

---

# **CREACION E IMPORTACIÓN DE PROYECTOS GRADLE Y MAVEN EN ECLIPSE**

**EDUARD LARA**

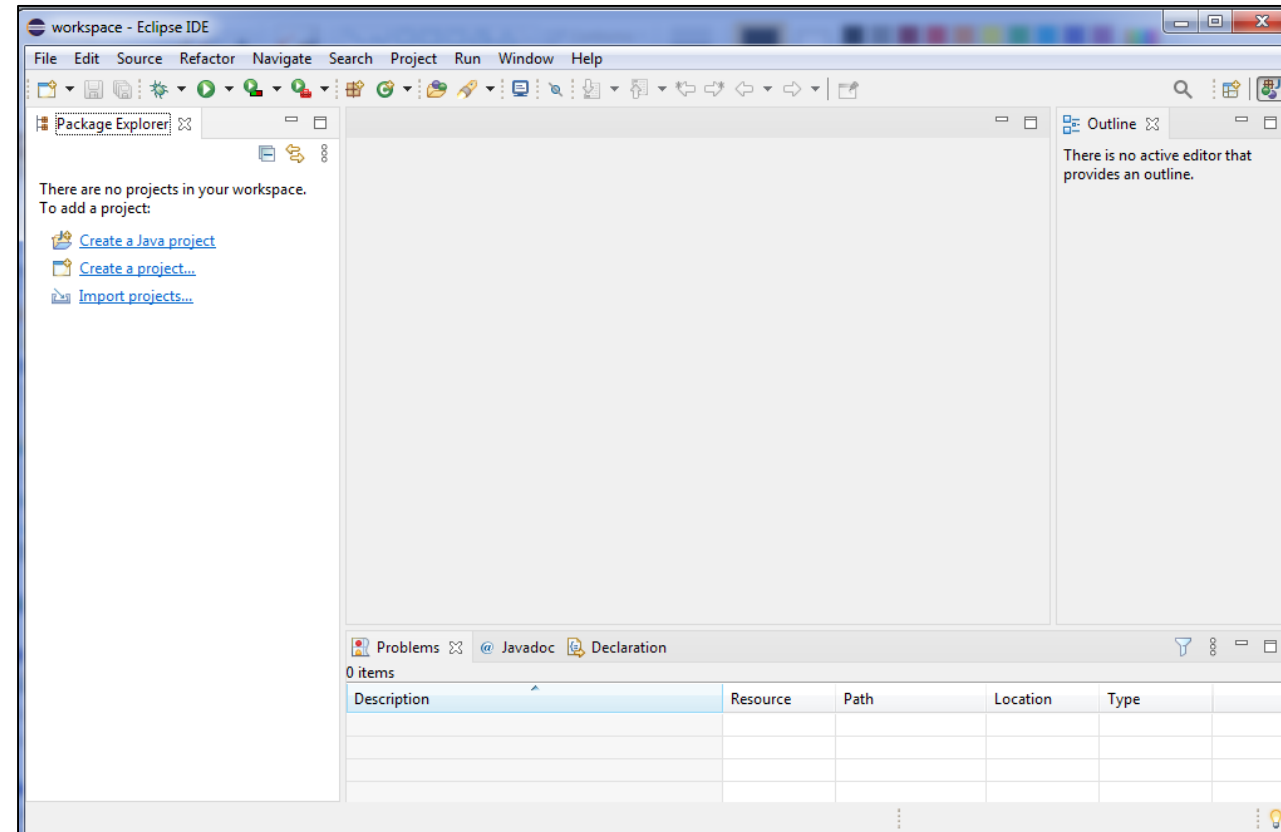
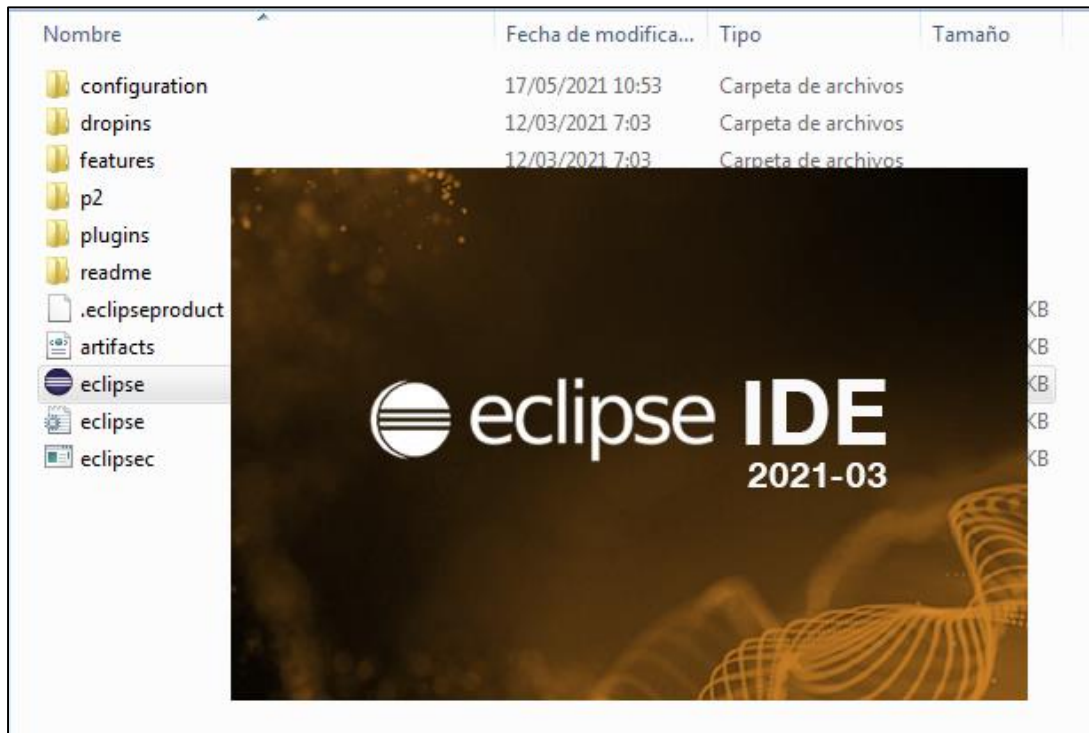
# INDICE

---

1. Instalación Plugin Spring Eclipse
2. Eclipse SpringToolSuite
3. Creación proyecto Spring
4. Importar desde spring.io
5. Agregar dependencias una vez creado el proyecto

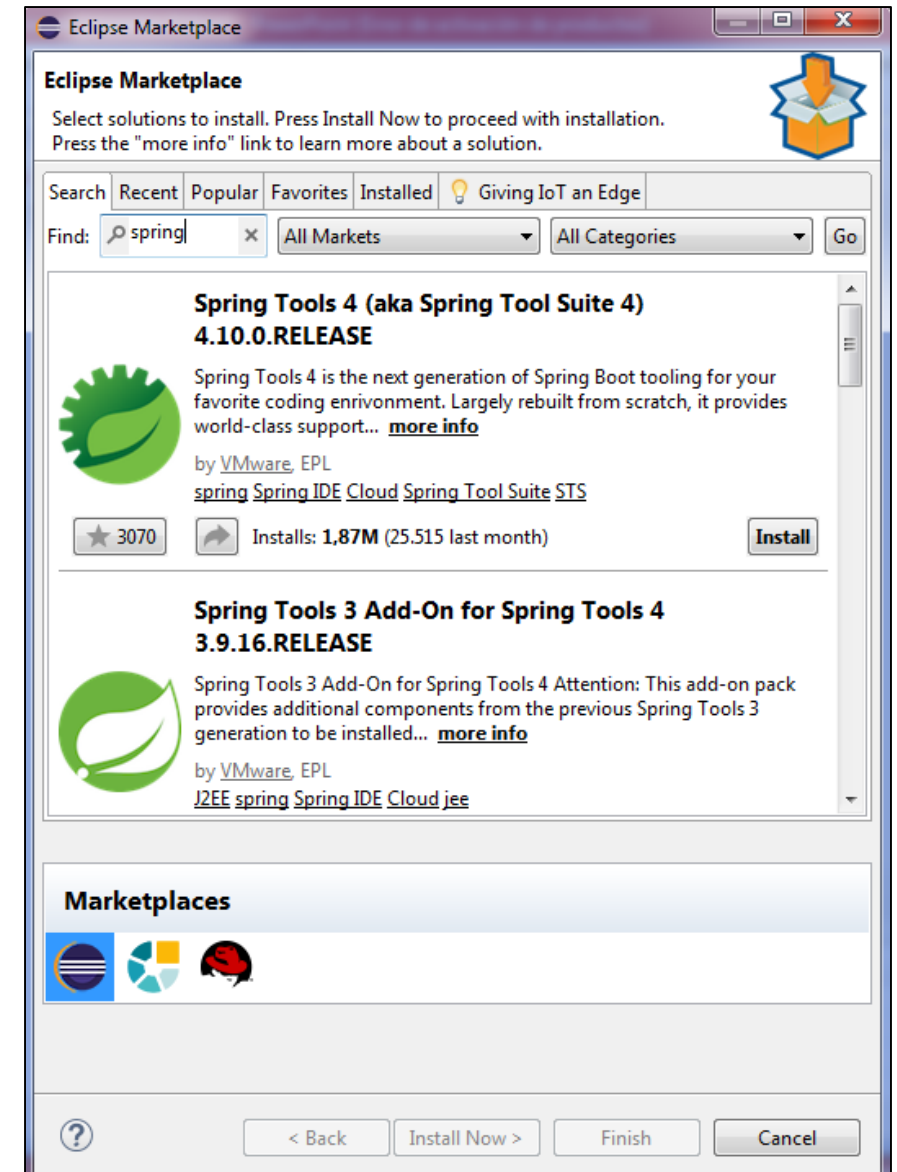
# 1. PLUGIN SPRING ECLIPSE

**Paso 1)** Para desarrollar con el Framework Spring, inicialmente en Eclipse debemos instalar el plugin de Spring, puesto que por defecto no viene instalado. Podemos arrancar nuestra versión de eclipse habitual, con la que hemos ido desarrollando, o bajarnos otra versión de <https://www.eclipse.org/downloads/packages/>:



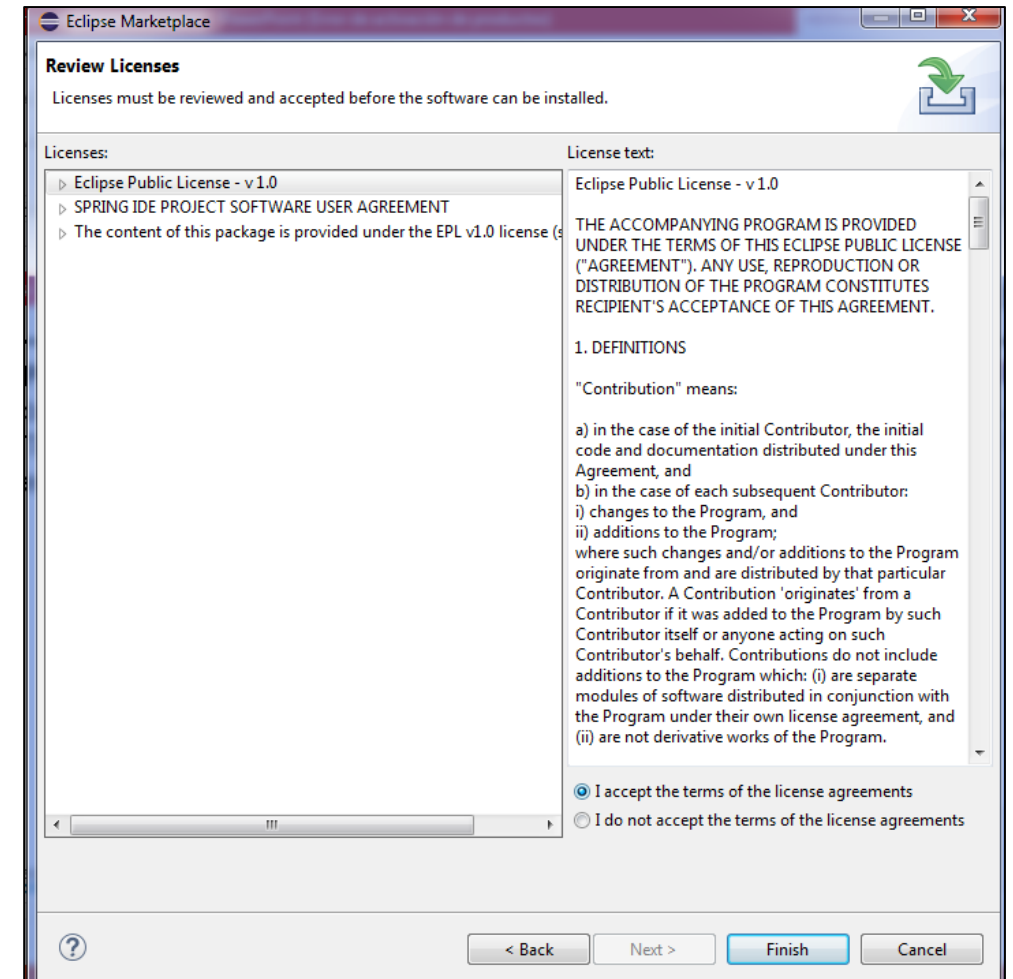
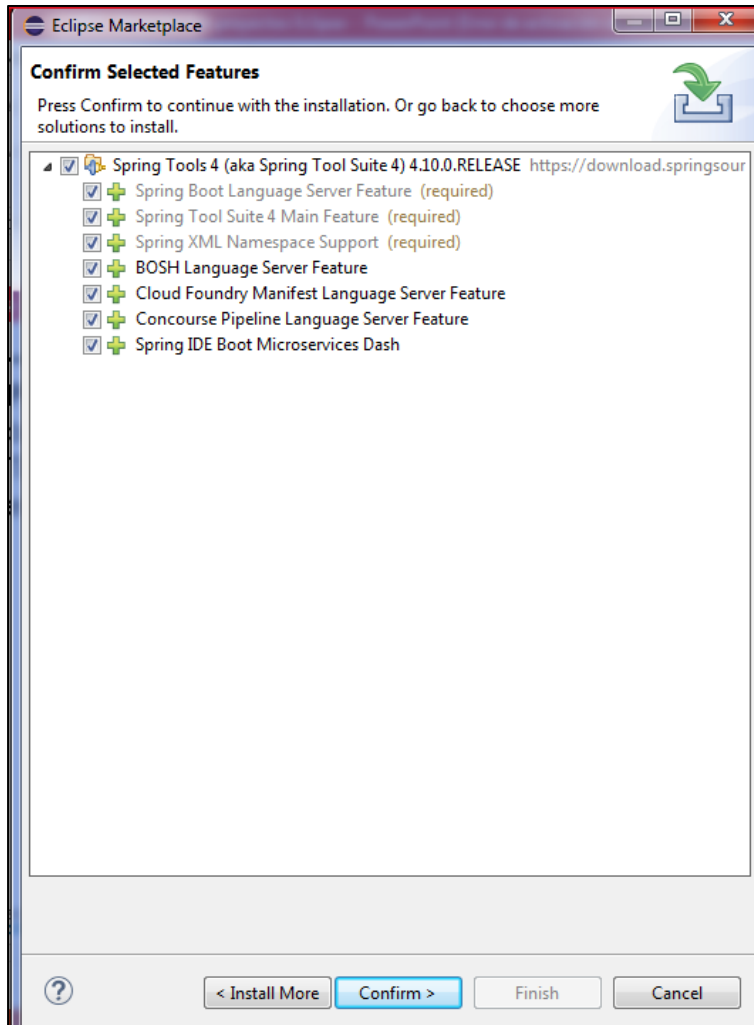
# 1. PLUGIN SPRING ECLIPSE

**Paso 2)** Vamos a Help/Eclipse MarketPlace y buscamos el término Spring. Vemos que el plugin Spring Tools 4 no se encuentra instalado. Hacemos click en Install:



# 1. PLUGIN SPRING ECLIPSE

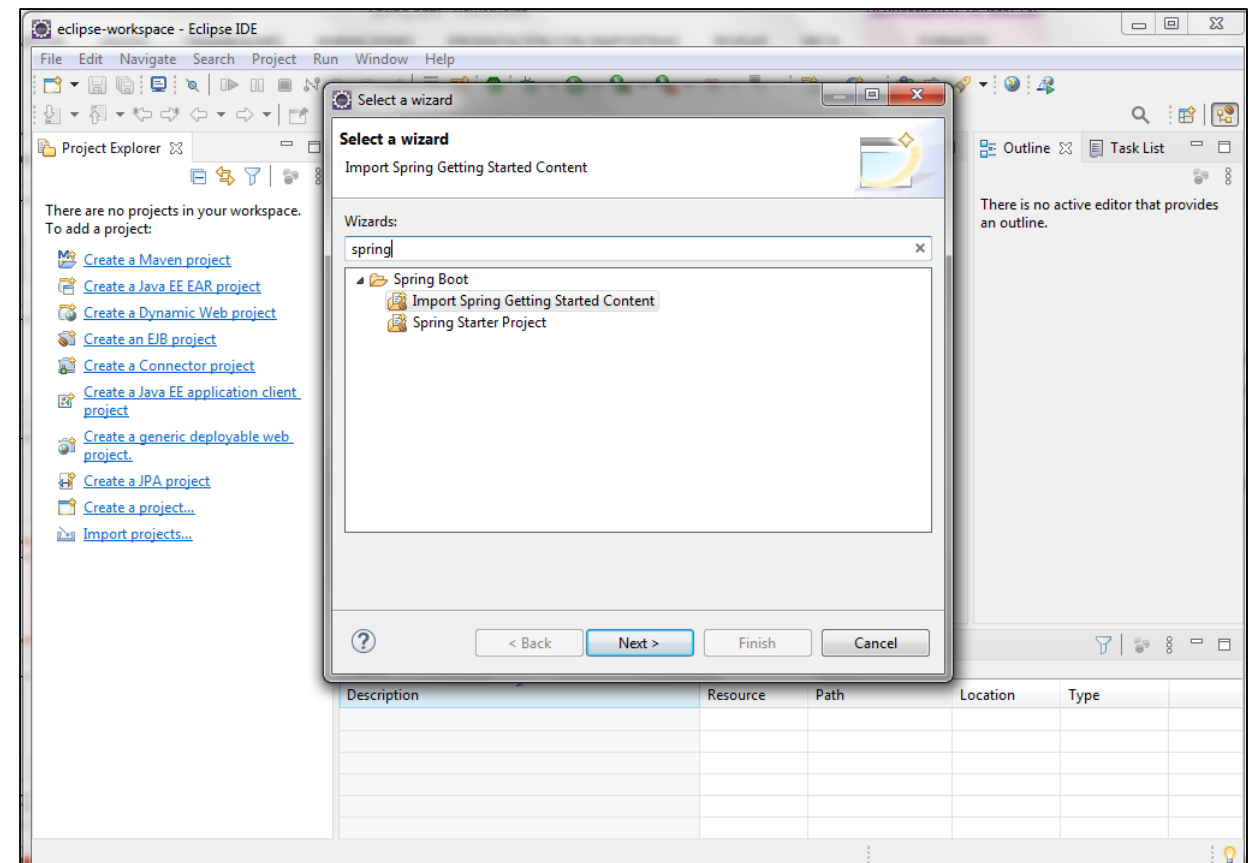
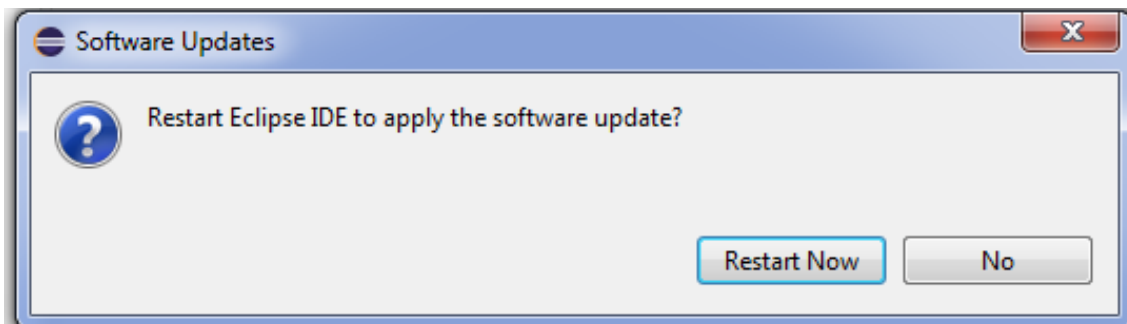
**Paso 3)** Confirmamos todo el paquete y en la siguiente pantalla aceptamos los términos de la licencia y hacemos click en Finish:



