API REST FULL con Spring BOOT

# Configurar Apache en Eclipe

1. Ahora pasan a descargar “Apache Tomcat 9” (el archivo Binary Distributions Core 32-bit o 64-bit – **zip**) esto de acuerdo a su PC aquí el link: <http://tomcat.apache.org/>
2. Luego de descargar el Tomcat crean una carpeta llamada “apache” en la unidad C:
3. en esta carpeta descomprimir el Archivo ZIP les quedara de la siguiente forma: C:/apache

# 

# Si están en Windows, Desde el menú de opciones Seleccionan Window -> Preferences

# les saldrá una ventana seleccionan Server-> Runtime Environments -> Clic en el botón Add

# agregar apache tomcat a eclipse

# En la nueva ventana que aparece seleccionan la carpeta “apache” y luego seleccionan ****Apache Tomcat v9.0****y botón Siguiente.

# Les saldrá otra ventana, en ella seleccionan browser y buscan la carpeta apache donde descomprimieron el zip seria C:/apache-tomcat (en mi caso) luego clic en el botón en Finish.

# 

# Ahora deben iniciar los servicios del servidor ****“Apache Tomcat”**** para poder hacer las aplicaciones WEB. Para Arrancar el Tomcat deben hacer clic derecho en la ventana Server si no aparece esta ventana pueden activarla sigan estos pasos: Desde el Menu de Opciones clic en Window->Show View->Other

# 

# Aqui les saldrá otra ventana seleccionan la carpeta “Server” luego clic en Servers y finalmente clic botón Open

# 

# Aparecera un mensaje

# 

# Ahora hacer clic en el botón derecho del ratón aparecerá new server -> seleccionar Tomcat v8.0 Server y clic en el botón finish.

# 

# 

Luego de hacer esto ya tendrán correctamente configurado el apache tomcat en Eclipse. Ppor ultimo les quedara de esta manera

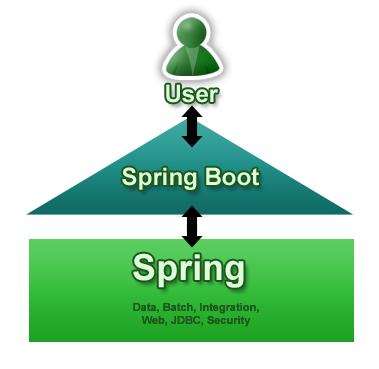
# 

# CÓMO CREAR UNA APLICACIÓN WEB HELLO WORLD SPRING BOOT EN ECLIPSE USANDO MAVEN.

En esta publicación de blog, explicaré cómo puede crear una aplicación Hello World Spring Boot en Eclipse utilizando Maven. Antes de profundizar en los detalles, permítanme dar una descripción rápida de Spring Boot.

#### ¿Qué es Spring Boot?

Spring Boot, apareció hace unos años con la idea de complementar a Spring, es decir, dar un añadido a Spring, no con la idea de substituirlo. Tal y como se puede interpretar en la imagen inferior, realmente Spring Boot trabaja por debajo con Spring, pero de una manera más fácil.



Podríamos definir Spring Boot como **un acelerador para la creación de proyectos de Spring**, que se basa principalmente en el concepto de convención antes de configuración. Otra manera de definir Spring Boot podría ser como un acelerador de creación de proyectos de Spring. Por tanto, la finalidad, es crear proyectos de Spring, pero de una manera más ágil, mediante a una serie de convenciones que prevalecen sobre la configuración. Ahorrándonos el tener que perder tiempo en realizar configuraciones pesadas en ficheros.

