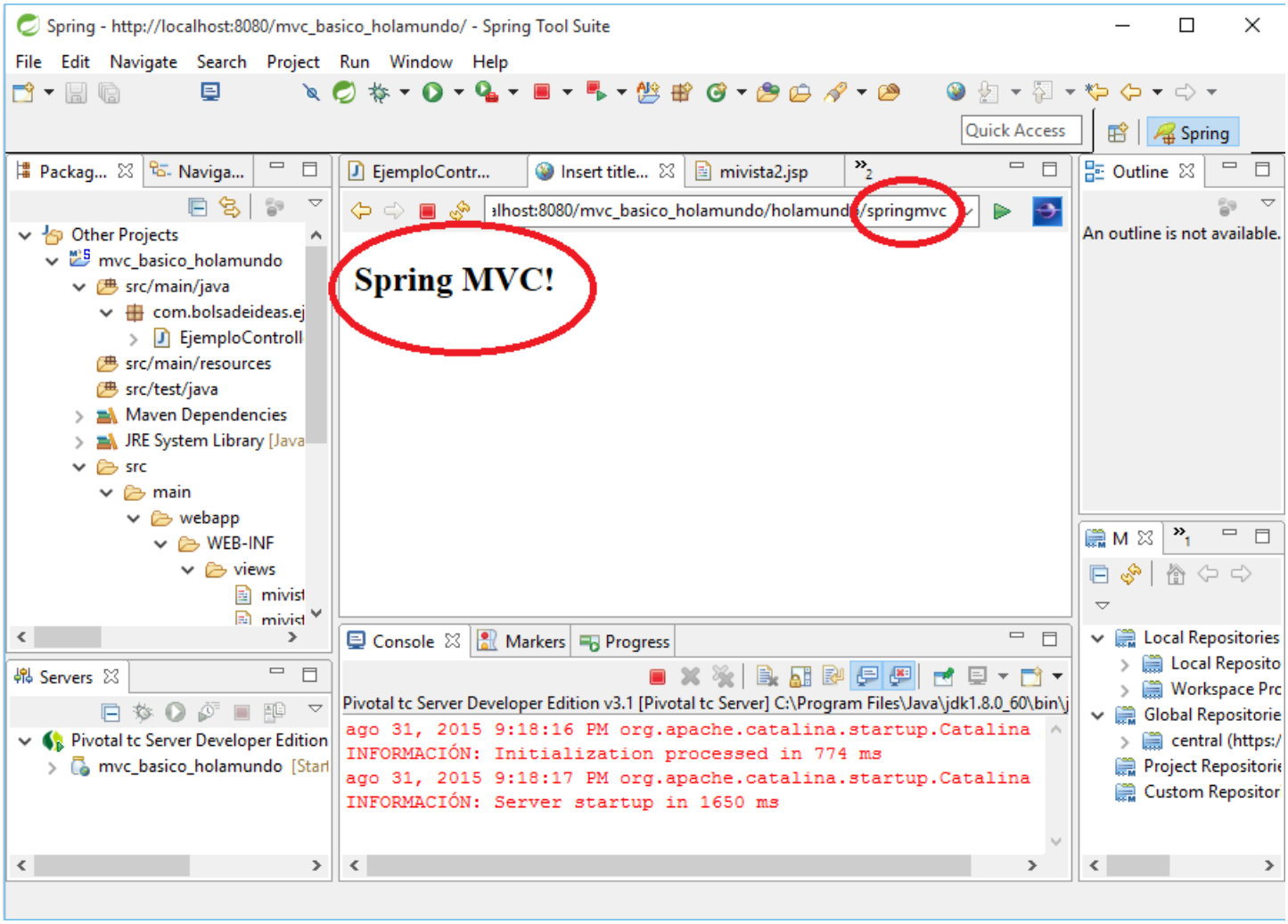
    - Creamos **mivista3.jsp** bajo el directorio **src->main->webapp->WEB-INF->views**.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2>${mensaje3}</h2>
</body>
</html>
```

3. Luego refrescamos en el navegador, para ver el resultado
    - Agregamos **/spring** al final de la URL y tecla enter.
    - Observe el resultado



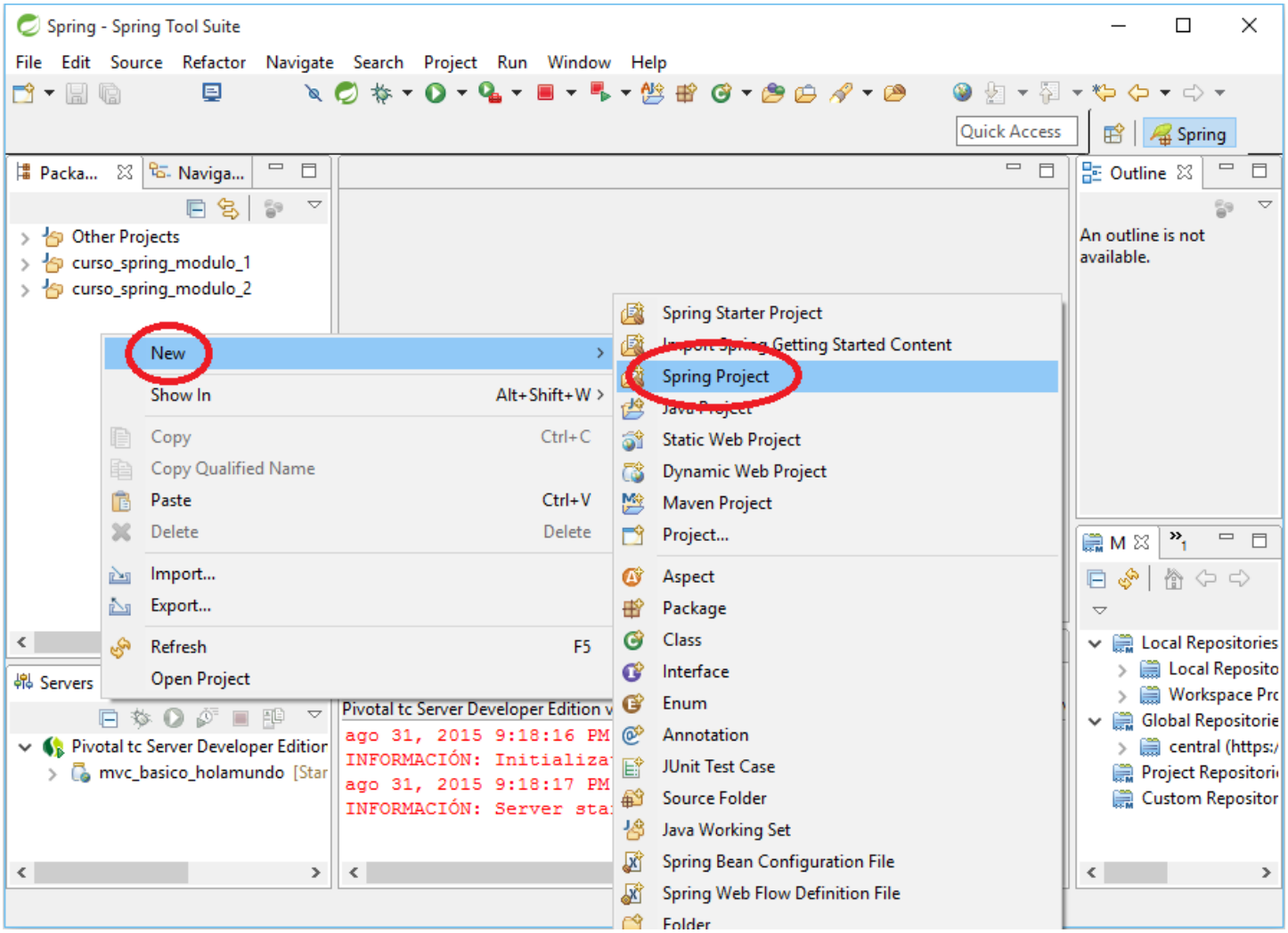
- Agregamos **/springmvc** al final de la URL y tecla enter.
- Observe el resultado



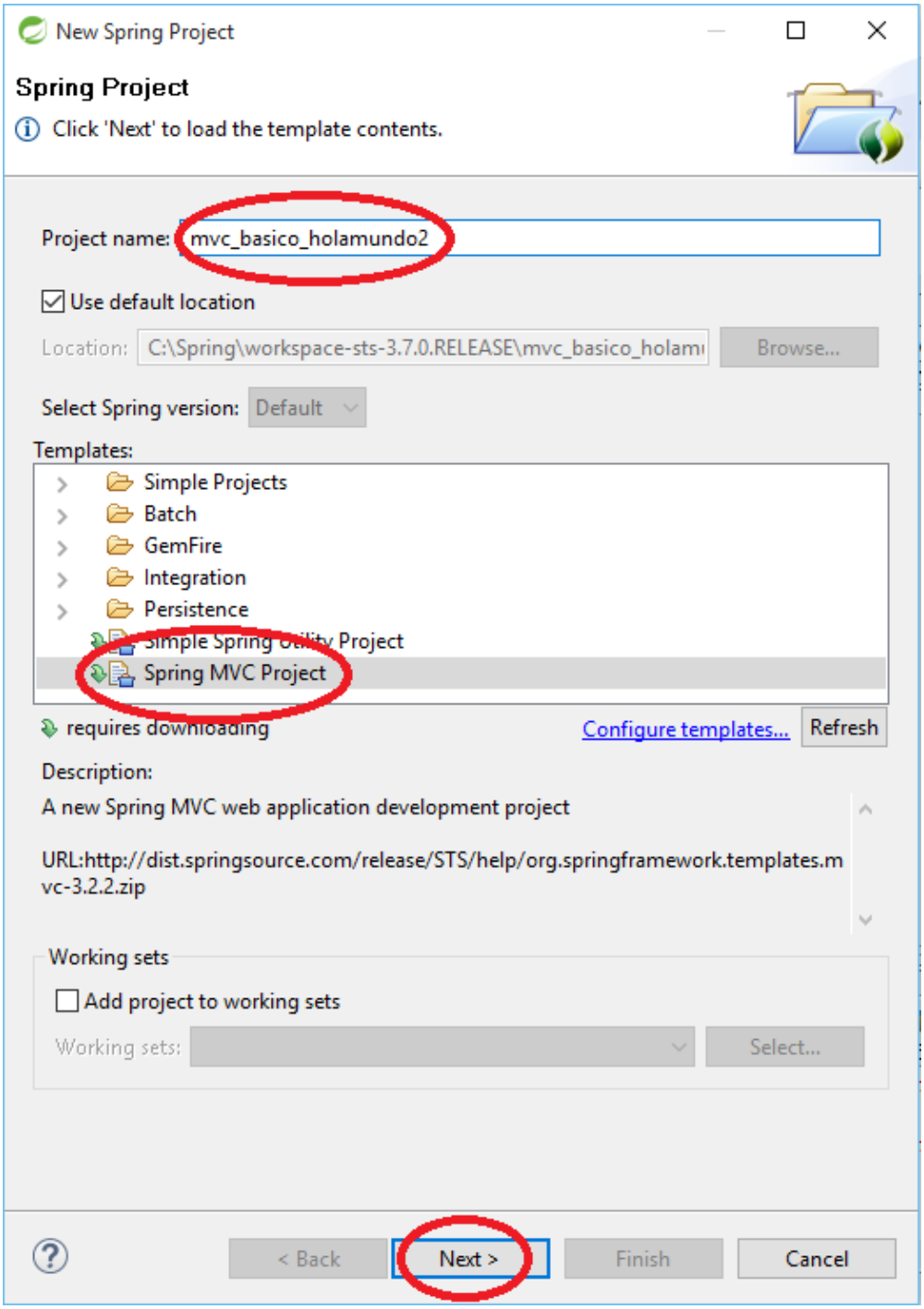