# 1. PLUGIN SPRING ECLIPSE

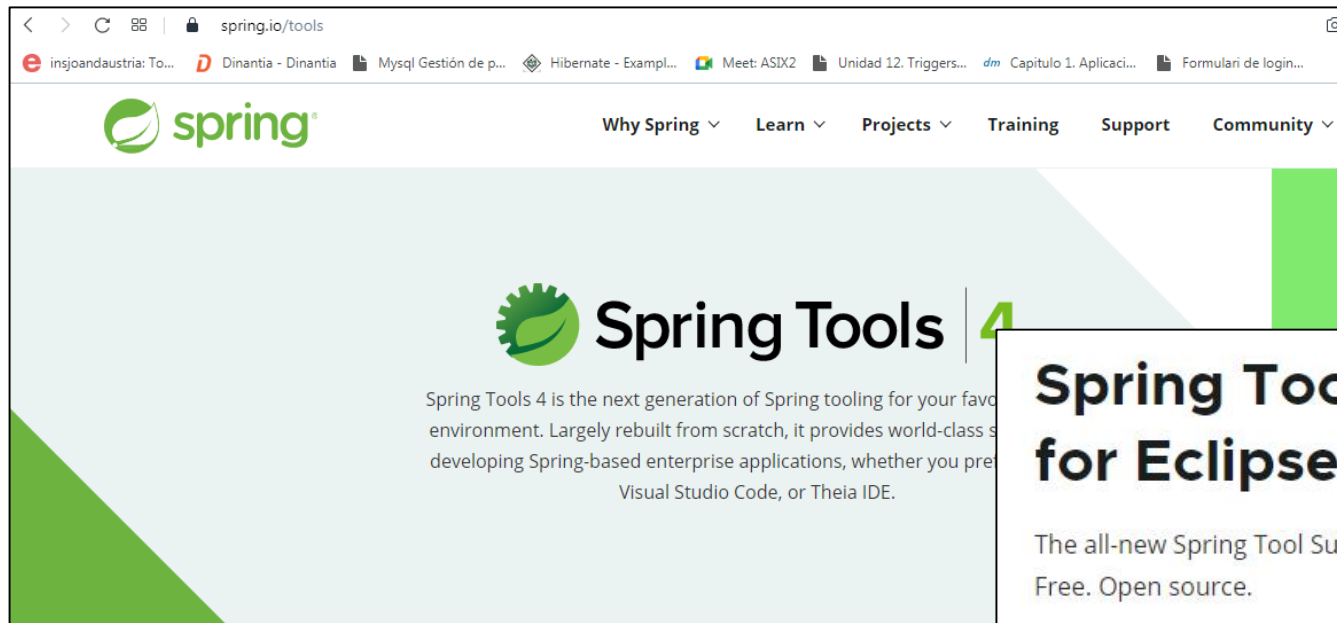
**Paso 4)** Empieza la instalación del plugin, que puede demorar un poco. Abajo a la derecha aparece la barra de proceso. Finalmente eclipse nos pide reiniciar y en el arranque vemos que ya podemos crear proyectos Spring Boot:

Resource	Path	Location	Type	
Installing Software: (48%)				



## 2. SPRINGTOOLSUITE

**Paso 1)** SpringToolSuite es una segunda opción para trabajar con Spring. Se trata de un eclipse preconfigurado con Spring, que se puede descargar desde la página [spring.io/tools](http://spring.io/tools), previa elección de la versión idónea para nuestro sistema operativo



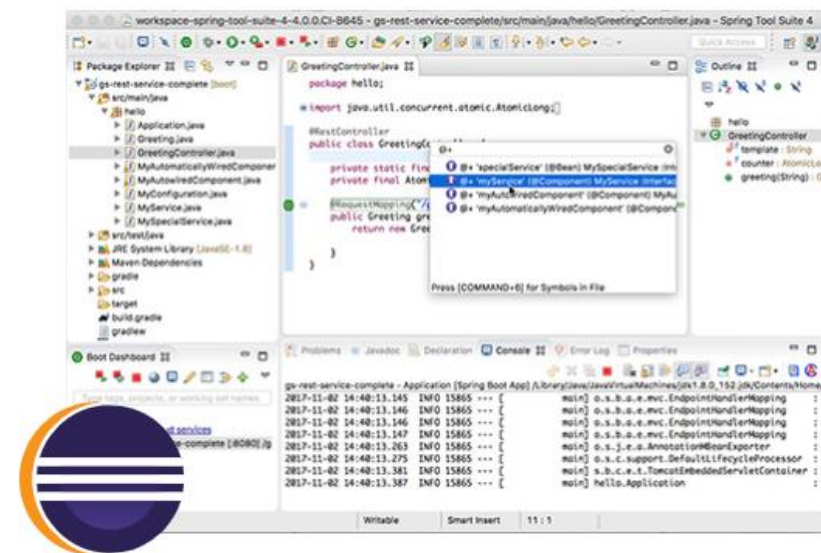
### Spring Tools 4 for Eclipse

The all-new Spring Tool Suite 4.  
Free. Open source.

4.9.0 - LINUX 64-BIT

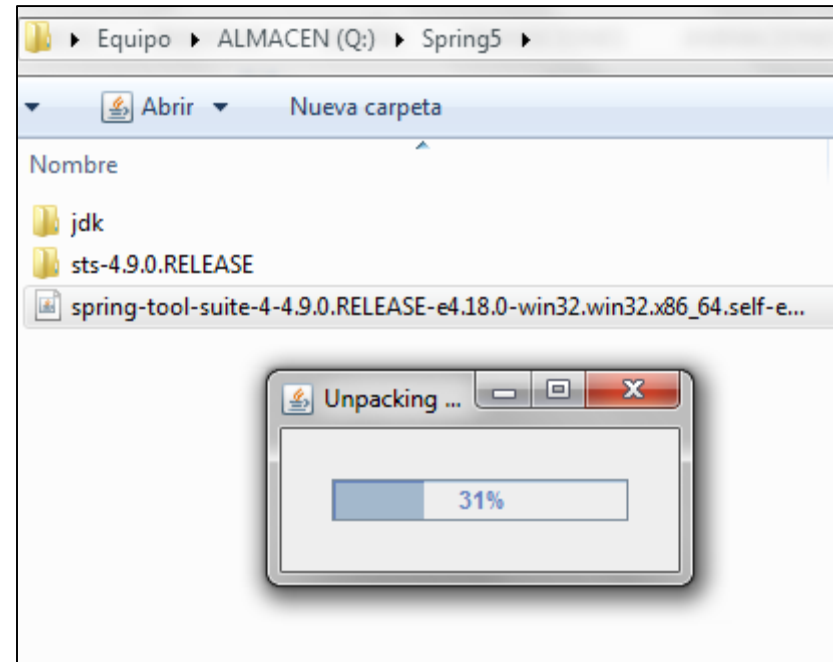
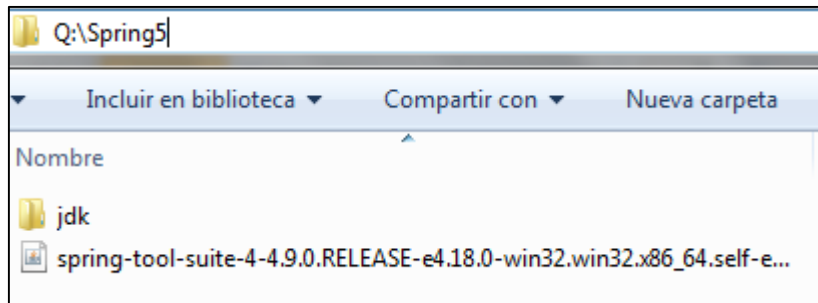
4.9.0 - MACOS 64-BIT

4.9.0 - WINDOWS 64-BIT



## 2. SPRINGTOOLSUITE

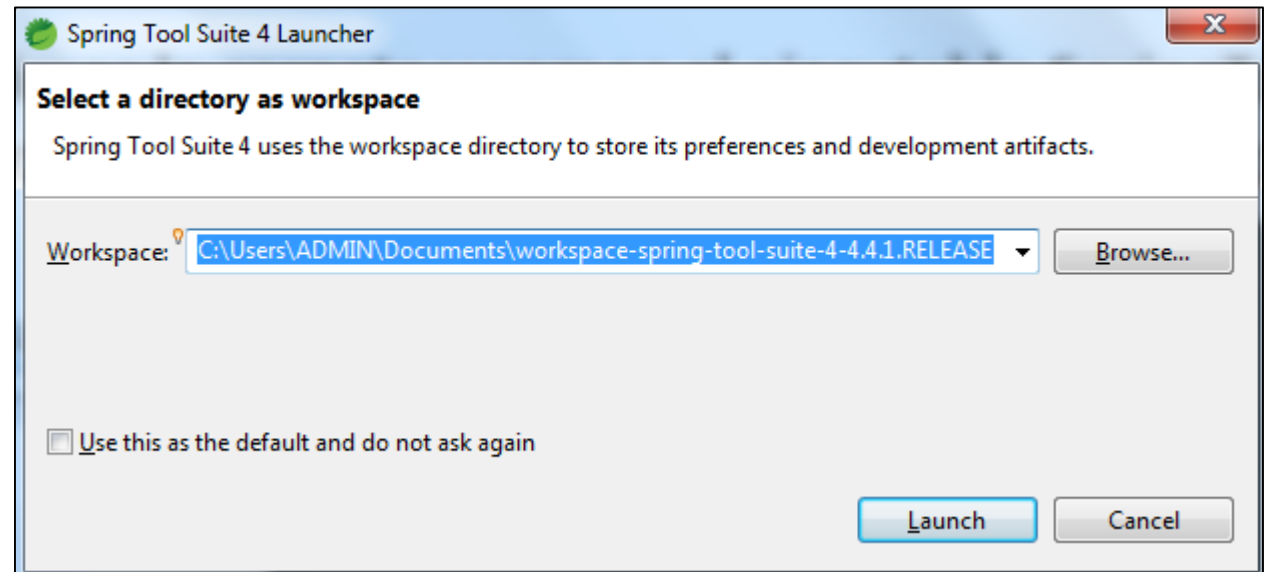
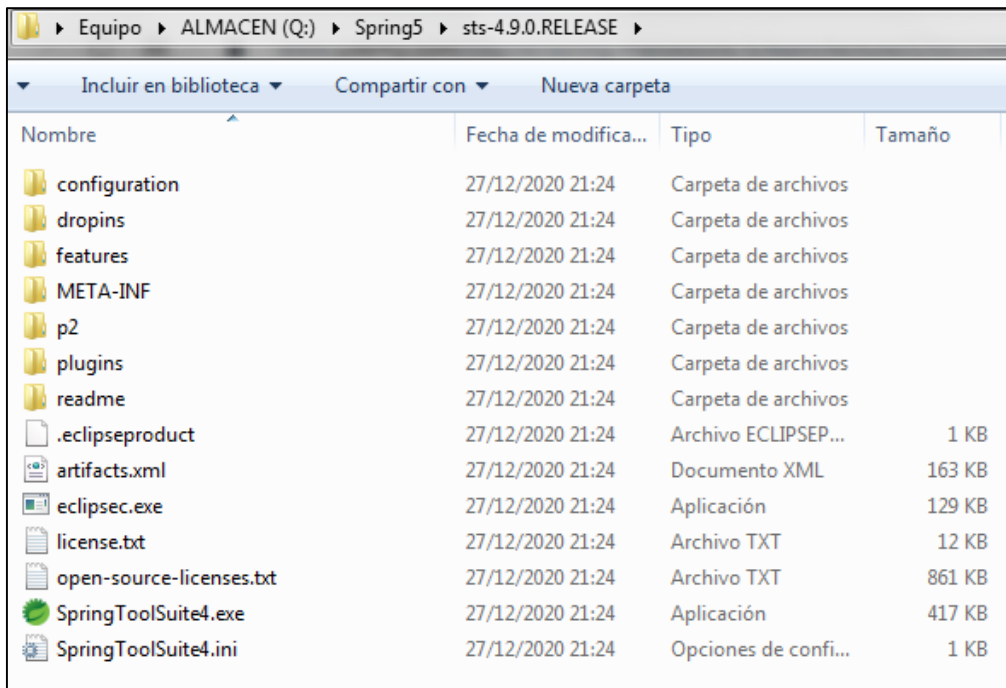
**Paso 2)** Una vez descargado SpringToolSuite, lo descomprimimos en una carpeta:





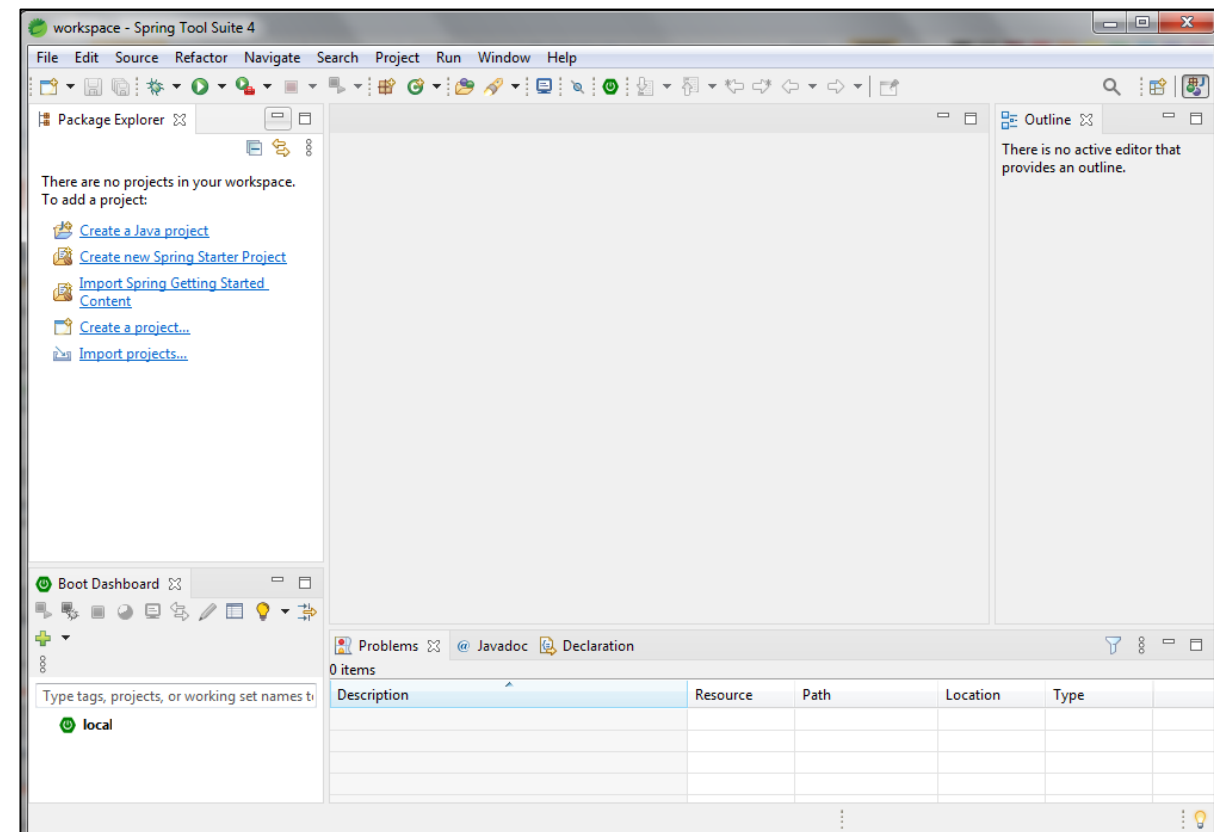
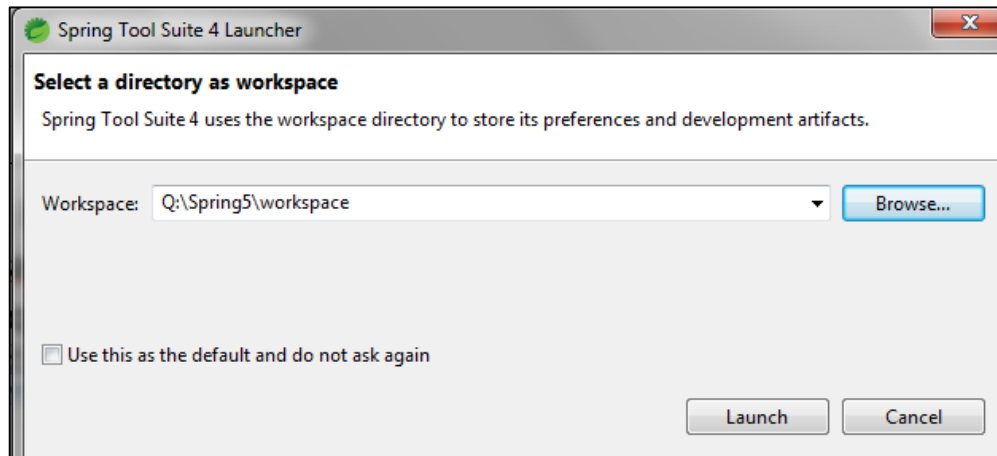
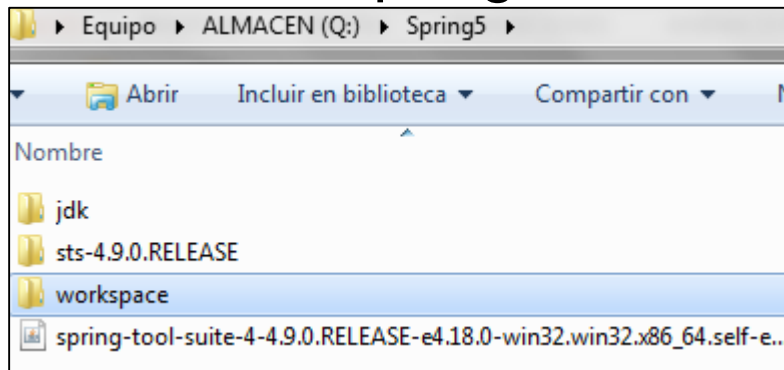
## 2. SPRINGTOOLSUITE

**Paso 3)** Abrimos la carpeta y vemos el ejecutable SpringToolSuite4. Hacemos dobleclick para iniciarlo y lo primero que nos pregunta es por nuestro espacio de trabajo o workspace, donde vamos a guardar y a crear nuestros proyectos.