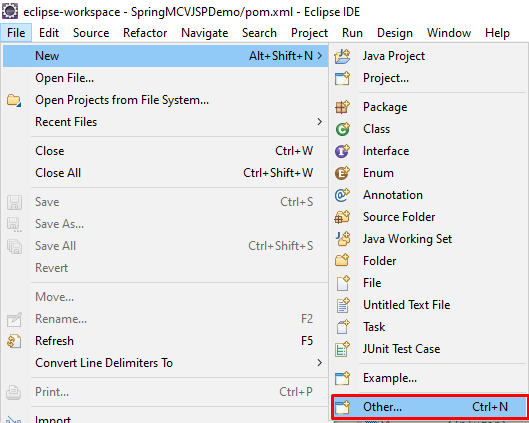
**Automáticamente** realiza la **configuración de la aplicación** por usted asumiendo algunos valores predeterminados. Estos valores predeterminados son suficientes para comenzar la mayoría de las veces, pero puede cambiar fácilmente estos valores predeterminados si es necesario. Esto le permite tener una aplicación de resorte en funcionamiento muy rápidamente sin tener que molestarse con las cosas de la placa de la caldera. Todo lo que tiene que hacer es decirle a Spring boot el tipo de aplicación que está creando. Proporciona servidores HTTP integrados como Tomcat, Jetty, etc., lo que facilita el desarrollo de aplicaciones web.

Hay que destacar que Spring Boot NO se considera como tal un Framework, en cambio, Spring si que tiene el privilegio de categorizarse como tal. Con Spring Boot, podremos hacer lo mismo que con Spring. Pero de una forma más rápida que si desarrollamos exclusivamente con Spring únicamente.

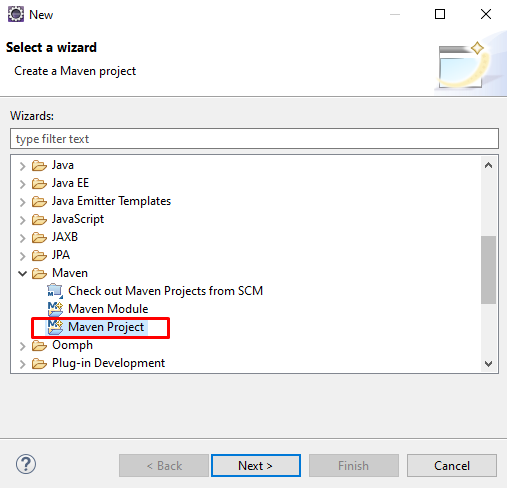
#### Pasos a seguir para crear una aplicación de arranque Hello World Spring en Eclipse usando Maven