Ejercicio 3: Construir una aplicación Spring MVC "HolaMundo" usando la plantilla Spring MVC (desde STS)

Crearemos una aplicación Spring MVC desde las plantillas de proyectos de Spring que vienen lista para usar en el IDE STS. Las plantillas de proyectos de Spring nos provee la típica estructura de aplicación Spring MVC incluyendo los archivos de configuración de maven pom.xml y el de Spring, como también un controlador de ejemplo con un Hola Mundo.

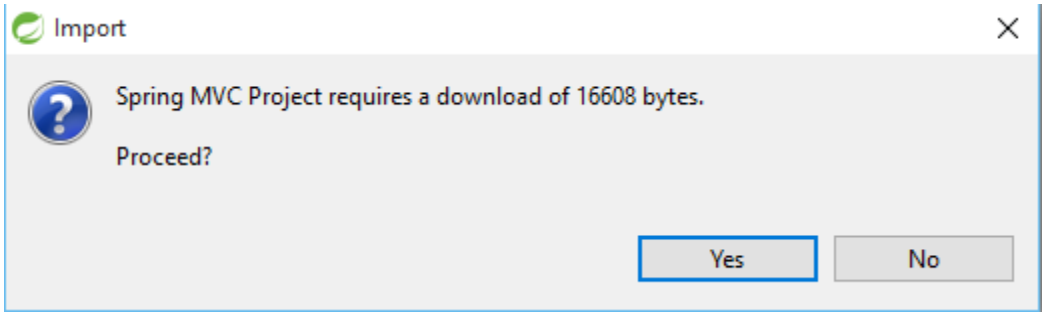
1. Crear un proyecto desde **Spring Template**.