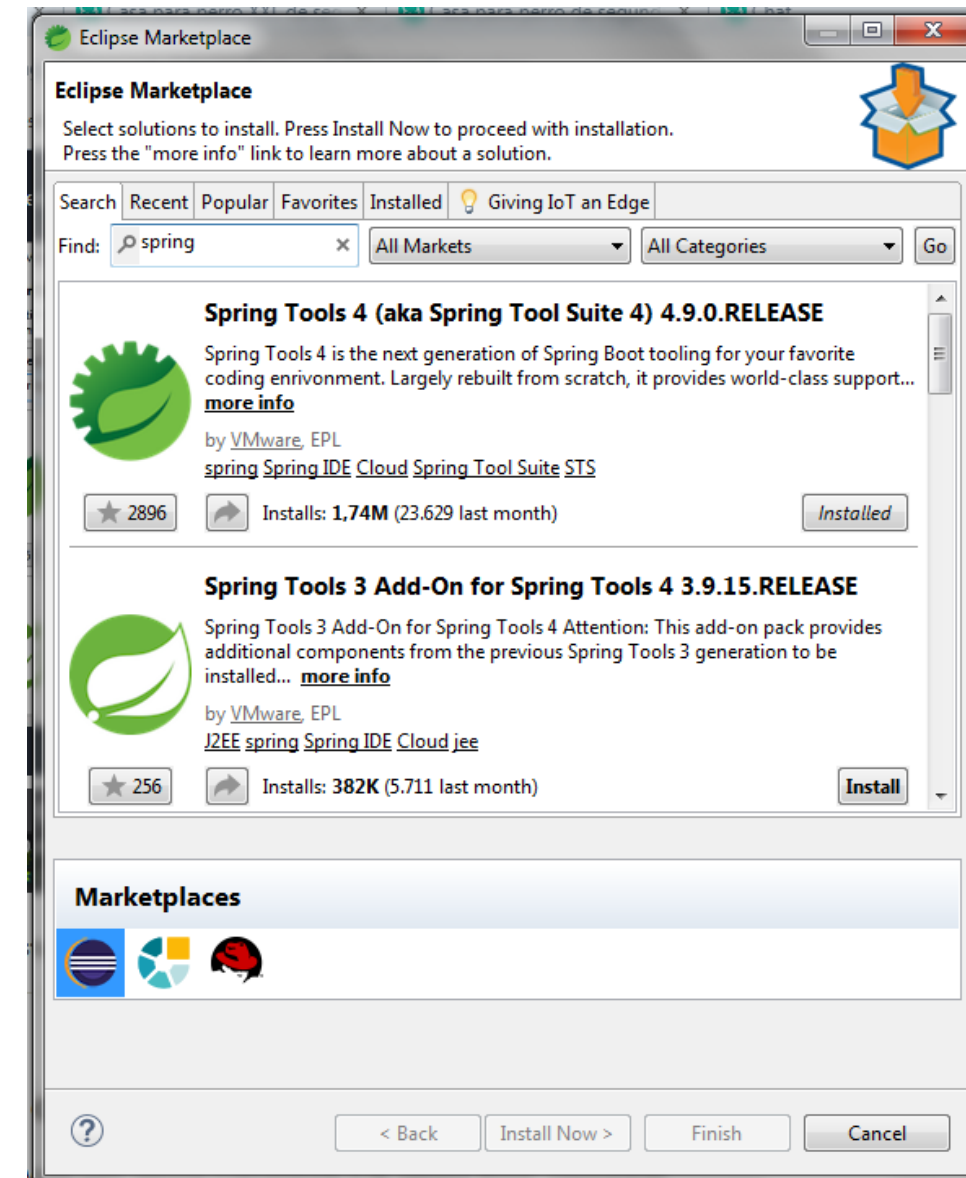
## 2. SPRINGTOOLSUITE

**Paso 4)** En el mismo directorio del SpringToolSuite, creamos la carpeta workspace, y así se la indicamos al SpringToolSuite. Cuando arranca vemos que se trata de un eclipse disfrazado de Spring.



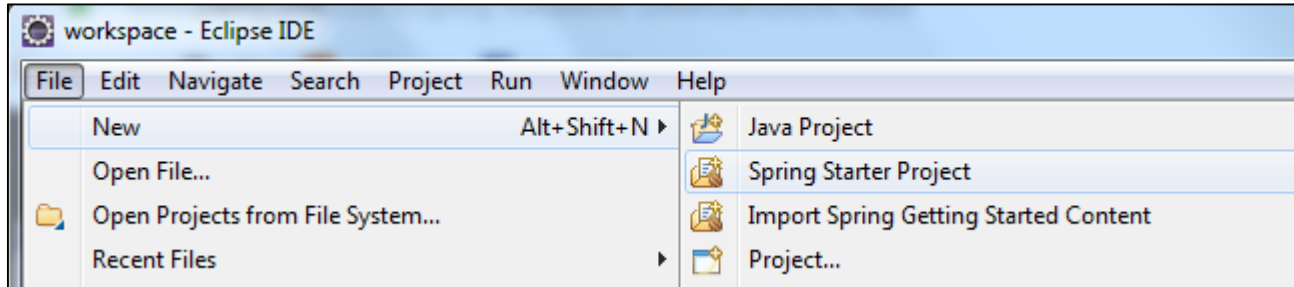
## 2. SPRINGTOOLSUITE

**Paso 5)** Vamos a Help/Eclipse MarketPlace y buscamos Spring. Vemos que el plugin Spring Tools 4 se encuentra instalado

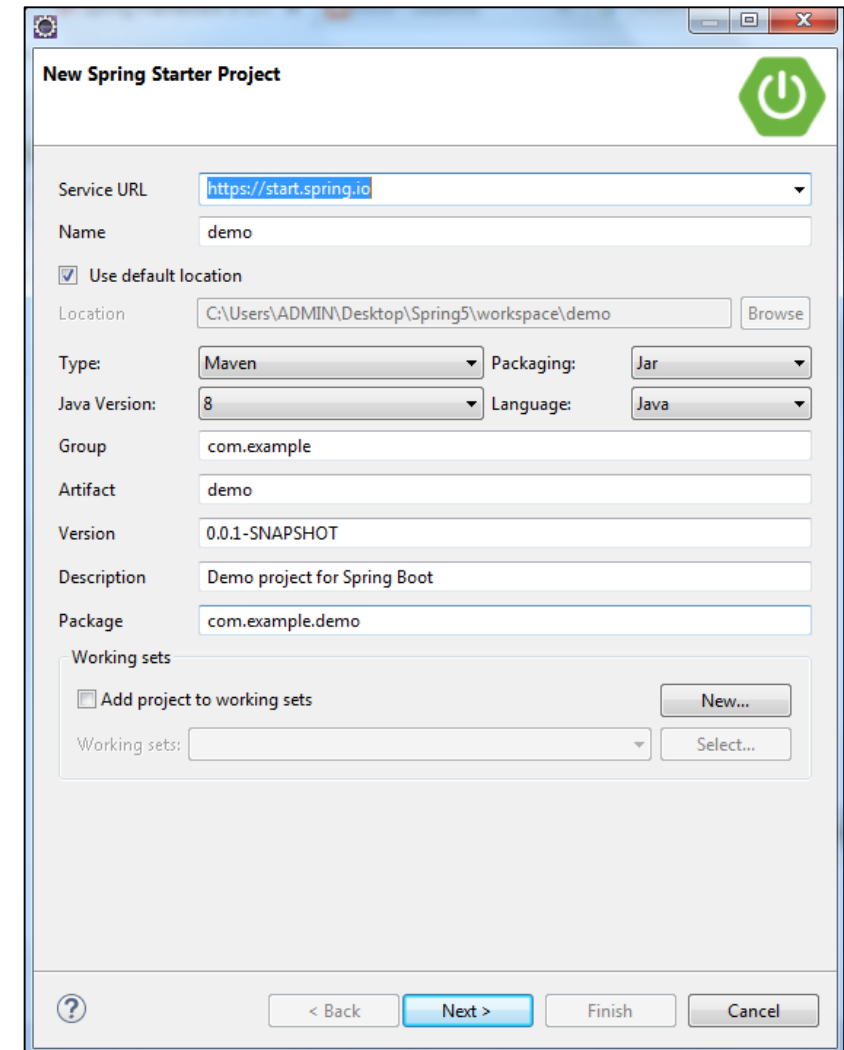
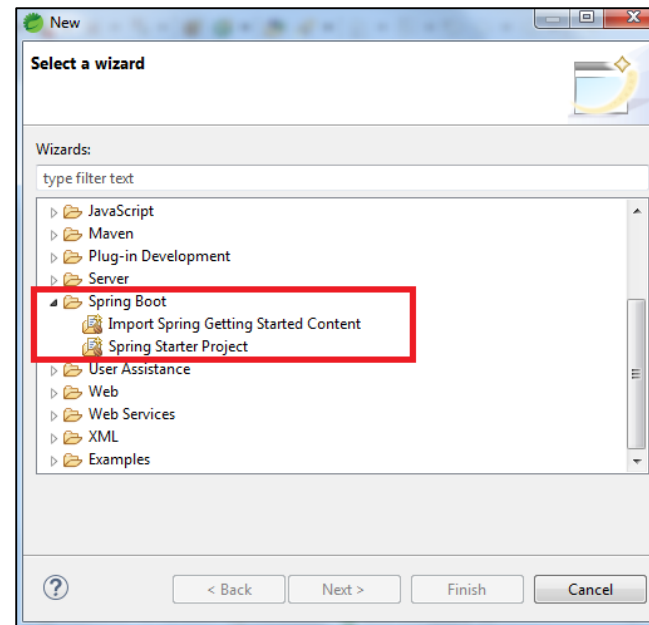


# 3. CREAR PROYECTO SPRING

**Paso 1)** Seleccionamos File/New/Spring Starter Project:



En caso de que no aparezca debemos ir por el camino largo. File/New/Otros. Ir a Spring boot



# 3. CREAR PROYECTO SPRING

**Paso 2)** Parámetros de creación del proyecto:

**Name:** Nombre del proyecto, sin espacios en blanco  
primer-springboot-web

**Type:** Maven/Gradle. Administrador de dependencias para generar el proyecto. Permiten crear nuestro proyecto, compilar o construir nuestro jar o war final que vamos a desplegar o publicar en producción.

**Java Version:** 16, 11, 8. Usaremos la 8

**Group:** Muy similar a los packages de java. Estructura de package y ahí podemos agrupar varias aplicaciones de Maven dentro del mismo Group Id

The screenshot shows the 'New Spring Starter Project' dialog box. The fields are filled with the following values:

- Service URL: `https://start.spring.io`
- Name: `primer-springboot-web`
- Location: `Q:\Spring5\workspace\primer-springboot-web` (with a 'Browse' button)
- Type: `Maven` (dropdown)
- Packaging: `Jar` (dropdown)
- Java Version: `8` (dropdown)
- Language: `Java` (dropdown)
- Group: `com.example`
- Artifact: `primer-springboot-web`
- Version: `0.0.1-SNAPSHOT`
- Description: `Demo project for Spring Boot`
- Package: `com.example.demo`

At the bottom, there is a 'Working sets' section with a checkbox 'Add project to working sets' and a 'New...' button. Below this is a 'Working sets:' dropdown and a 'Select...' button. The bottom of the dialog has four buttons: '?', '< Back', 'Next >', 'Finish', and 'Cancel'.

# 3. CREAR PROYECTO SPRING

---

**Paso 2)** Parámetros de creación del proyecto:

**Packaging:** Tipo de empaquetado del proyecto: jar/war. Jar se recomienda cuando creamos aplicaciones con SpringBoot. Inicialmente uno podría pensar que war es para proyectos web y jar para aplicaciones estándar, pero con jar también podemos crear proyectos web.

Spring boot incluye un servidor embebido Tomcat que permite desplegar aplicaciones en jar. Es mucho mas portable y mucho mas fácil desplegar aplicaciones con jar. Sólo se necesita el JDK, y el comando `java -jar nom_proyecto`. No requiere instalar ningún servidor como Tomcat externo, JBOSS O Glassfish.

Utilizamos War cuando queremos publicar en un servidor externo que no sea el que viene dentro de SpringBoot: JBOSS, Glasfish, Tomcat externo, o trabajamos con jsp. Cuando tenemos vistas por ejemplo con Thymeleaf o aplicaciones con API Rest Backend utilizamos jar.

# 3. CREAR PROYECTO SPRING

---

**Paso 2)** Parámetros de creación del proyecto:

**Artifact:** Nombre del proyecto

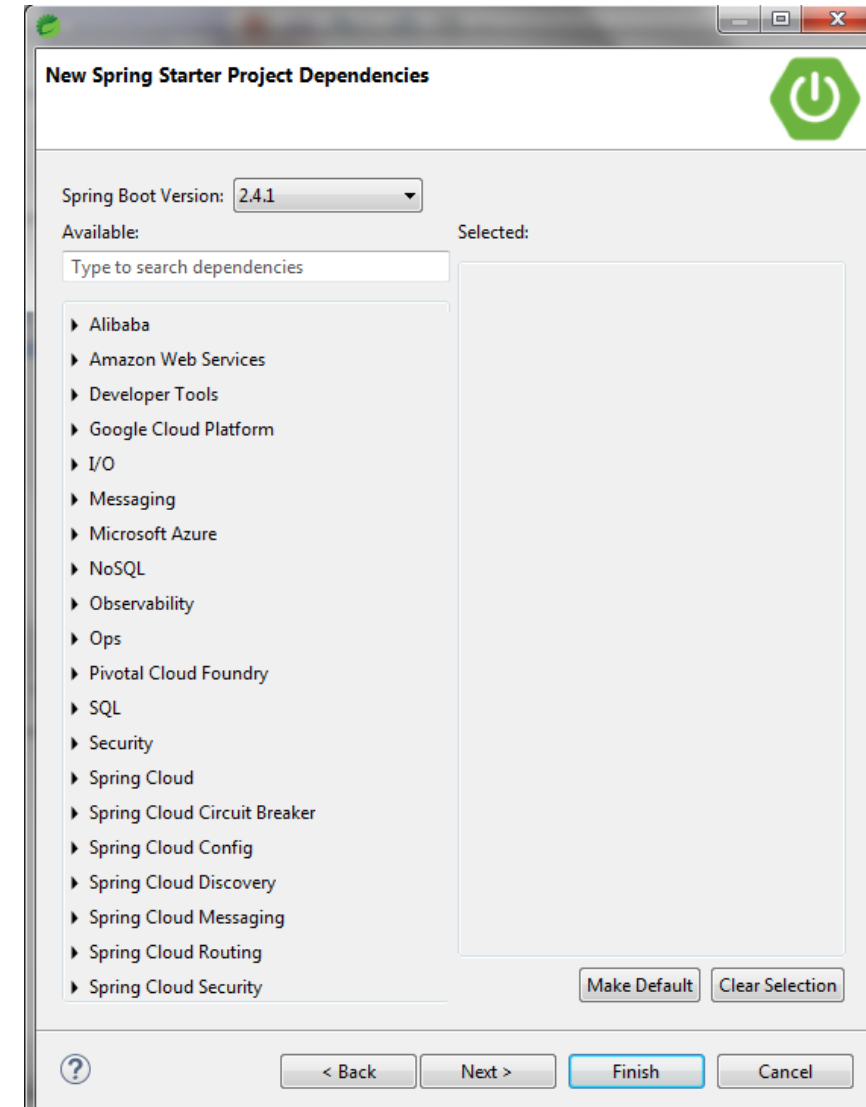
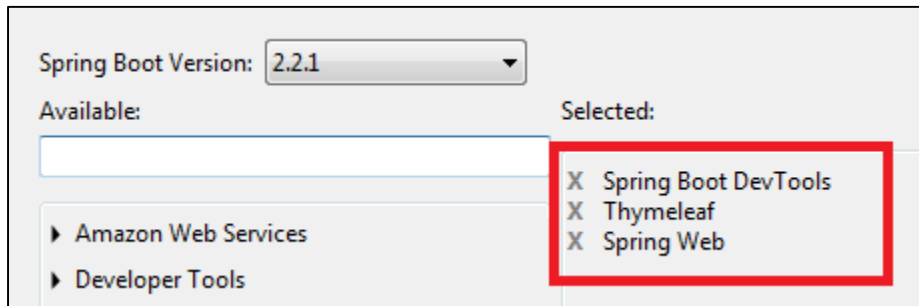
**Version:** La versión que le vamos a dar

**Package:** Package de java. Package base principal o raíz de nuestra aplicación springboot. Un package se compone del nombre\_del\_dominio\_de\_la\_empresa + nombre\_área\_trabajo o nombre\_del\_proyecto que estamos desarrollando. No hay una regla tan estricta para esto. Podemos poner el mismo package en Group ID.

# 3. CREAR PROYECTO SPRING

**Paso 3) Spring Boot Version:** Poner la última versión estable, que no sea SNAPSHOT. Estas versiones no se recomiendan porque son como versiones beta en desarrollo, solo se debe marcar la última final estable

**Dependencias:** Aquí configuramos las dependencias del proyecto SpringBoot. Para trabajar con una aplicación web, hemos de seleccionar los siguientes componentes: Spring web, SpringBoot Devtools y Thymeleaf.

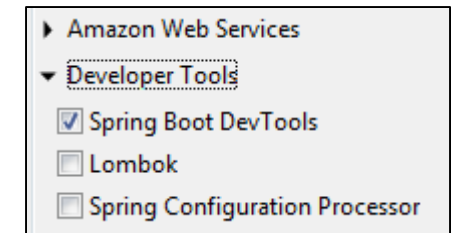
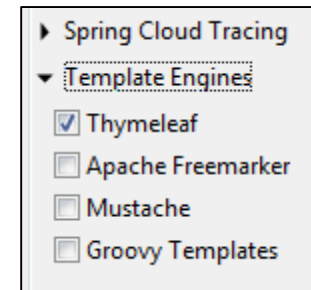
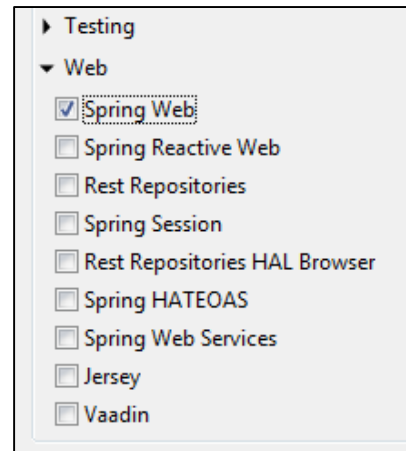




# 3. CREAR PROYECTO SPRING

**Paso 3)** Componente **Spring web**, dentro de Web. Es la base que incluye otras dependencias como:

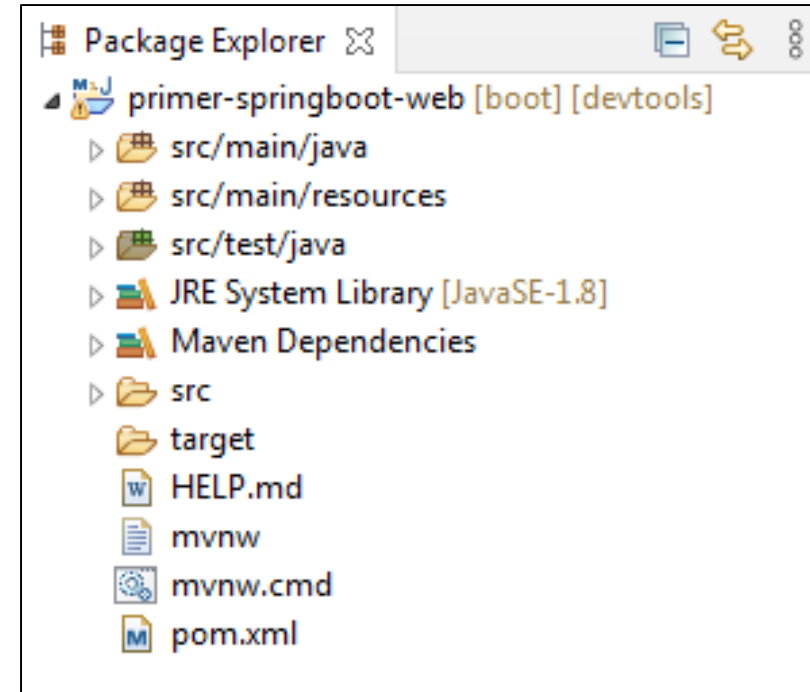
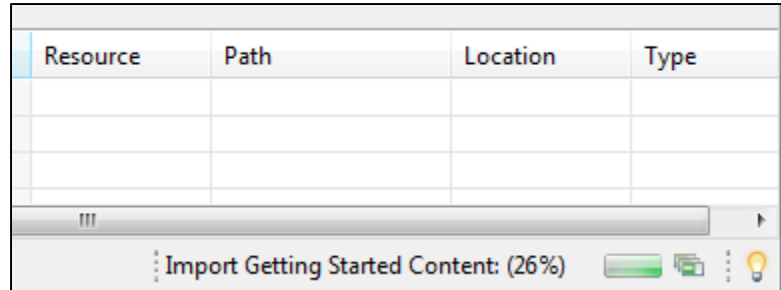
- Spring-core, spring-web
- Spring webmvc, Servlet-api
- Spring-boot-starter-tomcat
- Spring-boot-autoconfigure



- Componente **Thymeleaf** dentro de Templates Engines. Es el motor de plantillas que mas se usa y recomienda. Integracion total con Spring boot
- Componente **Spring Boot DevTools** dentro de Developer Tools. Es importante ya que cualquier cambio que hagamos en nuestro código java, se reinicia el servidor automáticamente, actualizando el despliegue de forma optimizada y rápida, sin tener que hacerlo de forma manual.

### 3. CREAR PROYECTO SPRING

**Paso 4)** Hacemos click en finish y puede tardar un rato la creación del proyecto por la descarga de las dependencias:



No ha marcado error pero lo podría haber hecho por la descarga mal de un fichero que marcara el fichero como corrupto por una marca en rojo

# 3. CREAR PROYECTO SPRING

**Paso 5)** En el fichero pom.xml podemos ver la versión de java, que se puede cambiar, las dependencias anteriormente agregadas (thymeleaf, web, devtools), todo configurado automáticamente con SpringBoot.



The screenshot shows a code editor with a file named 'primer-springboot-web/pom.xml'. The code is XML and defines project metadata and dependencies. Three sections are highlighted with red rectangles:

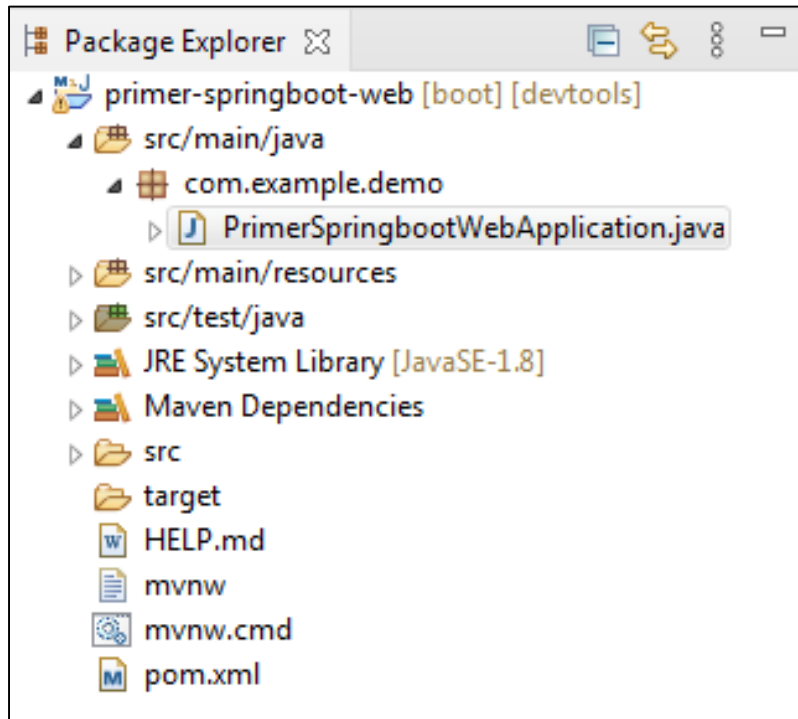
- Project Metadata (lines 11-15):** Includes groupId, artifactId, version, name, and description.
- Properties (lines 17-19):** Defines the java.version as 1.8.
- Dependencies (lines 21-42):** Lists four dependencies from org.springframework.boot: spring-boot-starter-thymeleaf, spring-boot-starter-web, spring-boot-devtools (with runtime scope and optional=true), and spring-boot-starter-test (with test scope).

```
11 <groupId>com.example</groupId>
12 <artifactId>primer-springboot-web</artifactId>
13 <version>0.0.1-SNAPSHOT</version>
14 <name>primer-springboot-web</name>
15 <description>Demo project for Spring Boot</description>
16
17 <properties>
18   <java.version>1.8</java.version>
19 </properties>
20
21 <dependencies>
22   <dependency>
23     <groupId>org.springframework.boot</groupId>
24     <artifactId>spring-boot-starter-thymeleaf</artifactId>
25   </dependency>
26   <dependency>
27     <groupId>org.springframework.boot</groupId>
28     <artifactId>spring-boot-starter-web</artifactId>
29   </dependency>
30
31   <dependency>
32     <groupId>org.springframework.boot</groupId>
33     <artifactId>spring-boot-devtools</artifactId>
34     <scope>runtime</scope>
35     <optional>true</optional>
36   </dependency>
37   <dependency>
38     <groupId>org.springframework.boot</groupId>
39     <artifactId>spring-boot-starter-test</artifactId>
40     <scope>test</scope>
41   </dependency>
42 </dependencies>
```

### 3. CREAR PROYECTO SPRING

**Paso 6)** Debajo de nuestro package com.example.demo están todas las clases del proyecto: controlador, models, service, etc.

Las clases con la notación component o service, permite registrar estos componentes Spring, pero para eso deben de estar dentro de este contexto.



```
PrimerSpringbootWebApplication.java
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4
5
6 @SpringBootApplication
7 public class PrimerSpringbootWebApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(PrimerSpringbootWebApplication.class, args);
11     }
12
13 }
14
```

Es el inicio de la aplicación PrimerSpringbootWebApplication. En el método estatico run de SpringApplication se le pasa como argumentos el class de nuestro fichero de inicio, junto con los argumentos del main

# 3. CREAR PROYECTO SPRING

**Paso 7)** En `PrimerSpringbootWebApplication` encontramos la anotación `@SpringBootApplication`. Haciendo Ctrl click vemos la configuración de spring

Podemos resaltar los componentes:

**@SpringBootConfiguration**

**@EnableAutoConfiguration** habilita la configuración automática

**@ComponentScan** que escanea y busca los componentes que están dentro de este package con anotaciones controller, service, repository. Busca todos los beans, y los registra.

```
SpringBootApplication.class
39 /**
40  * Indicates a {@link Configuration} class that declares one or more
41  * {@link Bean} methods and also triggers {@link EnableAutoConfiguration}
42  * auto-configuration and {@link ComponentScan} component scanning. This is
43  * annotation that is equivalent to declaring {@code @Configuration},
44  * {@code @EnableAutoConfiguration} and {@code @ComponentScan}.
45  *
46  * @author Phillip Webb
47  * @author Stephane Nicoll
48  * @author Andy Wilkinson
49  * @since 1.2.0
50  */
51 @Target(ElementType.TYPE)
52 @Retention(RetentionPolicy.RUNTIME)
53 @Documented
54 @Inherited
55 @SpringBootConfiguration
56 @EnableAutoConfiguration
57 @ComponentScan(excludeFilters = { @Filter(type = FilterType.CUSTOM, classes
58     @Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExclude
59 public @interface SpringBootApplication {
60
61 /**
62  * Exclude specific auto-configuration classes such that they will never
63  * @return the classes to exclude
64  */
65 @AliasFor(annotation = EnableAutoConfiguration.class)
66 Class<?>[] exclude() default {};
67
```

# 3. CREAR PROYECTO SPRING

**Paso 8)** Dependencias de Maven. Se descargan de forma automática en el proyecto indicadas por el fichero pom.xml.

Spring boot starter thymeleaf

Spring web

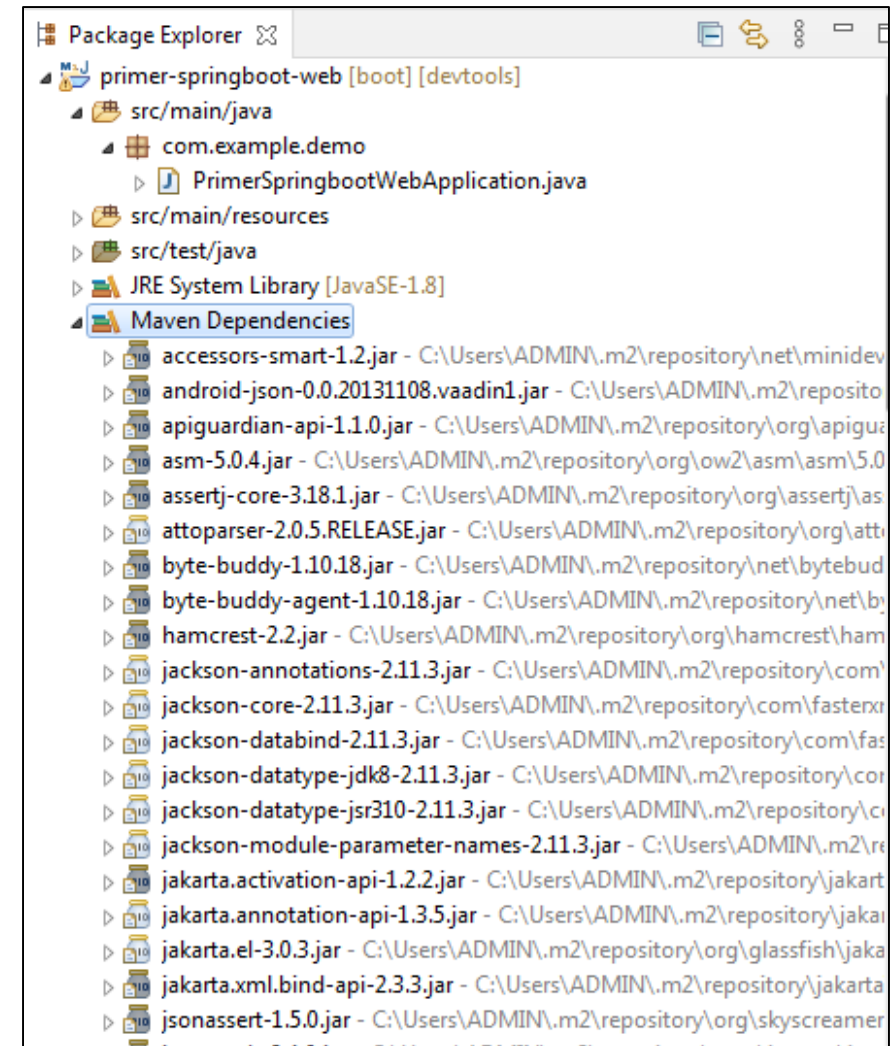
Spring boot devtools

Spring core

Tomcat embeded core

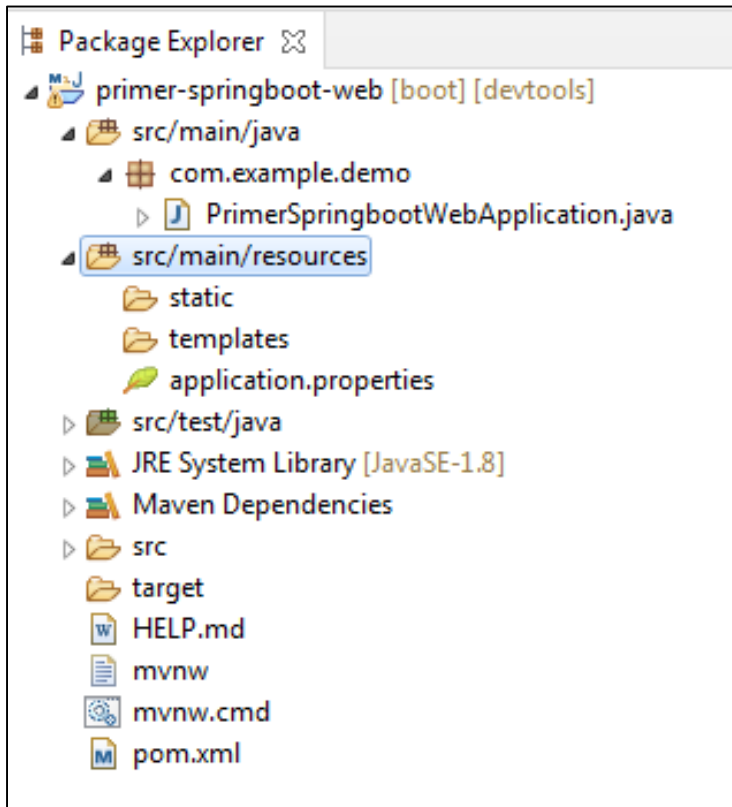
Spring web mvc

Spring aop



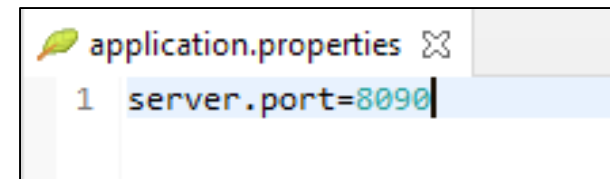
# 3. CREAR PROYECTO SPRING

**Paso 9) Directorio src/main/resources.** Tenemos el fichero **application.properties** con la configuración principal de springboot. Esta vacío porque toda la configuración viene por debajo.



Se puede utilizar para cambiar el puerto del servidor de 8080 a 8090, por ejemplo.

Importante: No dejar después un espacio en blanco. Aquí se pueden configurar parámetros de acceso a base de datos Mysql, el connectString, usuario, password, la clase jdbcDriver, JPA o hibernate, el dialecto del motor de base de datos, etc





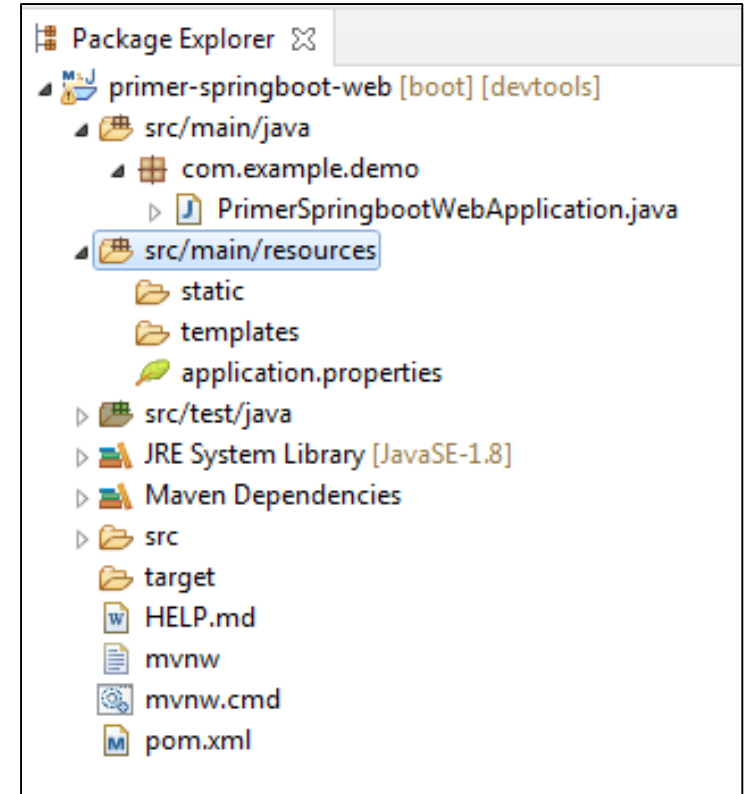
# 3. CREAR PROYECTO SPRING

## Paso 10)

**Carpeta templates:** Para guardar las plantillas de las vistas de los controladores. Se recomienda utilizar plantillas thymeleaf como motor de plantillas por encima de JSP ya que es mucho mas robusto y esta mas optimizado para HTML5. Se trabaja a través de atributos html que son propios de thymeleaf. No se compila a un servlet ni tiene dependencias con api servlet. Tampoco utiliza etiquetas ni taglib como jsp. El código es mucho mas limpio y puro.