2. Clic derecho dentro del Packages Explorer (Sección Izquierda): **New** -> **Spring Project.**



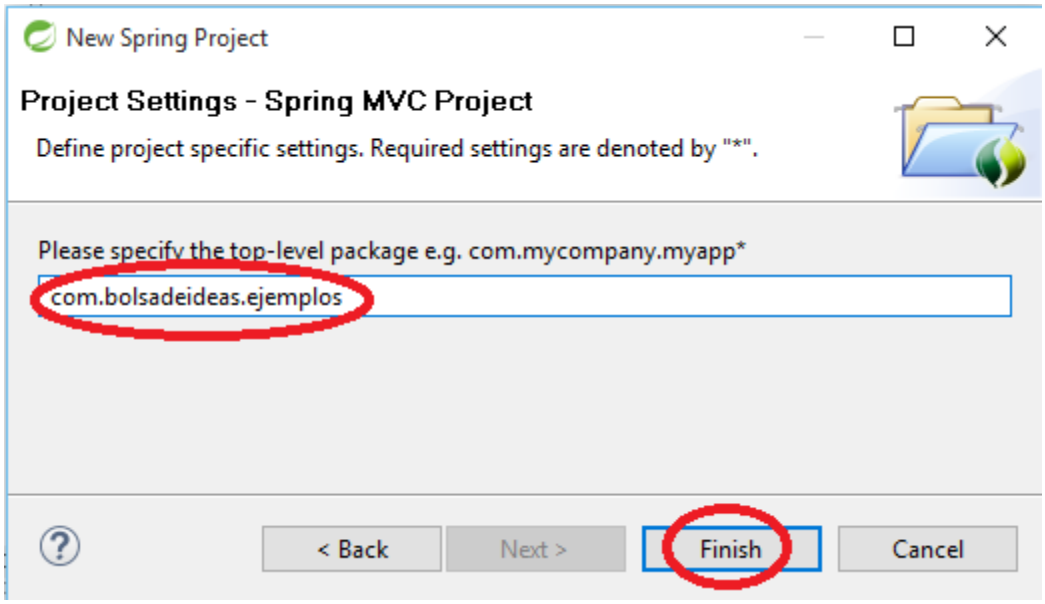
3. **Spring MVC Project**.
    - Para el campo **Project name**, ingresamos **mvc_basico_holamundo2**.
    - Seleccionamos **Spring MVC Project** y clic en **Next >**



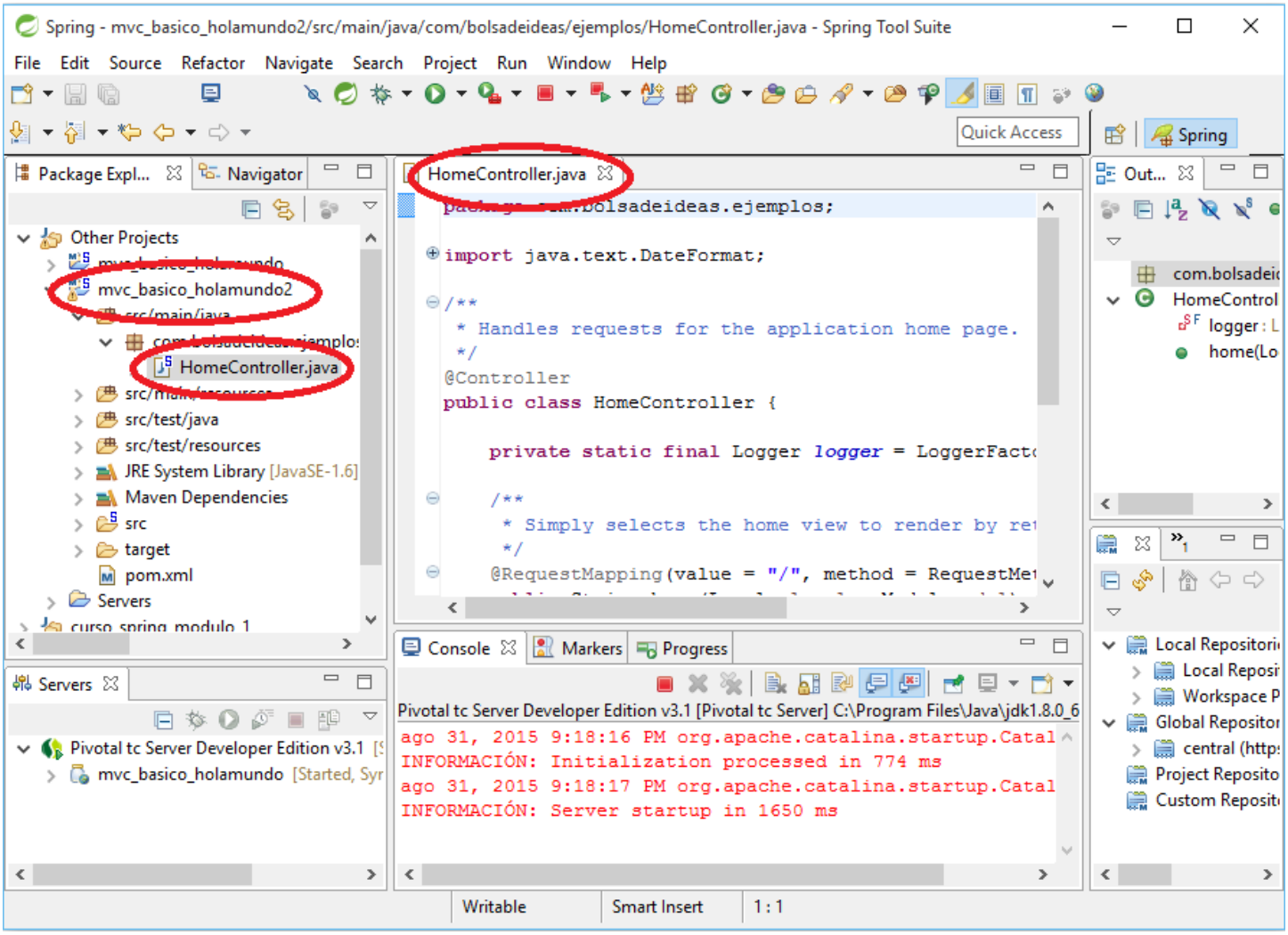
- Si esta es la primera vez que estamos creando un proyecto con la plantilla Spring MVC, entonces nos preguntará si deseamos descargar los archivos necesarios.
- Clic en Yes