**Carpeta static:** Para guardar los recursos estáticos: css, javascript, imágenes.

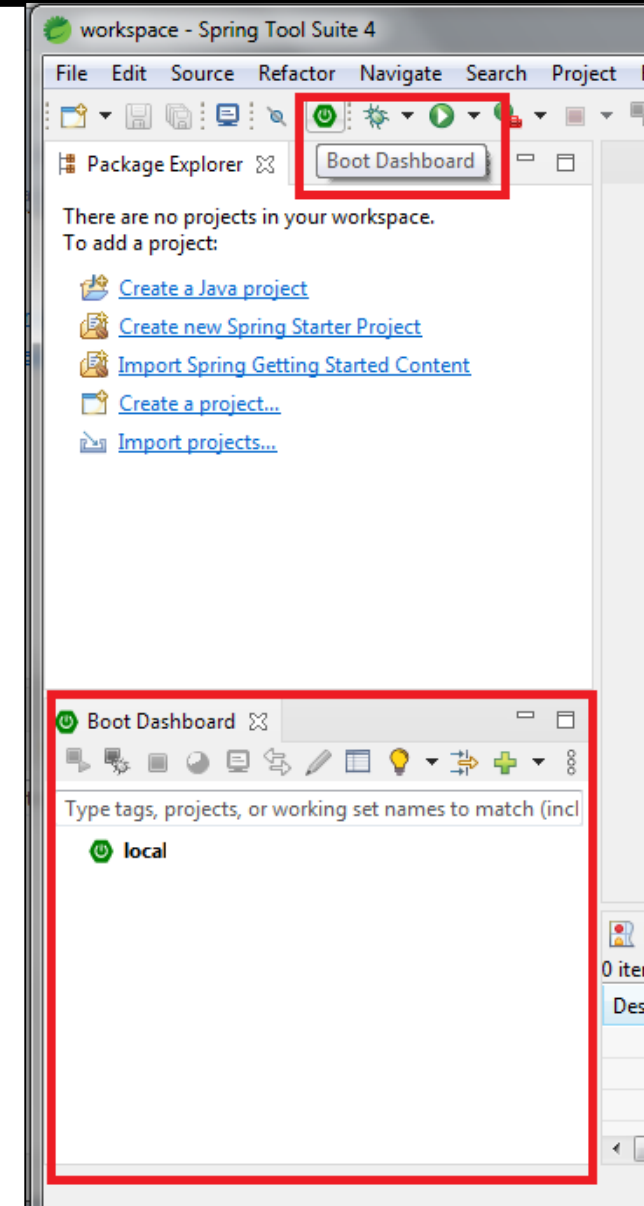
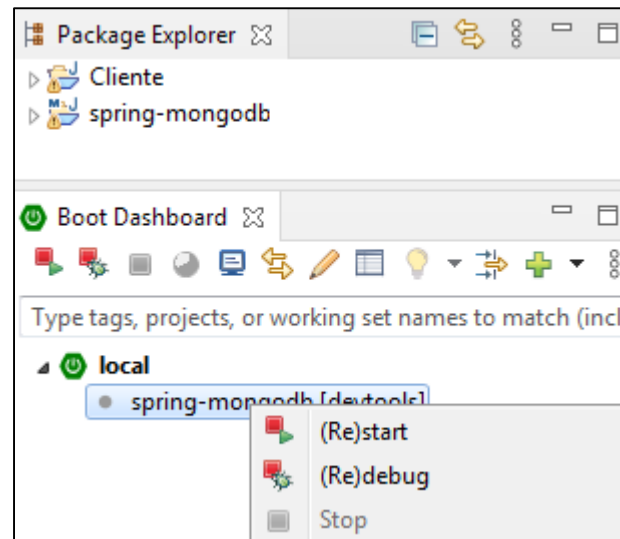
**Carpeta target:** Donde aparecerá el jar, el archivo ejecutable que podemos copiar y ejecutar con java -jar.





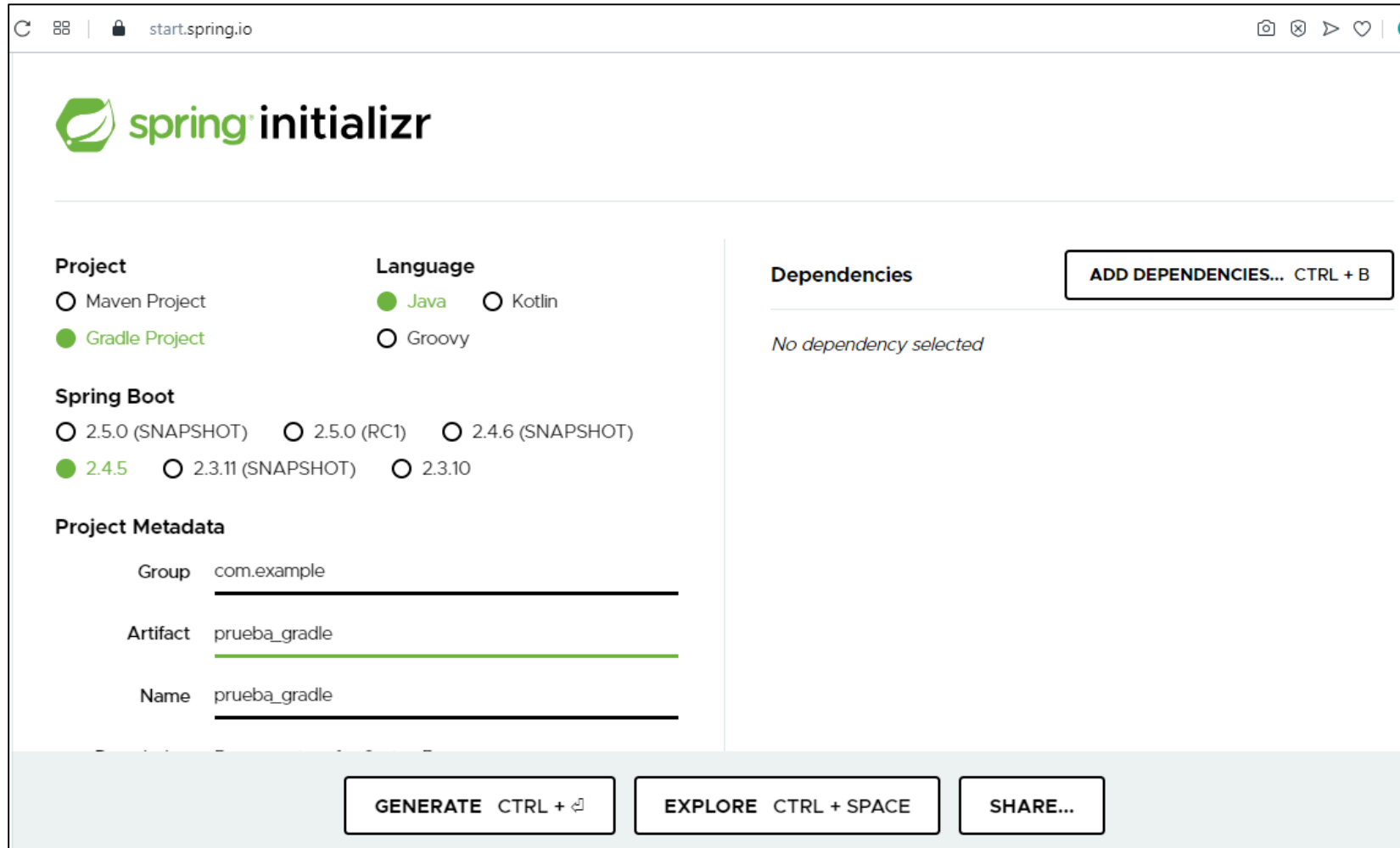
# 3. CREAR PROYECTO SPRING

**Paso 11) Pestaña Boot Dashboard.** Cada vez que creamos o abrimos un proyecto SpringBoot, éste aparece desplegado en la pestaña Dashboard bajo la etiqueta local. Es el lugar ideal para detenerlo, iniciarlo o reiniciarlo. La primera vez que se levanta un proyecto se puede hacer mediante **Run as/Spring boot**. Si se tiene que volver a reiniciar la aplicación, no se debe volver a hacer de esta forma porque da error al estar tomado el puerto. Se tiene que hacer desde la pestaña Boot Dashboard.



# 4. IMPORTACIÓN DESDE SPRING.IO

**Paso 1.** Vamos a la pagina <https://start.spring.io> y creamos un proyecto gradle, que llamaremos prueba\_gradle. No agregamos ninguna dependencia:



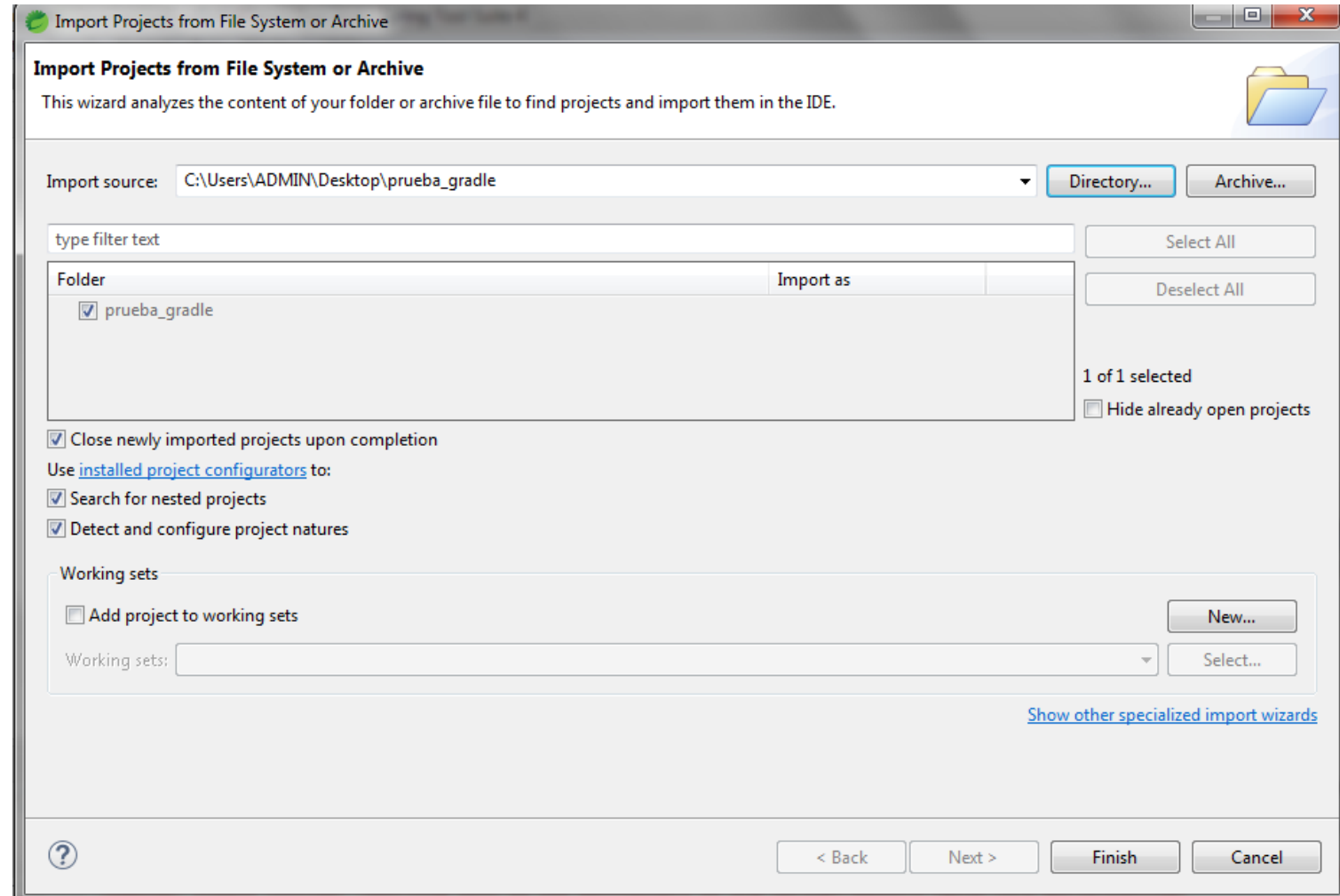
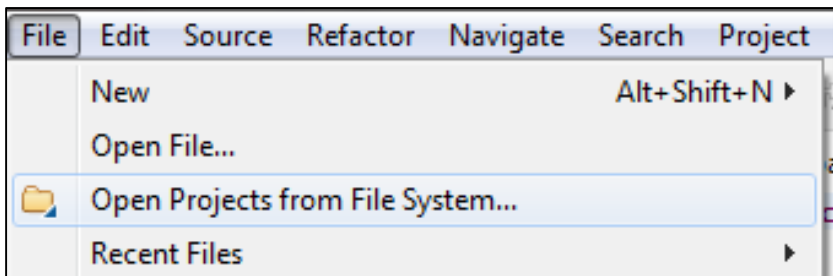
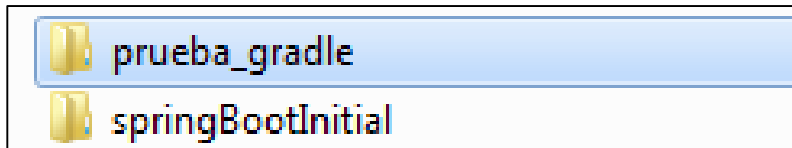
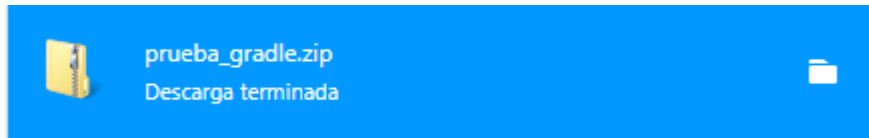
The screenshot shows the Spring Initializr web application interface. The browser address bar displays "start.spring.io". The page features the "spring initializr" logo at the top. Below the logo, the configuration options are organized into sections:

- Project:** Includes radio buttons for "Maven Project" and "Gradle Project" (which is selected).
- Language:** Includes radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot:** Includes radio buttons for various versions: "2.5.0 (SNAPSHOT)", "2.5.0 (RC1)", "2.4.6 (SNAPSHOT)", "2.4.5" (selected), "2.3.11 (SNAPSHOT)", and "2.3.10".
- Project Metadata:** Includes input fields for "Group" (containing "com.example"), "Artifact" (containing "prueba\_gradle"), and "Name" (containing "prueba\_gradle").
- Dependencies:** A section with the text "No dependency selected" and a button labeled "ADD DEPENDENCIES... CTRL + B".

At the bottom of the page, there are three buttons: "GENERATE CTRL + G", "EXPLORE CTRL + SPACE", and "SHARE...".

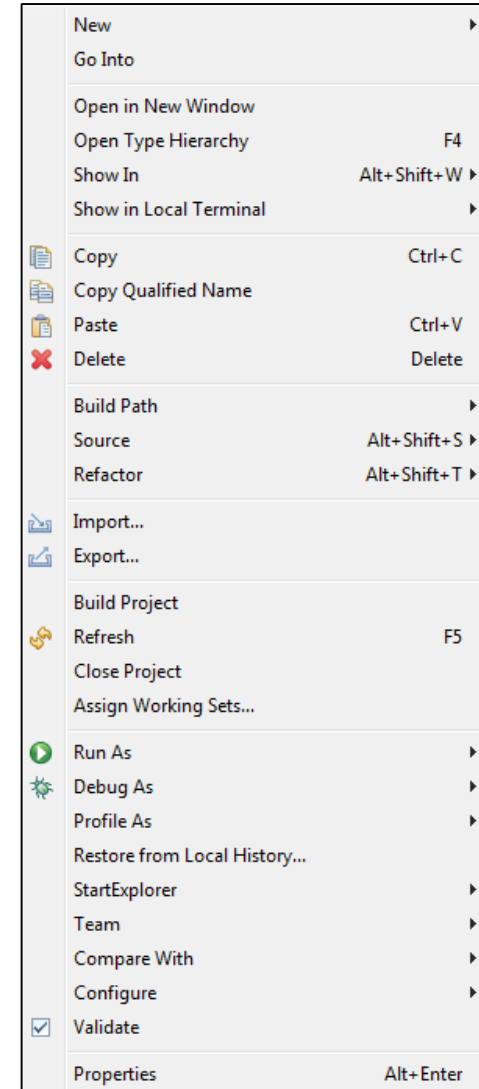
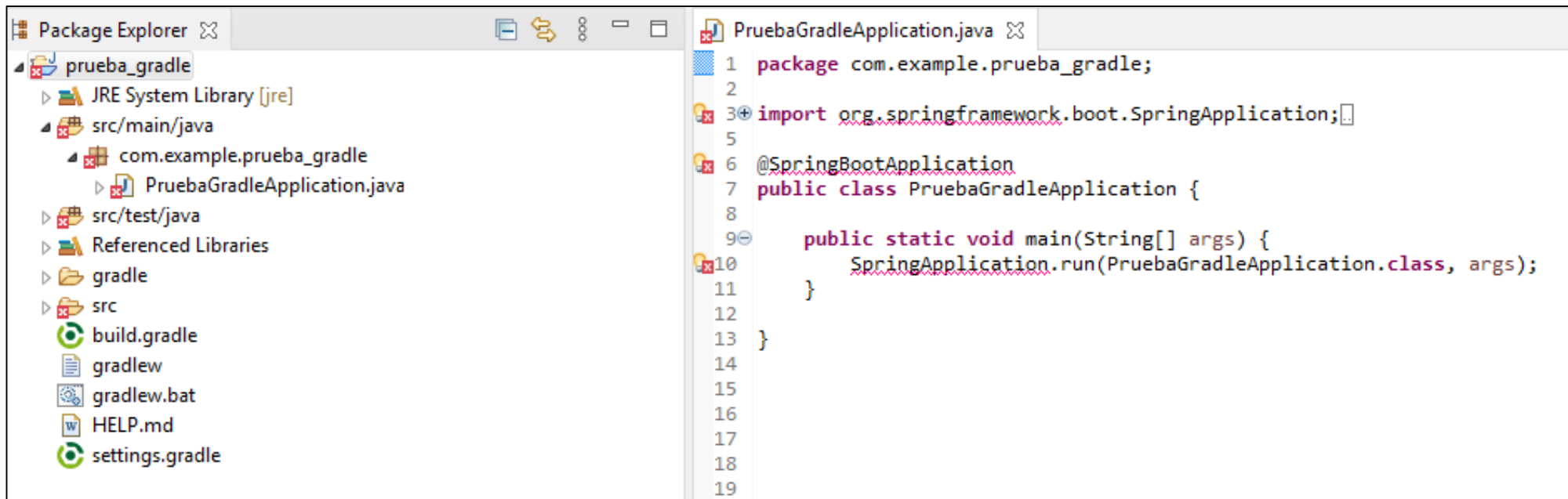
# 4. IMPORTACIÓN DESDE SPRING.IO

**Paso 2.** Descomprimos el zip y lo importamos a Eclipse, en File/Open Projects from File System...



# 4. IMPORTACIÓN DESDE SPRING.IO

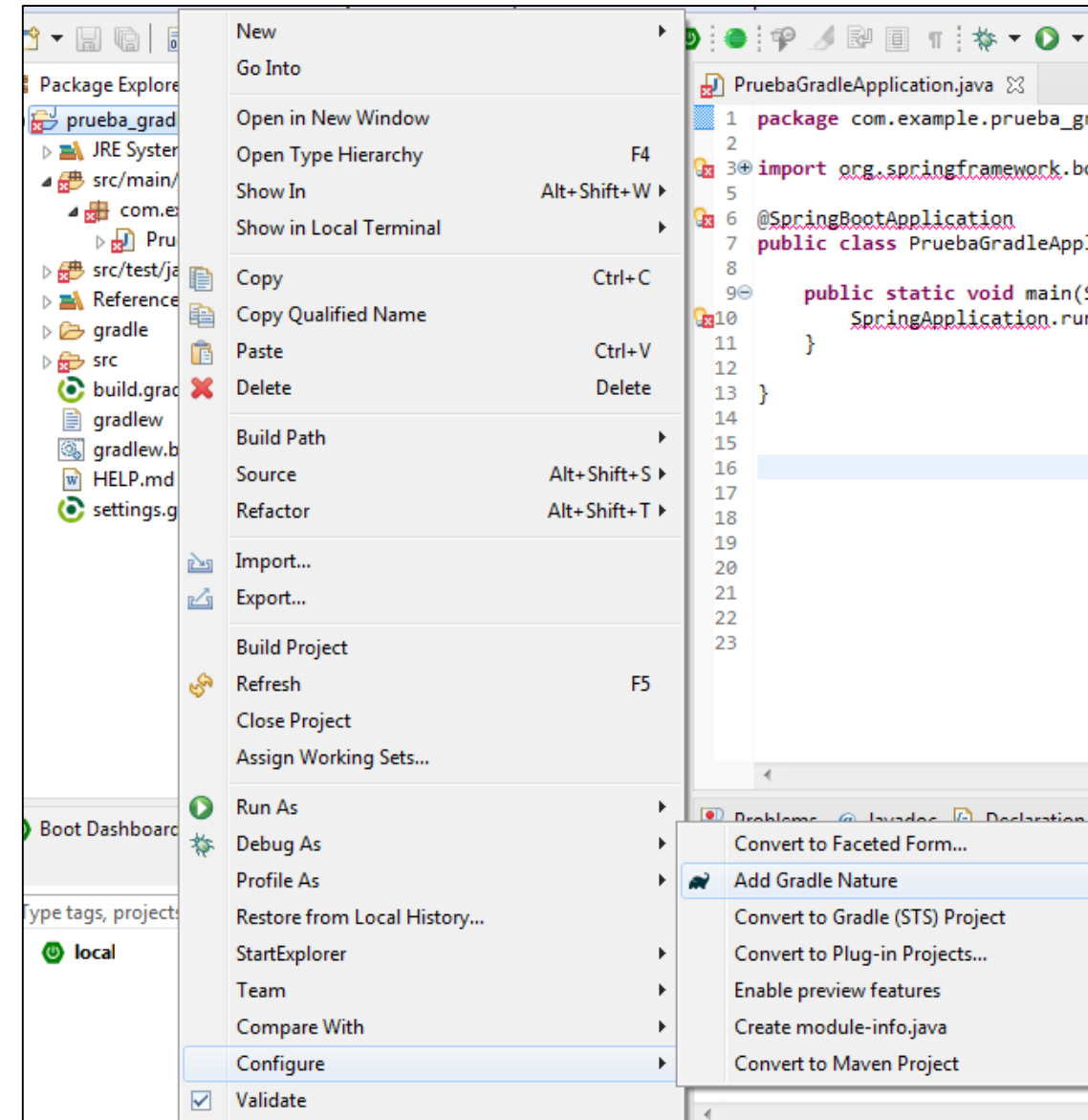
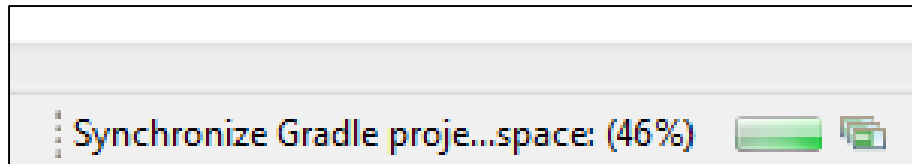
**Paso 3.** Vemos que el proyecto presenta errores y no reconoce las dependencias, ni como proyecto Gradle, ni como proyecto Spring. Hacemos click botón derecho encima del proyecto y vemos que no hay ninguna opción de **Gradle** ni de **Spring**, opciones que saldrían si el proyecto se reconociese como tal.



# 4. IMPORTACIÓN DESDE SPRING.IO

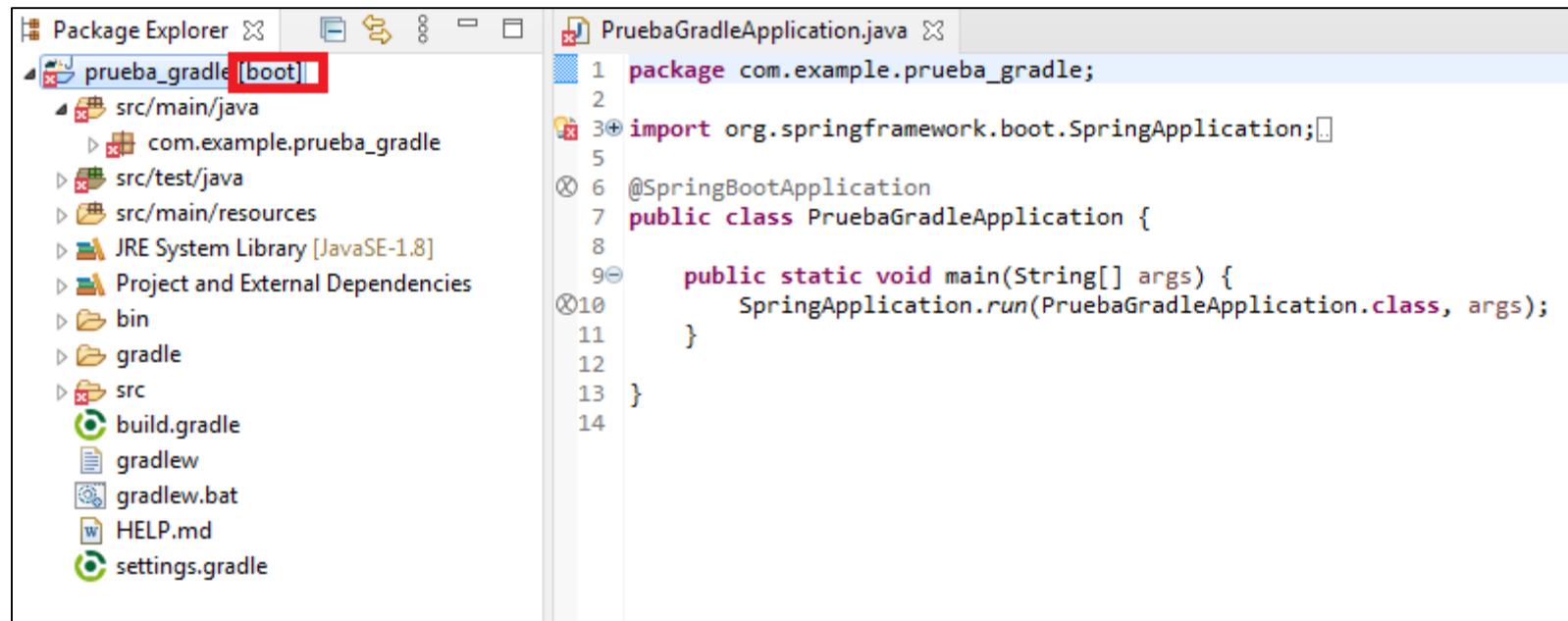
**Paso 4.** Para solventar este error debemos volver a configurar el proyecto como un proyecto Gradle. Hacemos click botón derecho encima del proyecto, y vamos a la opción **Configure/Add Gradle Nature**.

Eclipse inicia un proceso de sincronización en el repositorio Gradle que dura unos segundos:



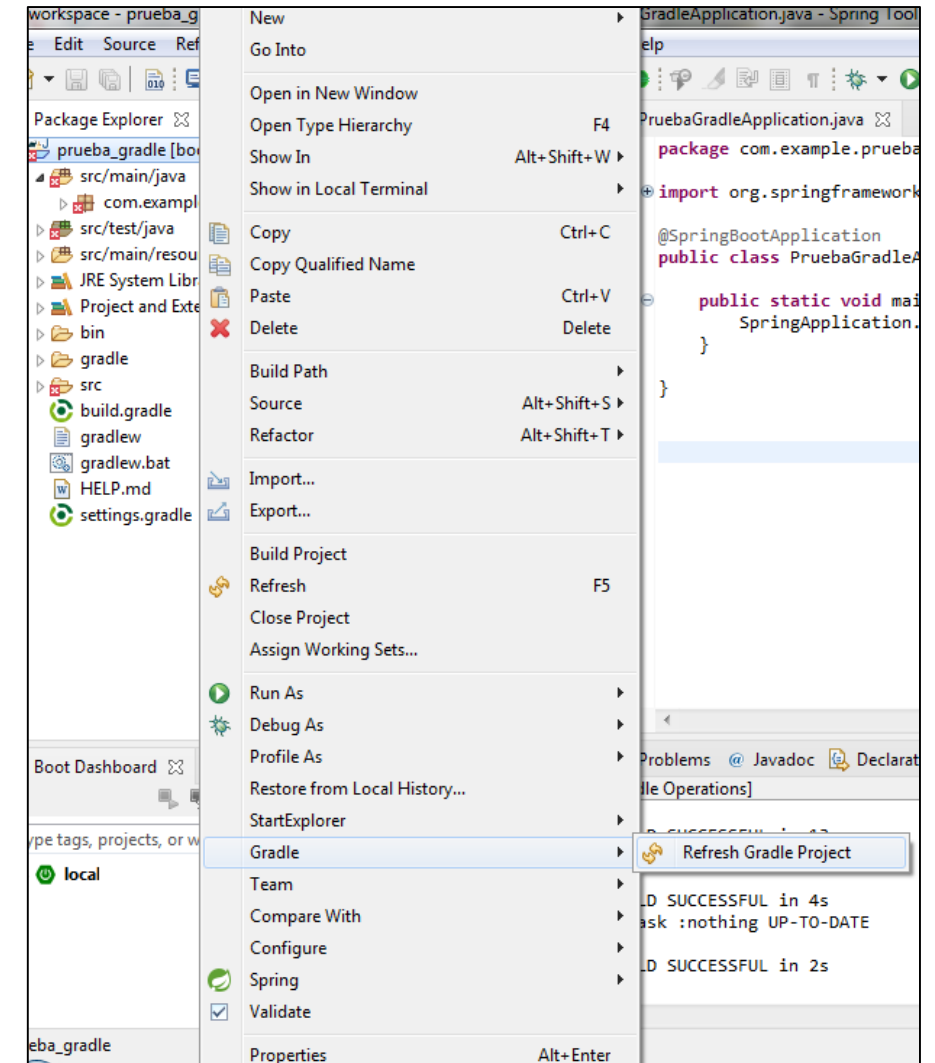
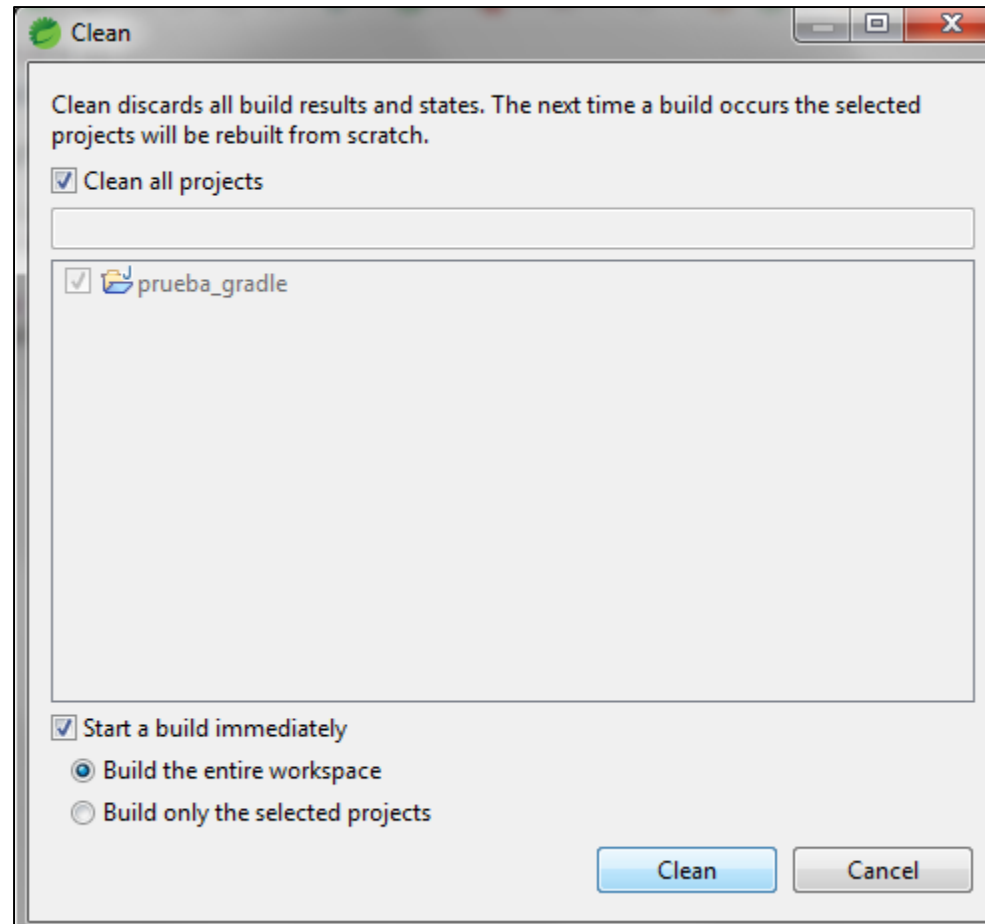
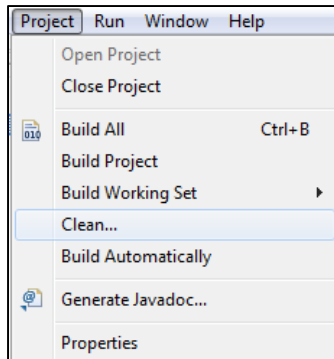
# 4. IMPORTACIÓN DESDE SPRING.IO

**Paso 5.** Después de este proceso, Eclipse por lo menos ya reconoce el proyecto como Spring. Solo falta un proceso mas para que todos los errores desaparezcan:



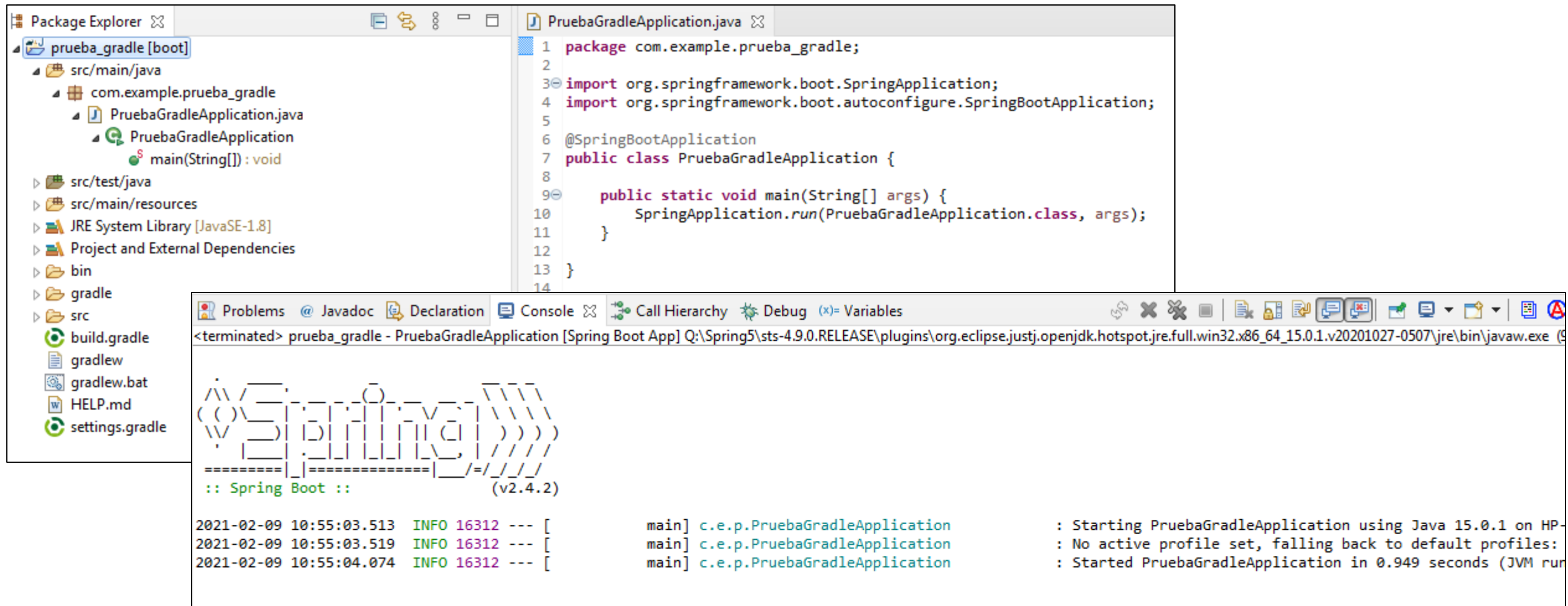
# 4. IMPORTACIÓN DESDE SPRING.IO

**Paso 6.** Podemos hacer un Clean del proyecto para actualizar las dependencias de Gradle.



# 4. IMPORTACIÓN DESDE SPRING.IO

**Paso 7.** Con cualquiera de los pasos anteriores, desaparecen todos los errores de la aplicación y podemos arrancar el servidor Tomcat SpringBoot sin problemas:



The screenshot displays an IDE interface with the following components:

- Package Explorer:** Shows the project structure for 'prueba\_gradle [boot]'. The 'src/main/java' directory contains the package 'com.example.prueba\_gradle', which includes the file 'PruebaGradleApplication.java'. This file contains a class 'PruebaGradleApplication' with a 'main' method.
- Editor:** Displays the code for 'PruebaGradleApplication.java'. The code is as follows:

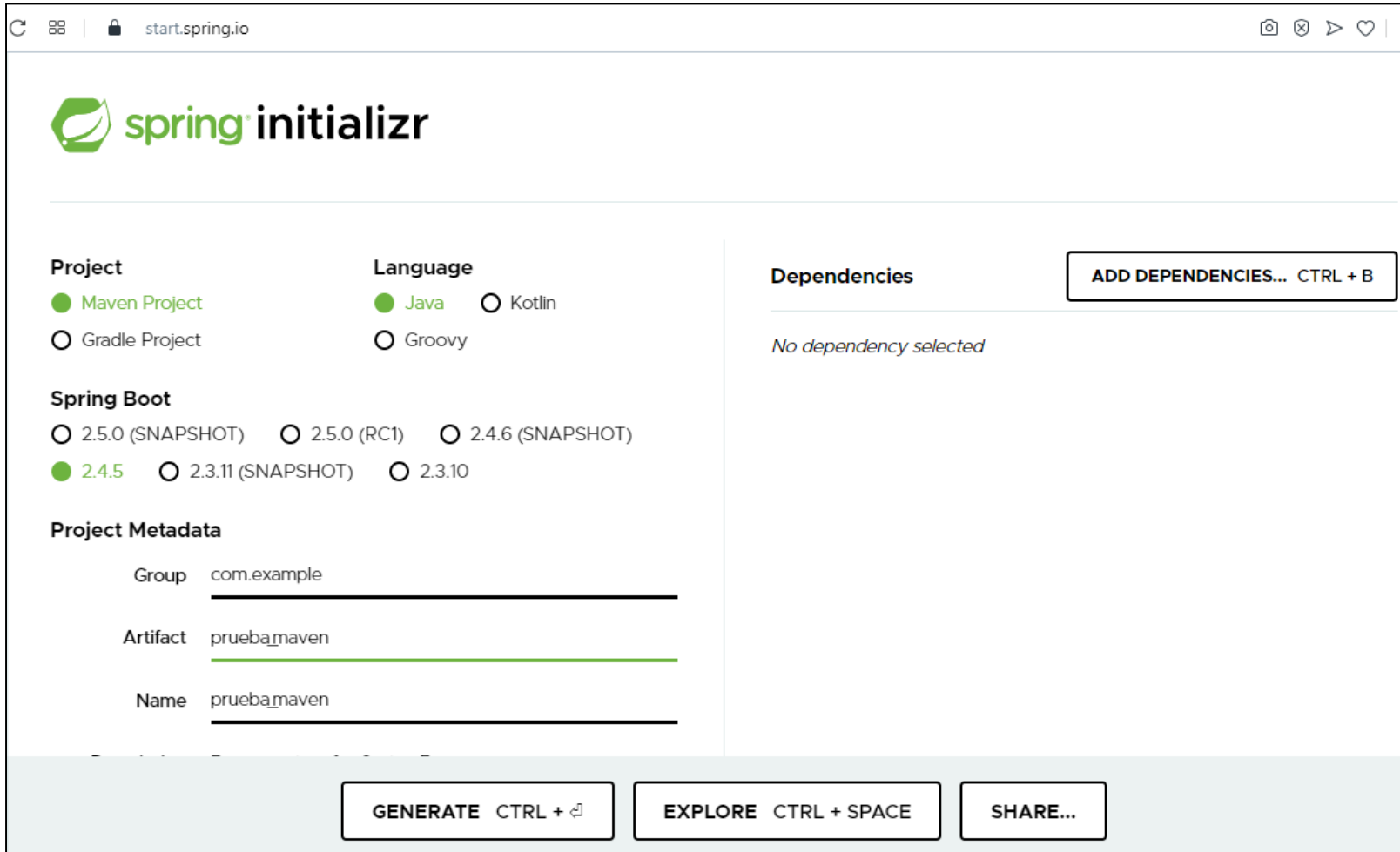
```
1 package com.example.prueba_gradle;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class PruebaGradleApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(PruebaGradleApplication.class, args);
11     }
12 }
13
14
```
- Console:** Shows the output of the application. It starts with the Spring Boot logo and the text ':: Spring Boot :: (v2.4.2)'. Below this, there are three log entries:

```
2021-02-09 10:55:03.513 INFO 16312 --- [main] c.e.p.PruebaGradleApplication : Starting PruebaGradleApplication using Java 15.0.1 on HP-
2021-02-09 10:55:03.519 INFO 16312 --- [main] c.e.p.PruebaGradleApplication : No active profile set, falling back to default profiles:
2021-02-09 10:55:04.074 INFO 16312 --- [main] c.e.p.PruebaGradleApplication : Started PruebaGradleApplication in 0.949 seconds (JVM run
```



# 4. IMPORTACIÓN DESDE SPRING.IO

**Paso 1.** Vamos a la pagina <https://start.spring.io> y creamos un proyecto maven, que llamaremos prueba\_maven. No agregamos ninguna dependencia:



The screenshot shows the Spring Initializr web application interface. The browser address bar displays "start.spring.io". The page features the "spring initializr" logo at the top. Below the logo, there are three main sections: "Project", "Language", and "Spring Boot".

- Project:** Includes radio buttons for "Maven Project" (selected) and "Gradle Project".
- Language:** Includes radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot:** Includes radio buttons for "2.5.0 (SNAPSHOT)", "2.5.0 (RC1)", "2.4.6 (SNAPSHOT)", "2.4.5" (selected), "2.3.11 (SNAPSHOT)", and "2.3.10".

Below these sections is the "Project Metadata" section, which contains three input fields:

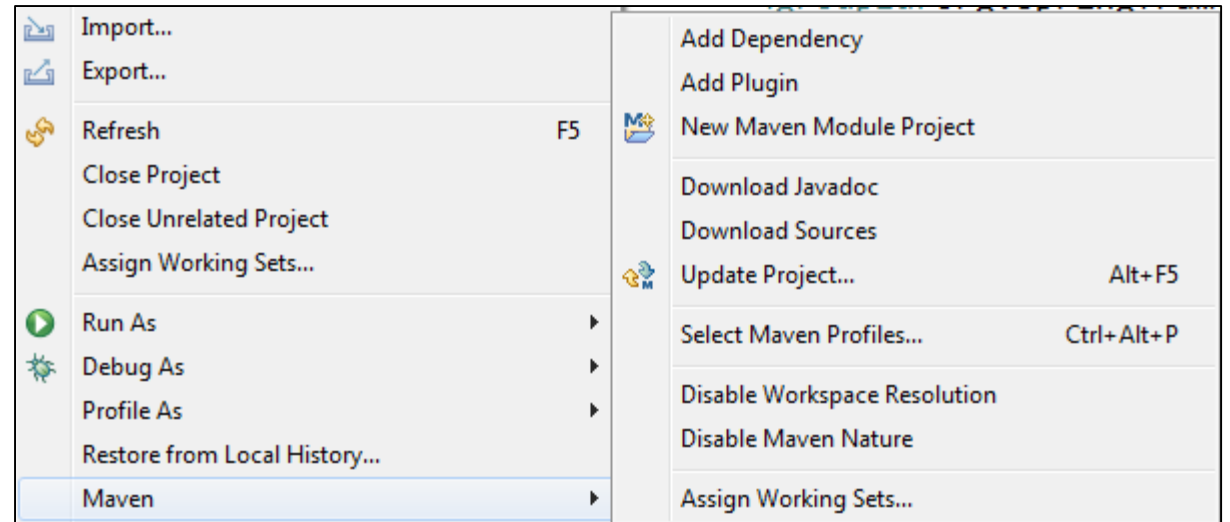
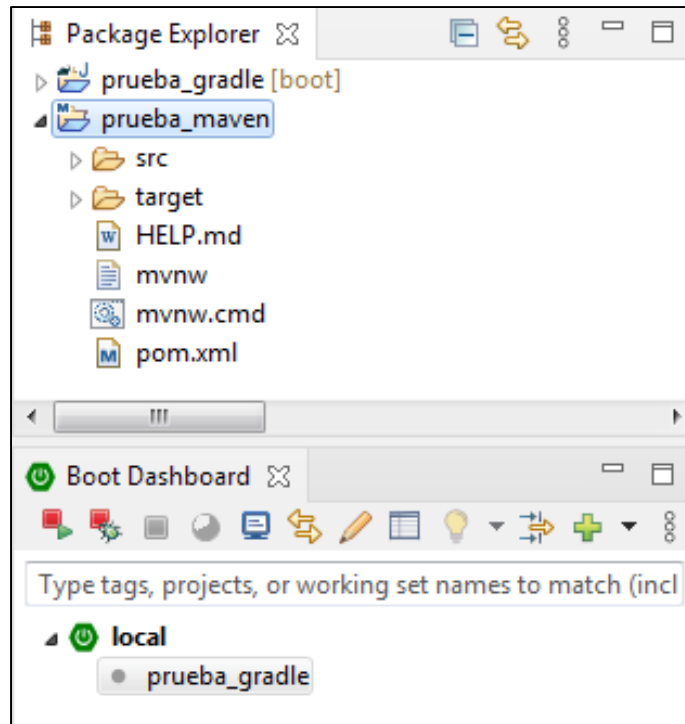
- Group:** com.example
- Artifact:** prueba\_maven
- Name:** prueba\_maven

On the right side of the interface, there is a "Dependencies" section with a button labeled "ADD DEPENDENCIES... CTRL + B". Below this button, it states "No dependency selected".

At the bottom of the page, there are three buttons: "GENERATE CTRL + G", "EXPLORE CTRL + SPACE", and "SHARE...".

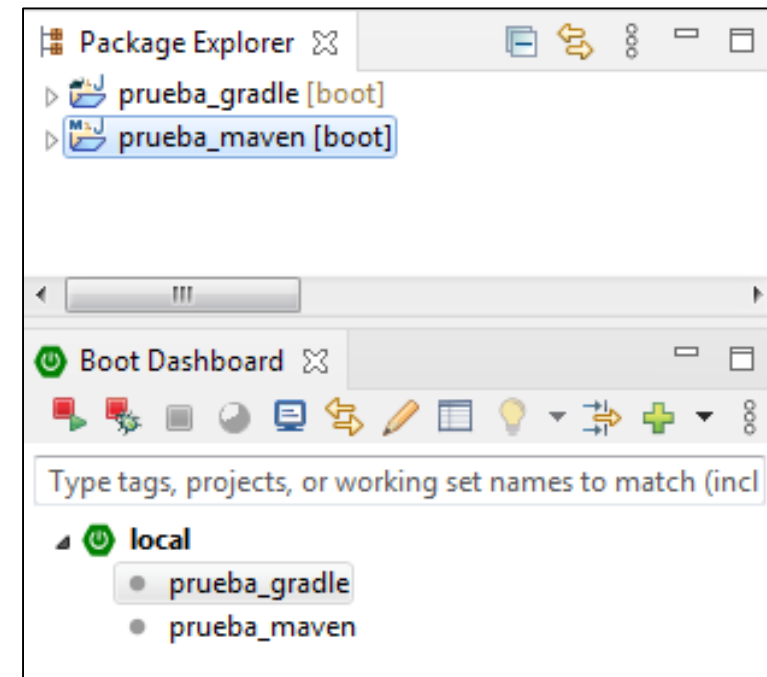
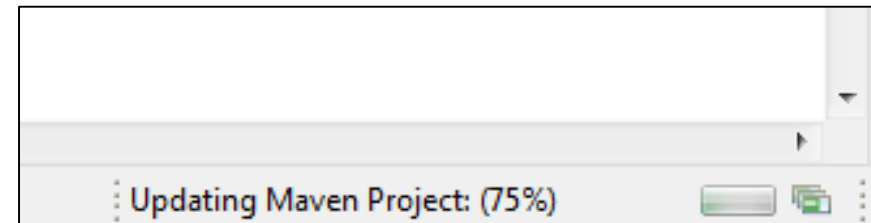
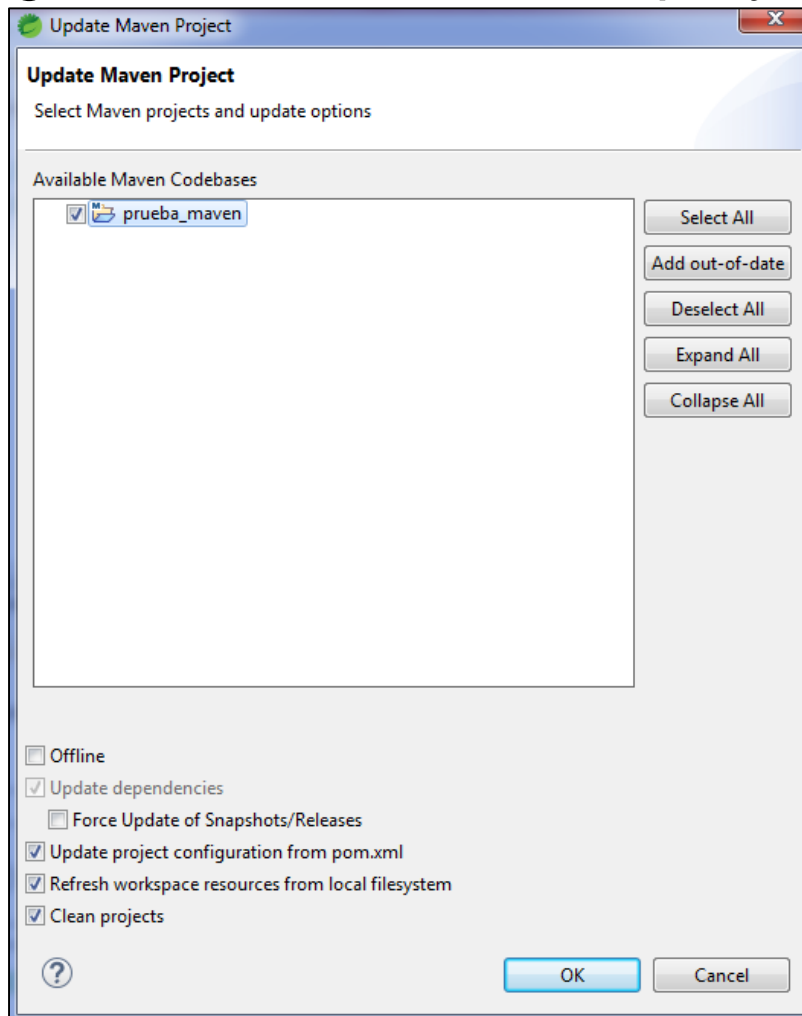
# 4. IMPORTACIÓN DESDE SPRING.IO

**Paso 2.** Descomprimos prueba\_maven.zip y lo importamos a Eclipse. Ahora el proyecto maven no presenta errores, se reconoce como un proyecto maven, pero no así como SpringBoot y por lo tanto no está desplegado en la pestaña DashBoard. Debemos volver a actualizar el proyecto como maven. Hacemos click botón derecho encima del proyecto, y vamos a la opción **Maven/UpdateProject**.



# 4. IMPORTACIÓN DESDE SPRING.IO

**Paso 3.** Iniciamos el proceso de actualización con el repositorio Maven que dura unos segundos. Finalmente el proyecto si se reconoce como SpringBoot:



# 5. AGREGAR DEPENDENCIAS

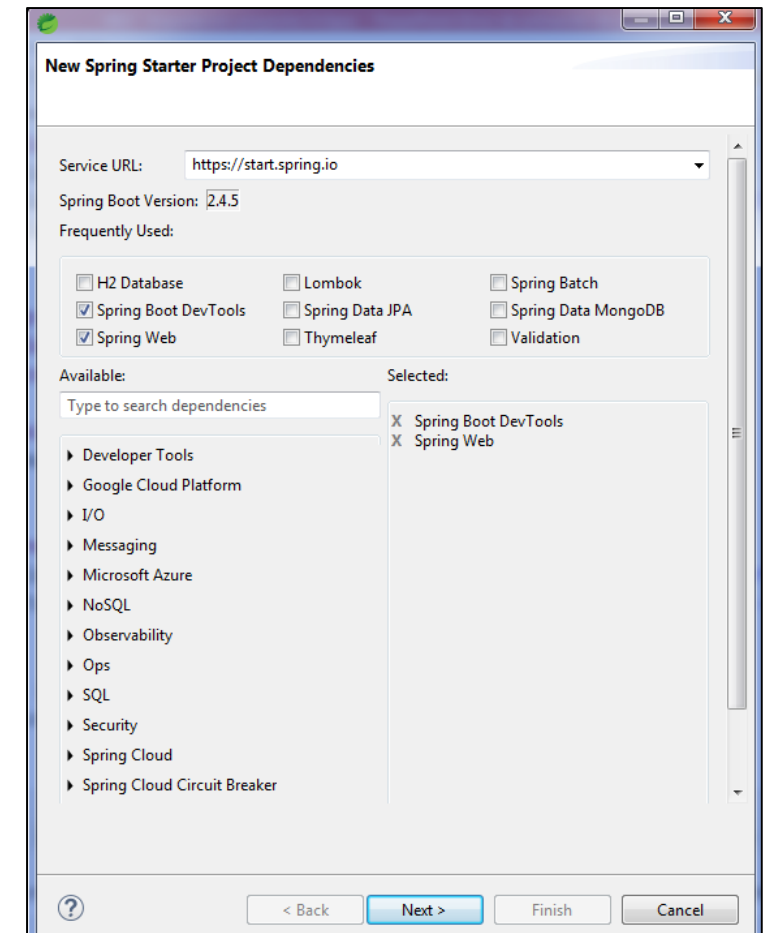
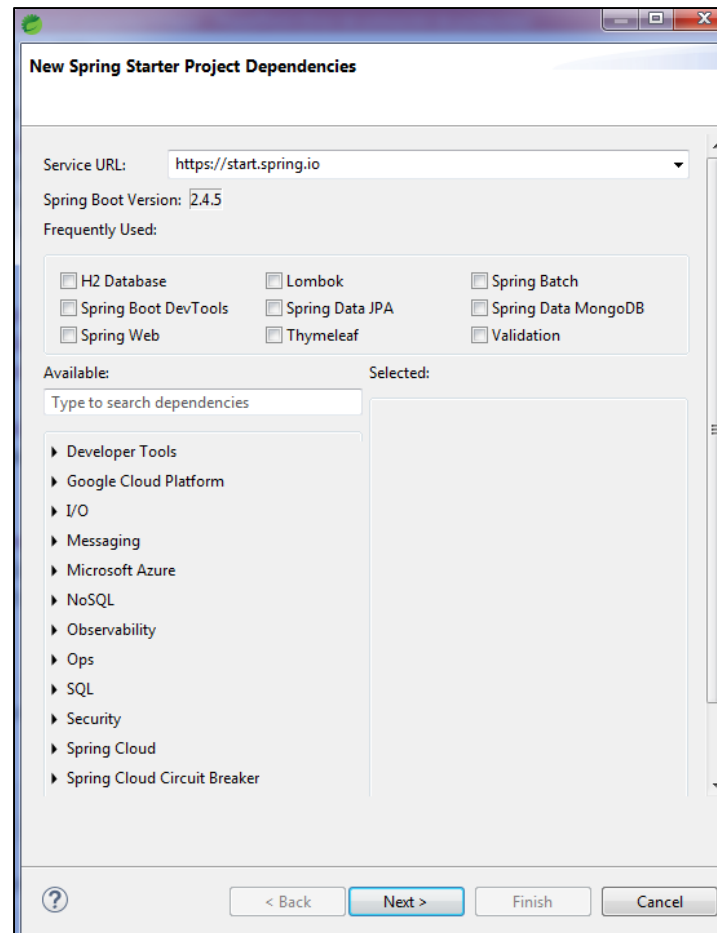
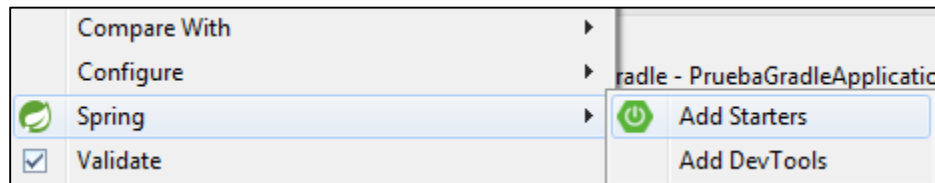
**Paso 1.** Nuestros proyectos de prueba solo contiene la dependencia por defecto que le distingue como proyecto Spring (spring-boot-starter y spring-boot-starter-test), pero no tiene ni la DevTools ni la Spring Web, necesarias para cualquier aplicación Spring

```
prueba_maven/pom.xml
17     <java.version>11</java.version>
18 </properties>
19 <dependencies>
20   <dependency>
21     <groupId>org.springframework.boot</groupId>
22     <artifactId>spring-boot-starter</artifactId>
23   </dependency>
24
25   <dependency>
26     <groupId>org.springframework.boot</groupId>
27     <artifactId>spring-boot-starter-test</artifactId>
28     <scope>test</scope>
29   </dependency>
30 </dependencies>
```

```
build.gradle
1 plugins {
2   id 'org.springframework.boot' version '2.4.5'
3   id 'io.spring.dependency-management' version '1.0.11.RELEASE'
4   id 'java'
5 }
6
7 group = 'com.example'
8 version = '0.0.1-SNAPSHOT'
9 sourceCompatibility = '11'
10
11 repositories {
12   mavenCentral()
13 }
14
15 dependencies {
16   implementation 'org.springframework.boot:spring-boot-starter'
17   testImplementation 'org.springframework.boot:spring-boot-starter-test'
18 }
19
20 test {
21   useJUnitPlatform()
22 }
```