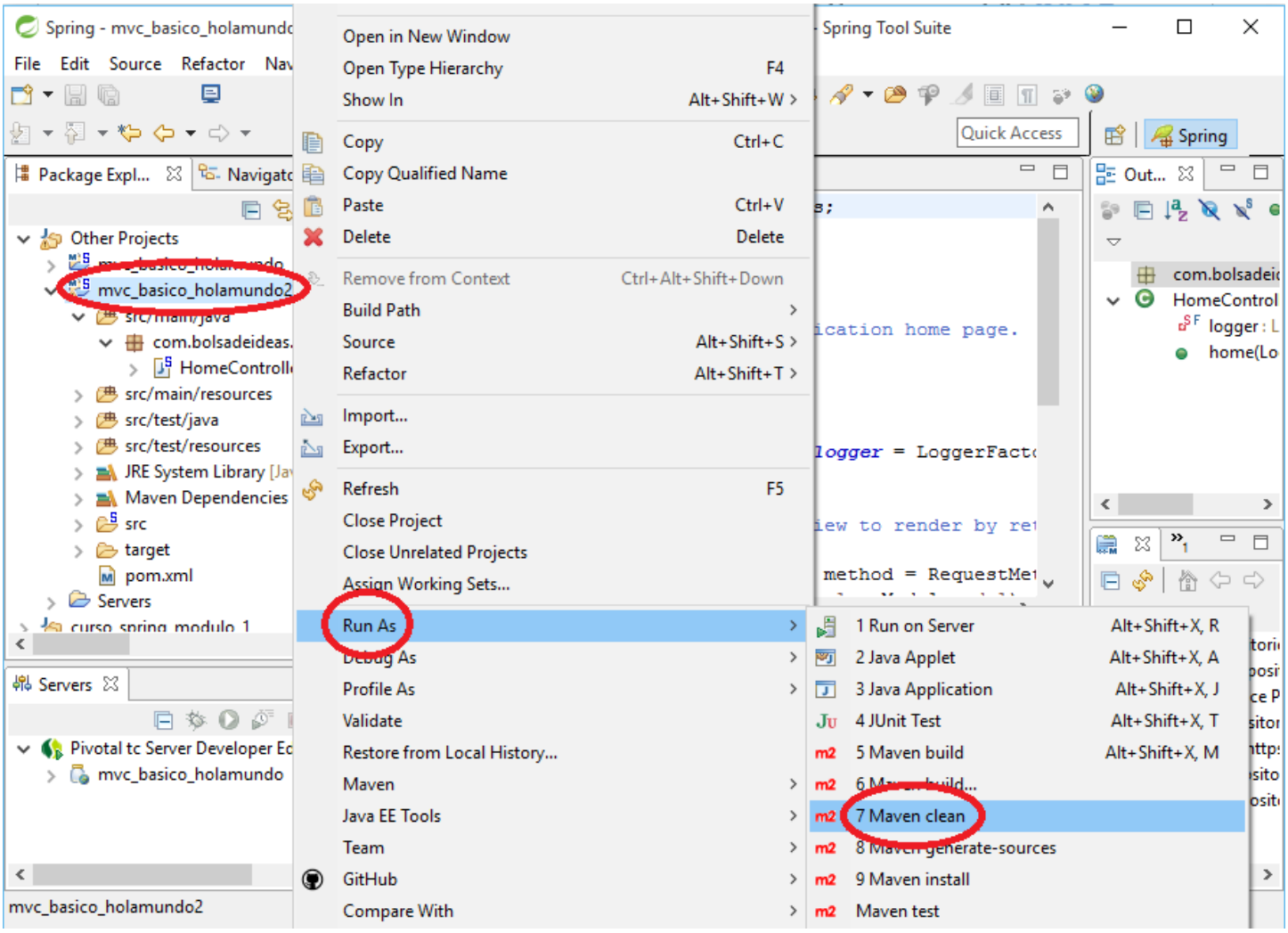
- Para el campo **top-level package** (el paquete base del proyecto), ingresamos **com.bolsadeideas.ejemplos**.
- Clic **Finish**.



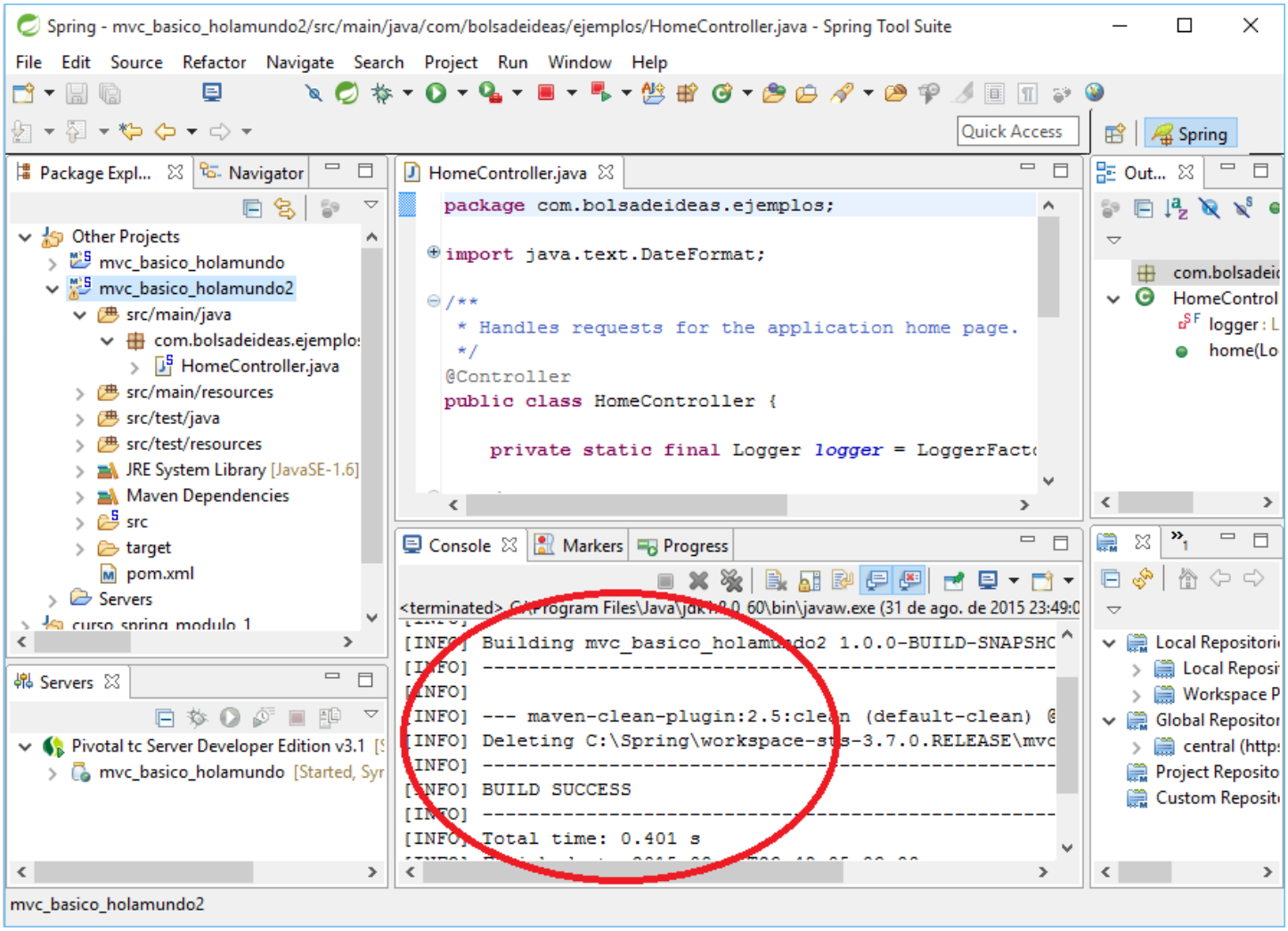
4. Observe que el IDE nos genera el proyecto complete incluyendo la clase controladora HomeController.java.



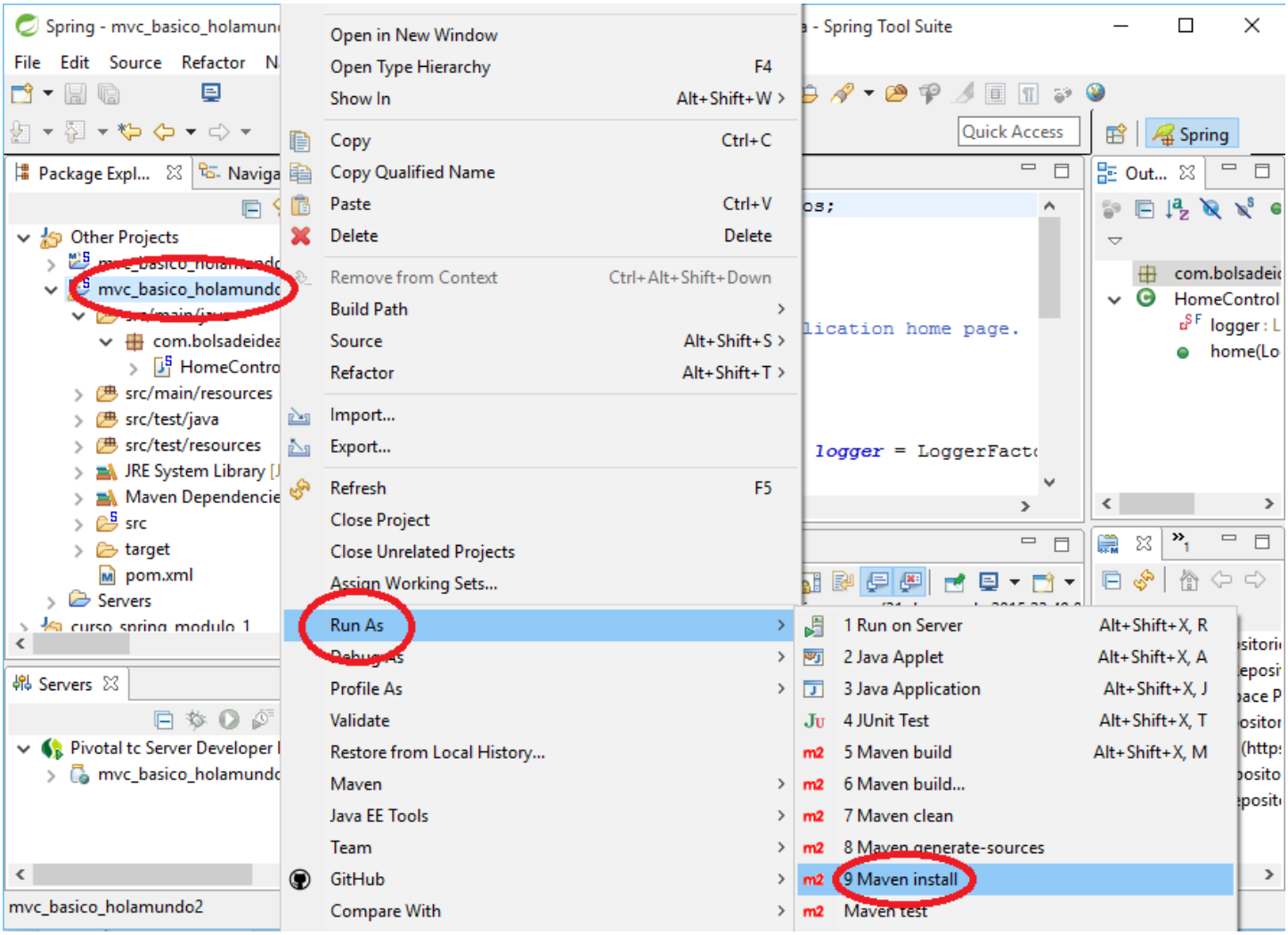
5. Ejecutamos la aplicación Spring MVC.
    - Clic derecho sobre el proyecto **mvc_basico_holamundo2->Run As**
    - Ejecutamos "**Maven clean**".



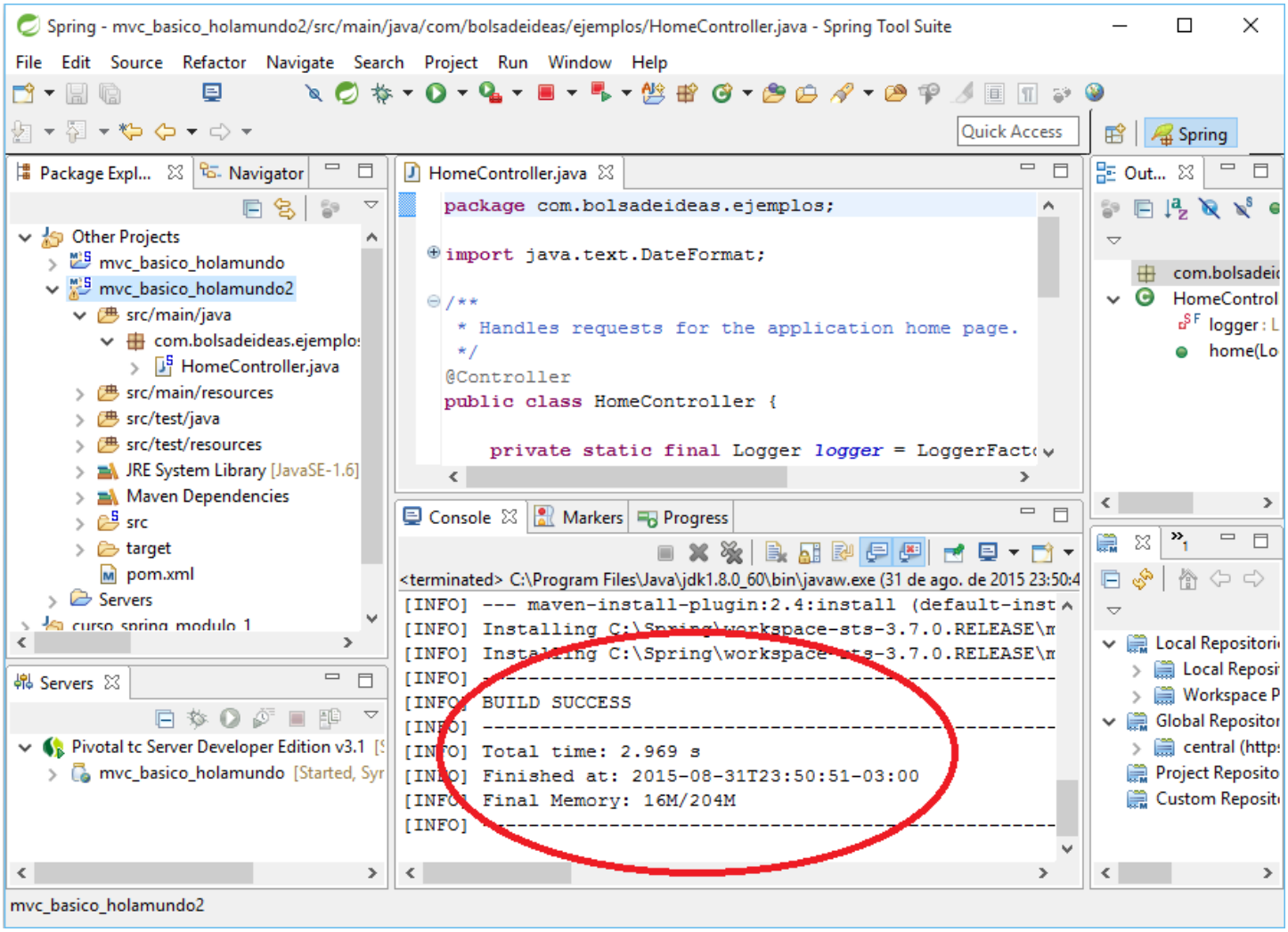