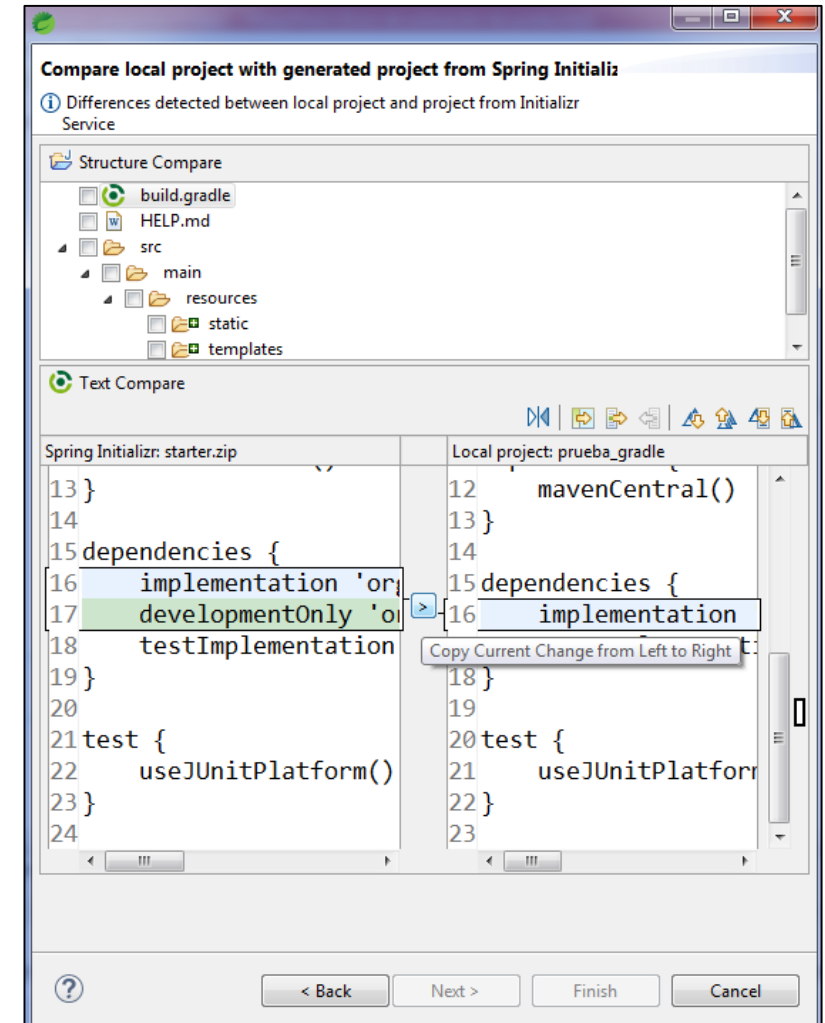
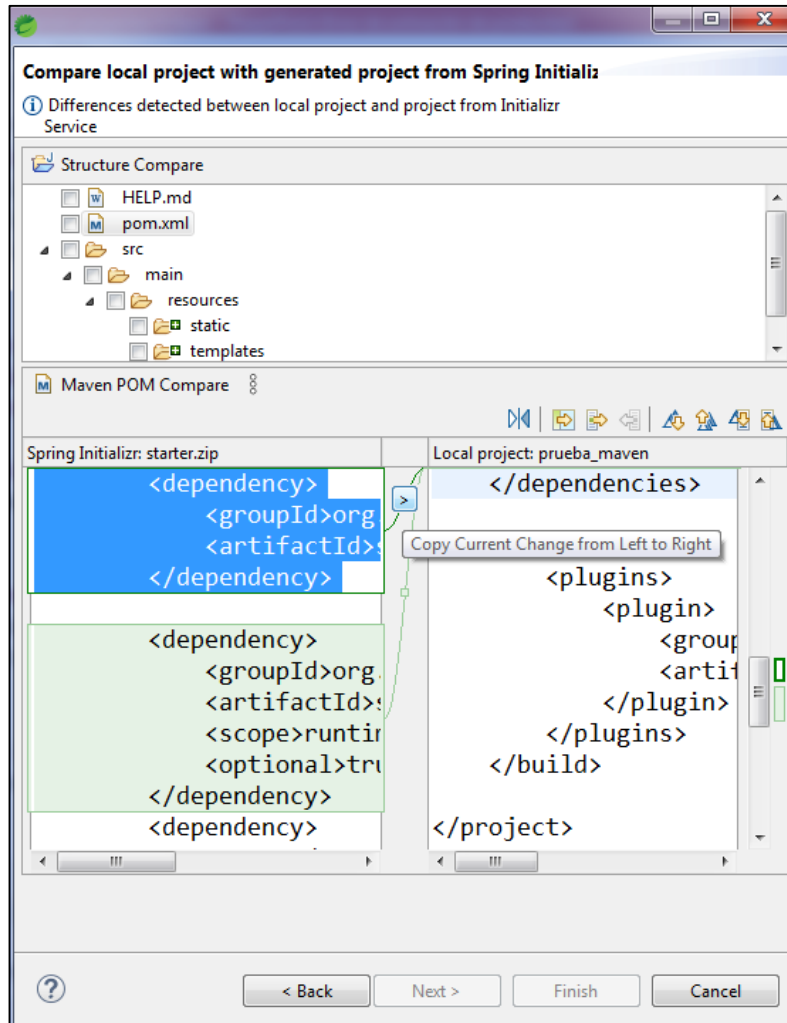
# 5. AGREGAR DEPENDENCIAS

**Paso 2.** Para agregar dependencias una vez ya creado el proyecto, debemos hacer click botón derecho y seleccionar la opción Spring/Add Starters. Seleccionamos las dependencias mínimas de Spring Web y Spring DevTools:



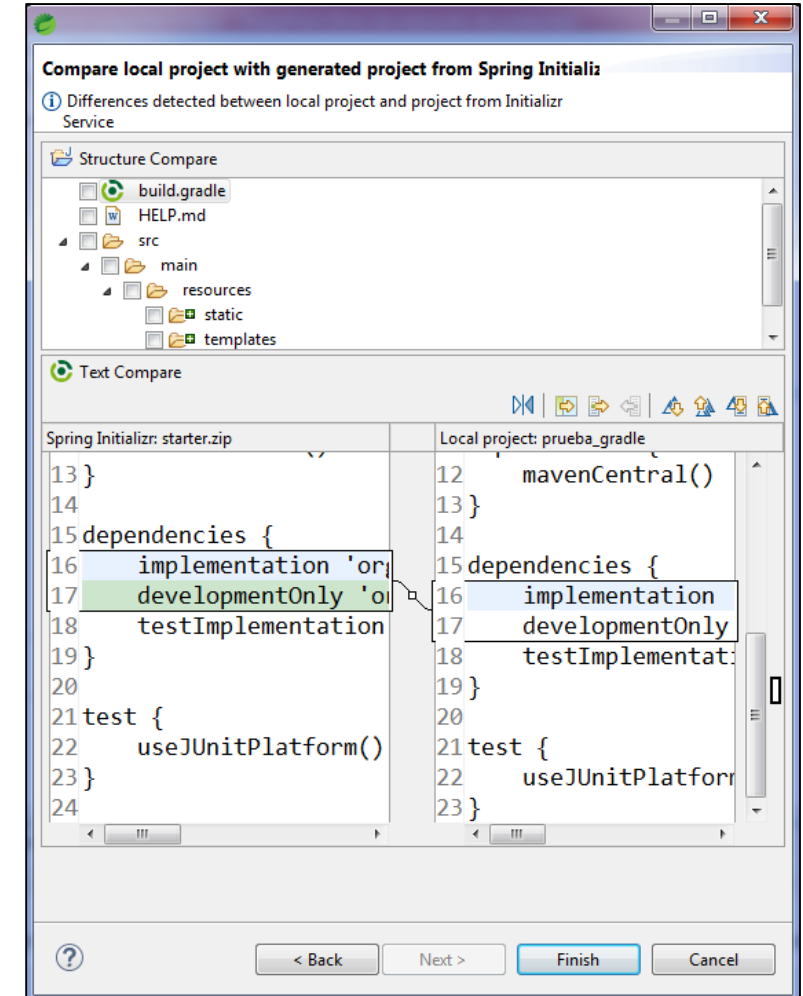
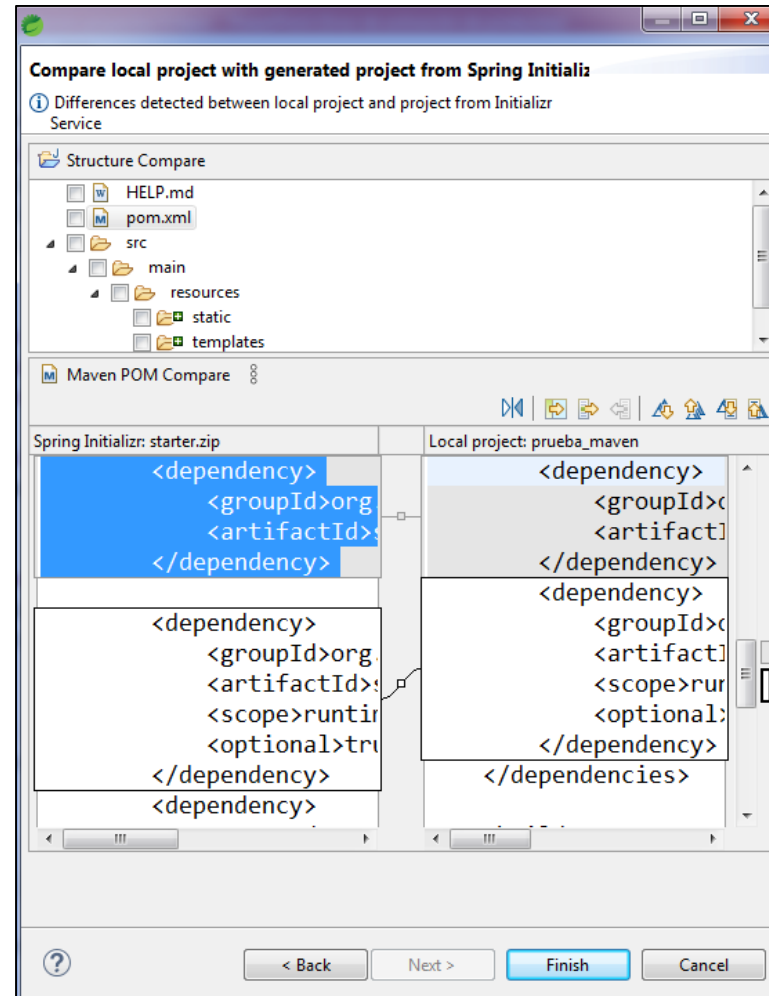
# 5. AGREGAR DEPENDENCIAS

**Paso 3.** En ambos tipos de proyectos (maven y gradle) se deben de agregar las dependencias haciendo click en un botón que las agrega gráficamente:



# 5. AGREGAR DEPENDENCIAS

**Paso 4.** Una vez hemos hecho click en el boton, se nos muestra como van a quedar modificados los ficheros pom.xml y build.gradle con las nuevas dependencias. Hacemos click en Finish:



# 5. AGREGAR DEPENDENCIAS

**Paso 5.** Vemos las dependencias agregadas en los archivos pom.xml y build.gradle:

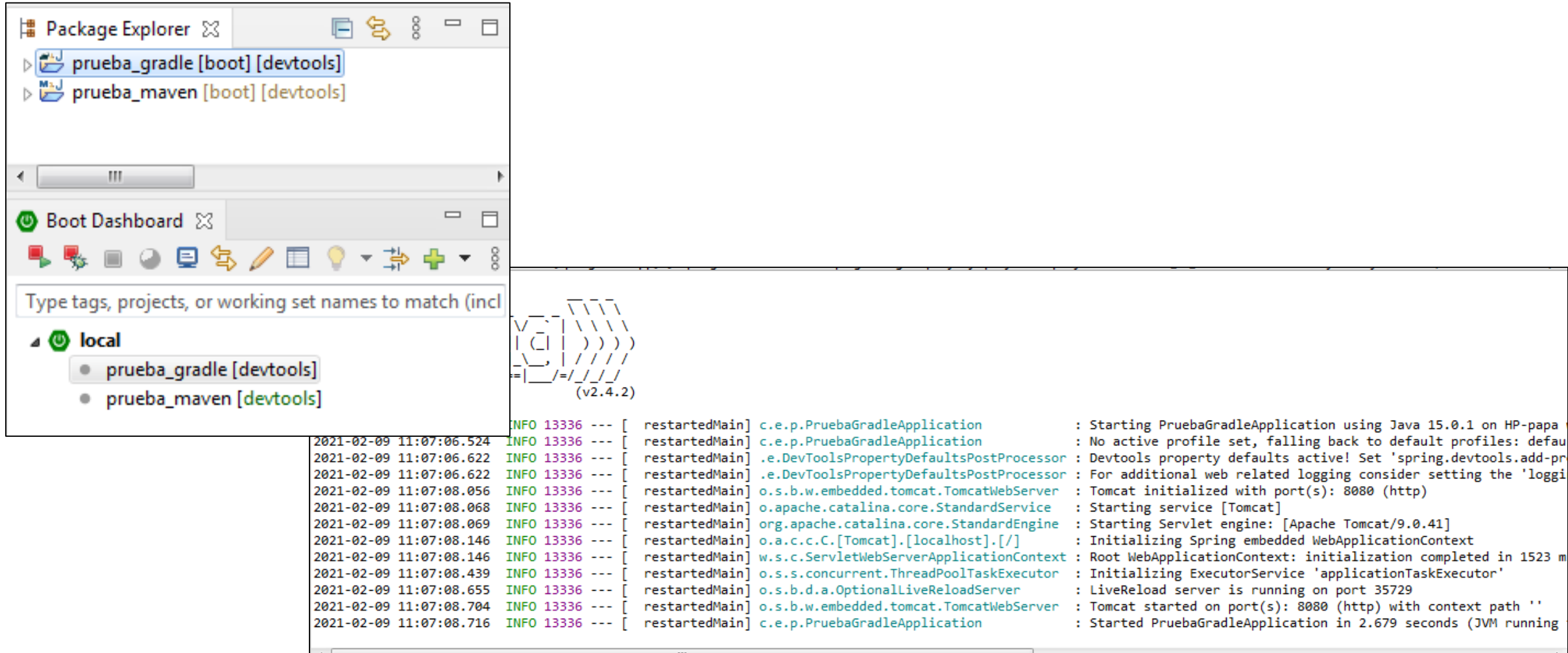
```
prueba_maven/pom.xml
24
25<dependency>
26    <groupId>org.springframework.boot</groupId>
27    <artifactId>spring-boot-starter-test</artifactId>
28    <scope>test</scope>
29</dependency>
30<dependency>
31    <groupId>org.springframework.boot</groupId>
32    <artifactId>spring-boot-starter-web</artifactId>
33</dependency>
34<dependency>
35    <groupId>org.springframework.boot</groupId>
36    <artifactId>spring-boot-devtools</artifactId>
37    <scope>runtime</scope>
38    <optional>true</optional>
39</dependency>
40</dependencies>
```

```
build.gradle
1|plugins {
2    id 'org.springframework.boot' version '2.4.5'
3    id 'io.spring.dependency-management' version '1.0.11.RELEASE'
4    id 'java'
5}
6
7group = 'com.example'
8version = '0.0.1-SNAPSHOT'
9sourceCompatibility = '11'
10
11repositories {
12    mavenCentral()
13}
14
15dependencies {
16    implementation 'org.springframework.boot:spring-boot-starter-web'
17    developmentOnly 'org.springframework.boot:spring-boot-devtools'
18    testImplementation 'org.springframework.boot:spring-boot-starter-test'
19}
20
21test {
22    useJUnitPlatform()
23}
```



# 5. AGREGAR DEPENDENCIAS

**Paso 6.** Podemos observar que ambos proyectos ya han agregado en su descripción [devtools] y ya podemos reiniciar nuestro servidor:



The screenshot displays the Eclipse IDE interface. The Package Explorer on the left shows two projects: 'prueba\_gradle [boot] [devtools]' and 'prueba\_maven [boot] [devtools]'. The Boot Dashboard below it lists 'prueba\_gradle [devtools]' and 'prueba\_maven [devtools]' under the 'local' environment. The console on the right shows a log of the application startup process, including messages about restarting the main application, initializing the Tomcat web server, and starting the Spring application context.

```
INFO 13336 --- [ restartedMain] c.e.p.PruebaGradleApplication : Starting PruebaGradleApplication using Java 15.0.1 on HP-papa
INFO 13336 --- [ restartedMain] c.e.p.PruebaGradleApplication : No active profile set, falling back to default profiles: defau
INFO 13336 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-pr
INFO 13336 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'loggi
INFO 13336 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
INFO 13336 --- [ restartedMain] org.apache.catalina.core.StandardService : Starting service [Tomcat]
INFO 13336 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Starting Servlet engine: [Apache Tomcat/9.0.41]
INFO 13336 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Initializing Spring embedded WebApplicationContext
INFO 13336 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor : Root WebApplicationContext: initialization completed in 1523 m
INFO 13336 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : Initializing ExecutorService 'applicationTaskExecutor'
INFO 13336 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : LiveReload server is running on port 35729
INFO 13336 --- [ restartedMain] c.e.p.PruebaGradleApplication : Tomcat started on port(s): 8080 (http) with context path ''
INFO 13336 --- [ restartedMain] c.e.p.PruebaGradleApplication : Started PruebaGradleApplication in 2.679 seconds (JVM running
```