- Una vez finalizado el **Maven clean**, podemos ver el resultado en consola: **Pestaña Console**.



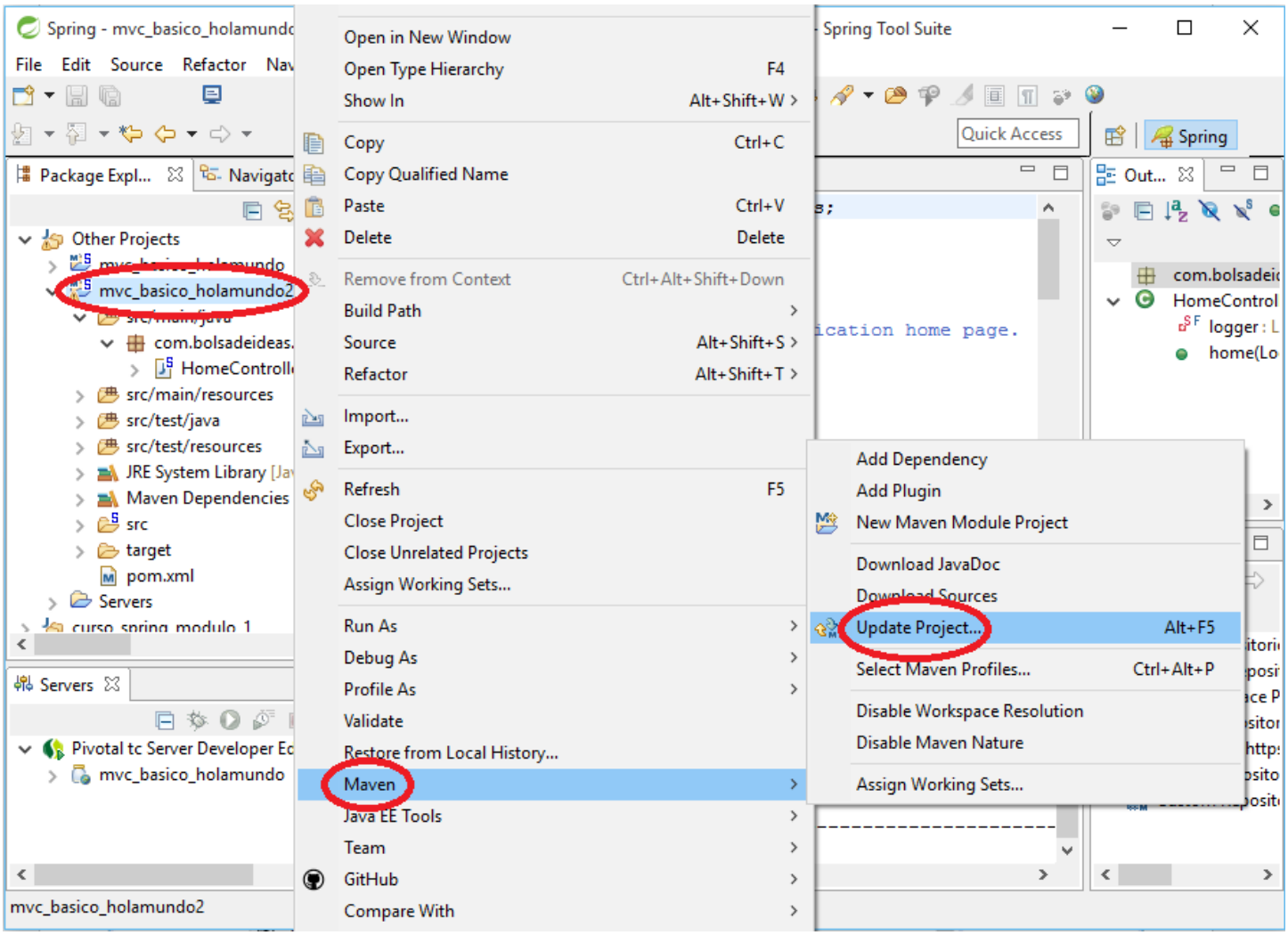
- Luego ejecutamos "**Maven install**" para generar el proyecto con sus dependencias manejadas por maven.



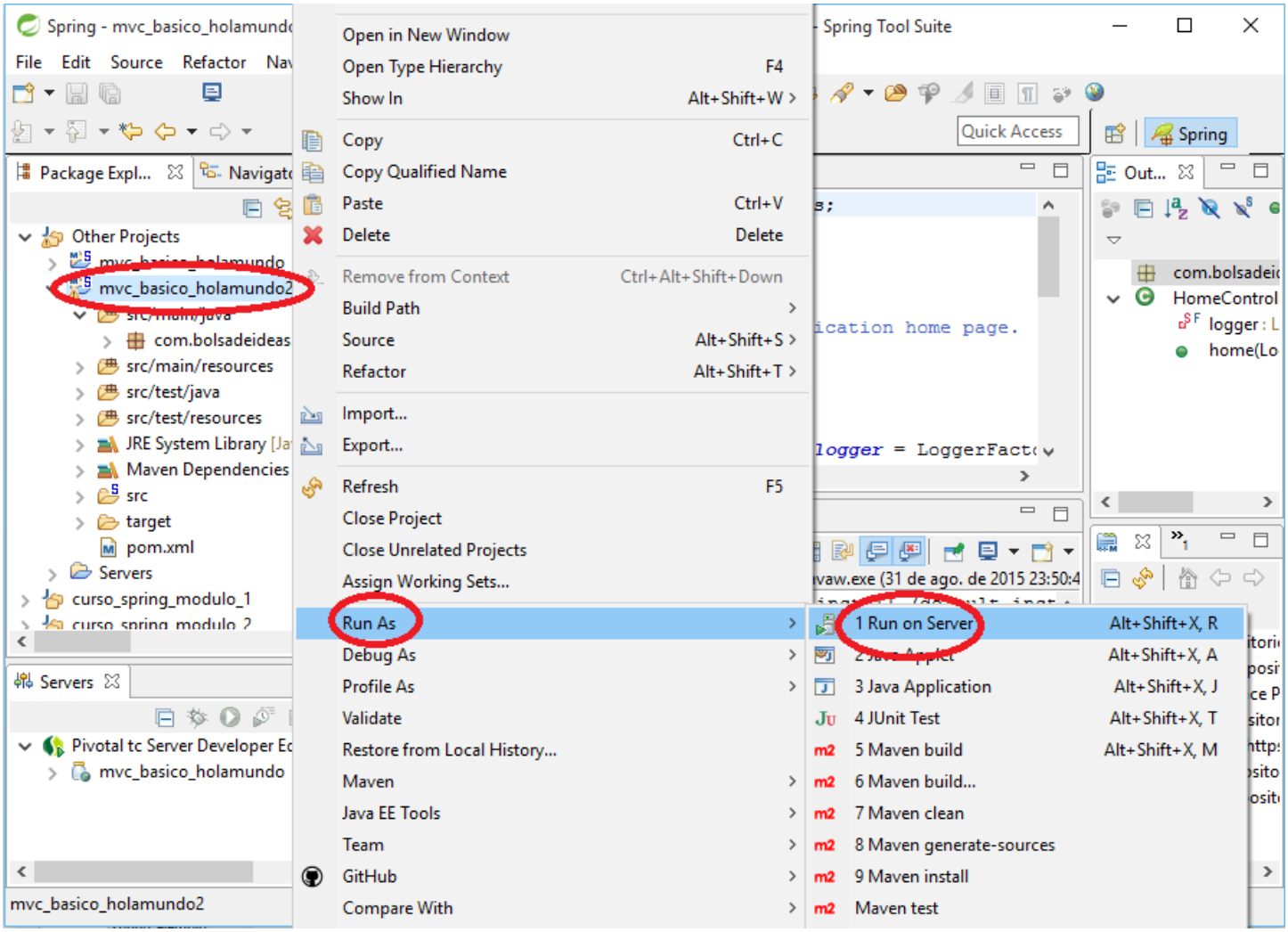
- Esperamos a que se haya construido y compilado el proyecto "**BUILD SUCCESS**", lo que demora un tiempo (la primera vez demora más ya que descarga las dependencias del repositorio), podemos ver el resultado en la pestaña Console de eclipse (sección inferior del IDE)



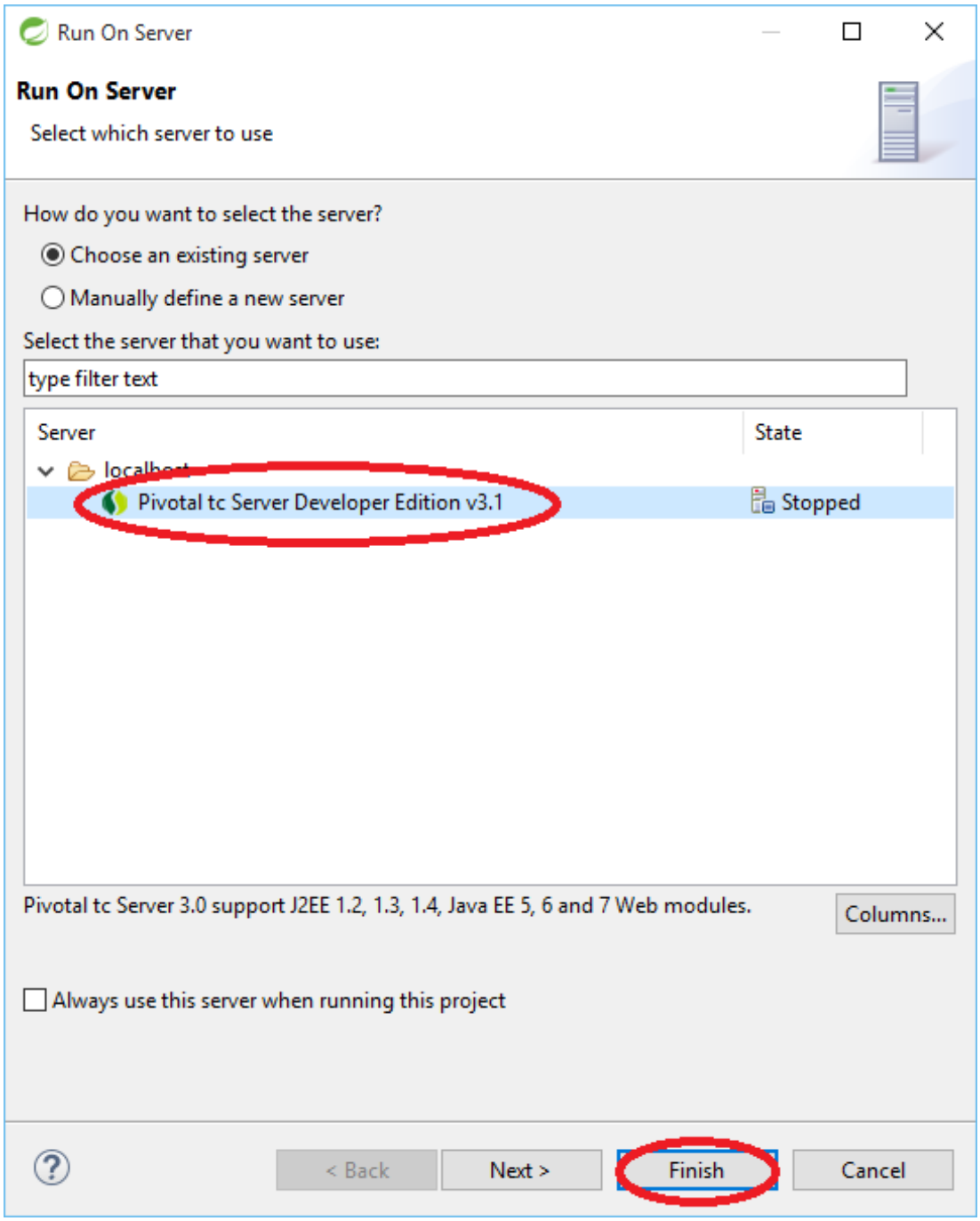
- Luego hacemos un **Maven->Update Project…**



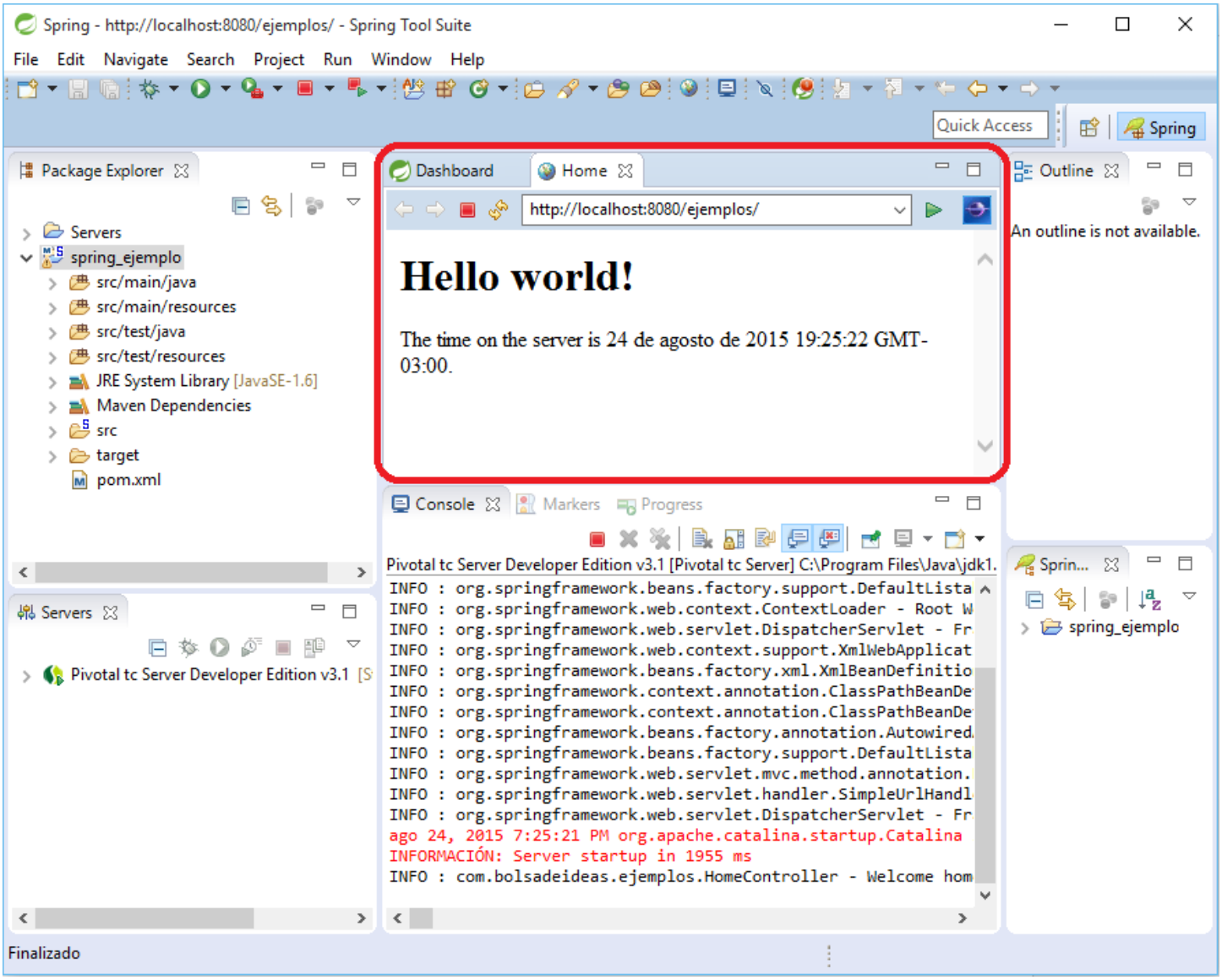
- Clic OK
- Ahora ejecutamos en el servidor
    i. Clic derecho sobre el proyecto "**mvc_basico_holamundo2**" => **Run As** => **Run on Server**.



- Seleccionamos el servidor.



- Observamos que aparece el resultado en el navegador Web interno de STS, cómo se muestra a continuación:



6. Estudiamos el proyecto generado
    - Abrir y estudiar el archivo **pom.xml**

```
<?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/maven-
v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.bolsadeideas</groupId>
    <artifactId>ejemplos</artifactId>
    <name>mvc_basico_holamundo2</name>
    <packaging>war</packaging>
    <version>1.0.0-BUILD-SNAPSHOT</version>
    <properties>
        <java-version>1.6</java-version>
        <org.springframework-version>3.1.1.RELEASE</org.springframework-version>
        <org.aspectj-version>1.6.10</org.aspectj-version>
        <org.slf4j-version>1.6.6</org.slf4j-version>
    </properties>
    <dependencies>
        <!-- Spring -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${org.springframework-version}</version>
            <exclusions>
                <!-- Exclude Commons Logging in favor of SLF4j -->
                <exclusion>
                    <groupId>commons-logging</groupId>
                    <artifactId>commons-logging</artifactId>
                 </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${org.springframework-version}</version>
        </dependency>

        <!-- AspectJ -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjrt</artifactId>
            <version>${org.aspectj-version}</version>
        </dependency>

        <!-- Logging -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
```

```
        <version>${org.slf4j-version}</version>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>jcl-over-slf4j</artifactId>
        <version>${org.slf4j-version}</version>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <version>${org.slf4j-version}</version>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.15</version>
        <exclusions>
            <exclusion>
                <groupId>javax.mail</groupId>
                <artifactId>mail</artifactId>
            </exclusion>
            <exclusion>
                <groupId>javax.jms</groupId>
                <artifactId>jms</artifactId>
            </exclusion>
            <exclusion>
                <groupId>com.sun.jdmk</groupId>
                <artifactId>jmxtools</artifactId>
            </exclusion>
            <exclusion>
                <groupId>com.sun.jmx</groupId>
                <artifactId>jmxri</artifactId>
            </exclusion>
        </exclusions>
        <scope>runtime</scope>
    </dependency>

    <!-- @Inject -->
    <dependency>
        <groupId>javax.inject</groupId>
        <artifactId>javax.inject</artifactId>
        <version>1</version>
    </dependency>

    <!-- Servlet -->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
        <version>2.5</version>
        <scope>provided</scope>
```

```
            </dependency>
            <dependency>
                <groupId>javax.servlet.jsp</groupId>
                <artifactId>jsp-api</artifactId>
                <version>2.1</version>
                <scope>provided</scope>
            </dependency>
            <dependency>
                <groupId>javax.servlet</groupId>
                <artifactId>jstl</artifactId>
                <version>1.2</version>
            </dependency>

            <!-- Test -->
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>4.7</version>
                <scope>test</scope>
            </dependency>
        </dependencies>
        <build>
            <plugins>
                <plugin>
                    <artifactId>maven-eclipse-plugin</artifactId>
                    <version>2.9</version>
                    <configuration>
                        <additionalProjectnatures>

    <projectnature>org.springframework.ide.eclipse.core.springnature</projectnature>
                        </additionalProjectnatures>
                        <additionalBuildcommands>

    <buildcommand>org.springframework.ide.eclipse.core.springbuilder</buildcommand>
                        </additionalBuildcommands>
                        <downloadSources>true</downloadSources>
                        <downloadJavadocs>true</downloadJavadocs>
                    </configuration>
                </plugin>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>2.5.1</version>
                    <configuration>
                        <source>1.6</source>
                        <target>1.6</target>
                        <compilerArgument>-Xlint:all</compilerArgument>
                        <showWarnings>true</showWarnings>
                        <showDeprecation>true</showDeprecation>
                    </configuration>
                </plugin>
                <plugin>
```

```
                    <groupId>org.codehaus.mojo</groupId>
                    <artifactId>exec-maven-plugin</artifactId>
                    <version>1.2.1</version>
                    <configuration>
                        <mainClass>org.test.int1.Main</mainClass>
                    </configuration>
                </plugin>
            </plugins>
        </build>
    </project>
```

- Abrir y estudiar el archivo: **/src/main/webapp/WEB-INF/web.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd">

	<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
	<context-param>
		<param-name>contextConfigLocation</param-name>
		<param-value>/WEB-INF/spring/root-context.xml</param-value>
	</context-param>

	<!-- Creates the Spring Container shared by all Servlets and Filters -->
	<listener>
		<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
	</listener>

	<!-- Processes application requests -->
	<servlet>
		<servlet-name>appServlet</servlet-name>
		<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
		<init-param>
			<param-name>contextConfigLocation</param-name>
			<param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
		</init-param>
		<load-on-startup>1</load-on-startup>
	</servlet>

	<servlet-mapping>
		<servlet-name>appServlet</servlet-name>
		<url-pattern>/</url-pattern>
	</servlet-mapping>

</web-app>
```

- Abrir y estudiar el archivo Spring root context **/src/main/webapp/WEB-INF/spring/root-context.xml**. En este ejemplo no tenemos ningún beans.

```
<?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-3.0.xsd">

    <!-- Root Context: defines shared resources visible to all other web components -->

</beans>
```

- Abrir y estudiar el archivo Spring MVC context **/src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:beans="http://www.springframework.org/schema/beans"
        xmlns:context="http://www.springframework.org/schema/context"
        xsi:schemaLocation="http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd
                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">

        <!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->

        <!-- Enables the Spring MVC @Controller programming model -->
        <annotation-driven />

        <!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources in the
${webappRoot}/resources directory -->
        <resources mapping="/resources/**" location="/resources/" />

        <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views
directory -->
        <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
                <beans:property name="prefix" value="/WEB-INF/views/" />
                <beans:property name="suffix" value=".jsp" />
        </beans:bean>

        <context:component-scan base-package="com.bolsadeideas.ejemplos" />


</beans:beans>
```

- Abrir y estudiar la clase controladora **/src/main/java/com.bolsadeideas.ejemplos/HomeController.java**

```
package com.bolsadeideas.ejemplos;

import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

/**
 * Handles requests for the application home page.
 */
@Controller
public class HomeController {

	private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

	/**
	 * Simply selects the home view to render by returning its name.
	 */
	@RequestMapping(value = "/", method = RequestMethod.GET)
	public String home(Locale locale, Model model) {
		logger.info("Welcome home! the client locale is "+ locale.toString());

		Date date = new Date();
		DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, locale);

		String formattedDate = dateFormat.format(date);

		model.addAttribute("serverTime", formattedDate );

		return "home";
	}

}
```

- Abrir y estudiar el archivo vista **/src/main/webapp/WEB-INF/views/home.jsp**

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" %>
<html>
<head>
	<title>Home</title>
</head>
<body>
<h1>
	Hello world!
</h1>
<P>  The time on the server is ${serverTime}. </P>
</body>
</html>
```

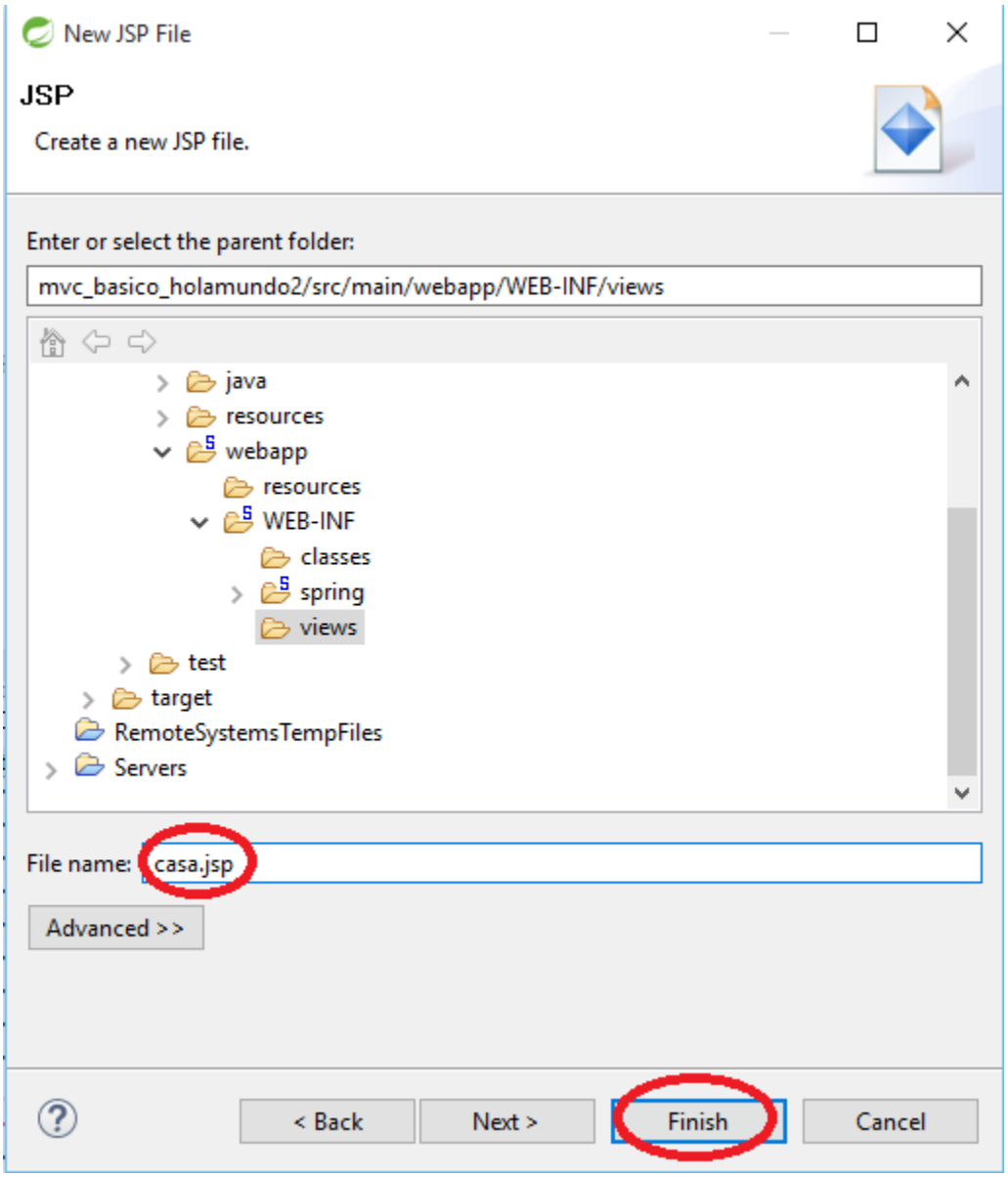
AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

7. Modificar el controller HomeController

    - Agregamos un nuevo método handler

```
package com.bolsadeideas.ejemplos;

import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

/**
 * Handles requests for the application home page.
 */
@Controller
public class HomeController {

	private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

	/**
	 * Simply selects the home view to render by returning its name.
	 */
	@RequestMapping(value = "/", method = RequestMethod.GET)
	public String home(Locale locale, Model model) {
		logger.info("Welcome home! the client locale is "+ locale.toString());

		Date date = new Date();
		DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG,
DateFormat.LONG, locale);

		String formattedDate = dateFormat.format(date);

		model.addAttribute("serverTime", formattedDate );

		return "home";
	}

	 @RequestMapping(value="/bienvenidos", method=RequestMethod.GET)
	 public String holaCasa(ModelMap modelMap) {
	     logger.info("Bienvenidos a mi casa!");
	     modelMap.addAttribute("color", "Piedra");
	     modelMap.addAttribute("jardin", 1000);
	     return "casa";
	 }
}
```

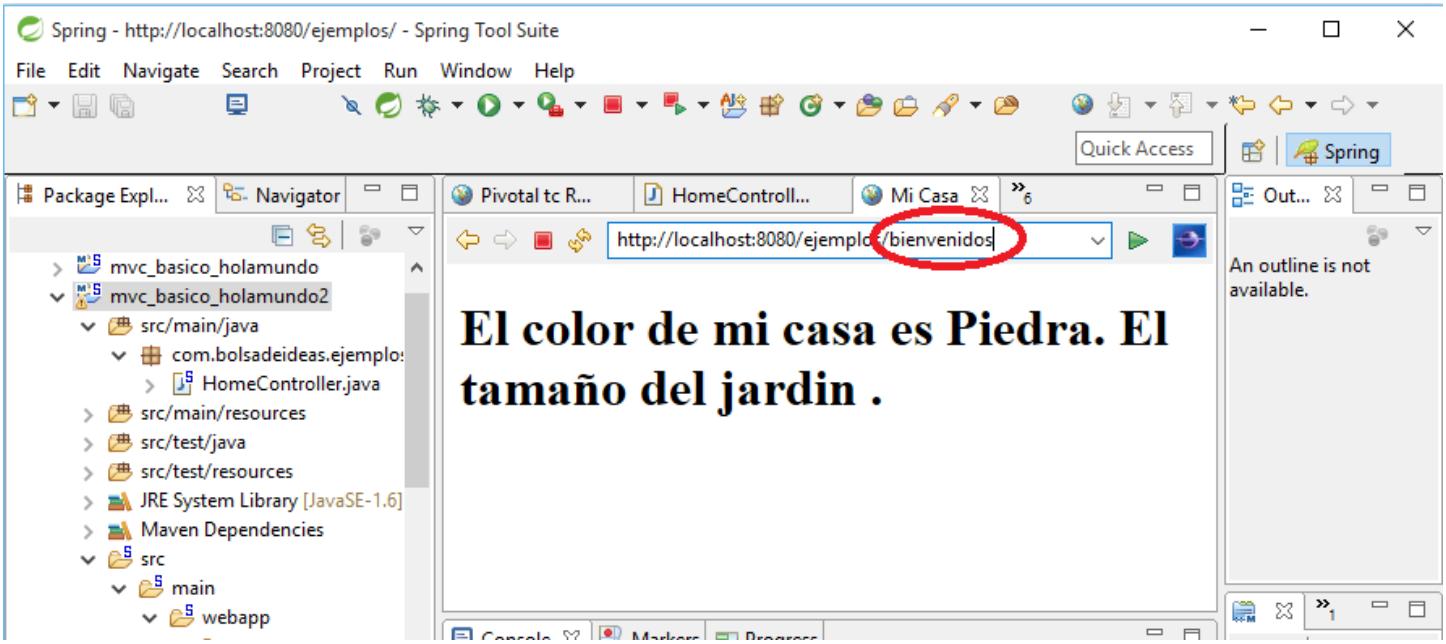
- Creamos la vista casa.jsp bajo el directorio **src->main->webapp->WEB-INF->views**



- Modificamos la vista casa.jsp con:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Mi Casa</title>
</head>
<body>
<h1>
El color de mi casa es ${color}.
El tamaño del jardin ${size}.
</h1>
</body>
</html>
```

4. Luego refrescamos en el navegador, para ver el resultado
   - Agregamos **/bienvenidos** al final de la URL y tecla enter.
   - Observe el resultado



Resumen

En este workshop de laboratorio, hemos aprendido a utilizar e implementar las funciones básicas de Spring Framework Web MVC, mediante diversas formas y técnicas, destacando entre ellas los controladores implementando con anotaciones siempre utilizando STS como IDE.

En el próximo módulo veremos las funciones más avanzadas de Spring MVC, como URI Template, Interceptores, objetos model, recurso, view controller etc.

Envía tus consultas a los foros!

Aquí es cuando debes sacarte todas las dudas haciendo consultas en los foros correspondientes

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

Lectura Recomendada y Bibliografía

- Web MVC framework, recomendable leer está sección del manual oficial para complementar con este workshop.
- Overview of the Spring Web Stack video presentado por Keith Donald.
- What's new in Spring por Arjen Poutsma.
- Spring framework, the next generation por Juergen Hoeller