##### **Creando Proyecto Maven en Eclipse**

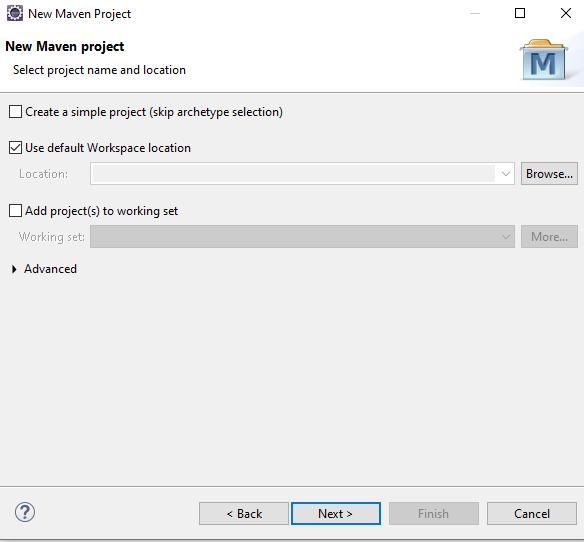
**Paso 1 - Archivo -> Nuevo -> Otro**



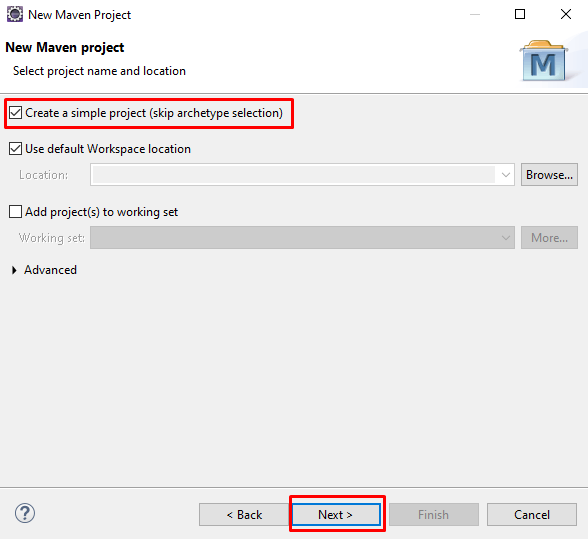
**Paso 2: selecciona Proyecto Maven.**

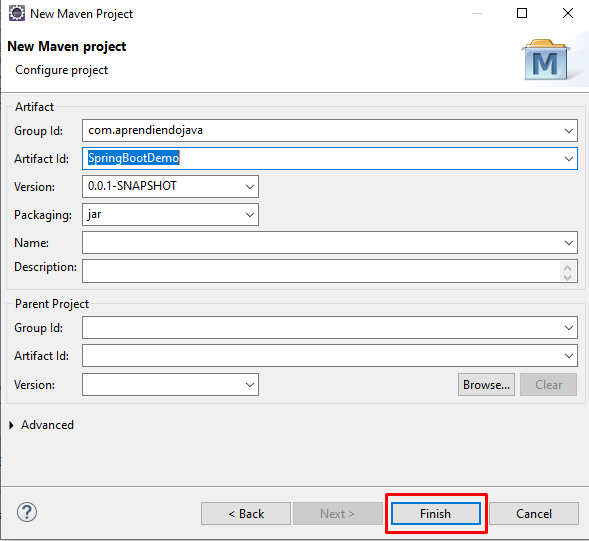


**Paso 3 - Haz clic en Siguiente.**

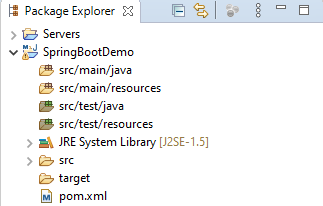


**Paso 4: haz clic en Crear un proyecto simple y haz clic en Siguiente**



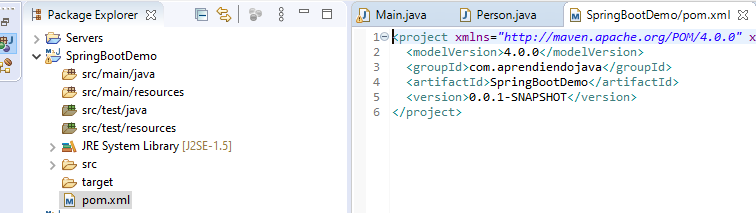
**Paso 5 - Ingrese la identificación del grupo y la identificación del artefacto** 

**Paso 5 - Haz clic en Finalizar. Tu proyecto Maven ya está listo**



##### **Agregar dependencias de arranque de Spring**

**Paso 6 - Abra pom.xml**



**Paso 7: agregue las dependencias de Spring Boot (agregue el siguiente texto):**

<parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>2.1.2.RELEASE</version>

    </parent>

    <dependencies>

    <dependency>

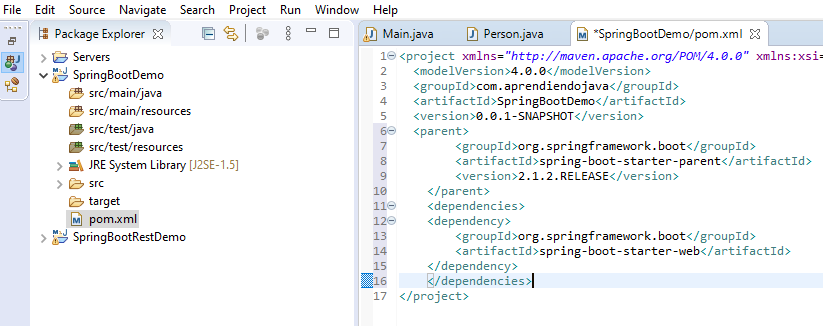
        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-web</artifactId>

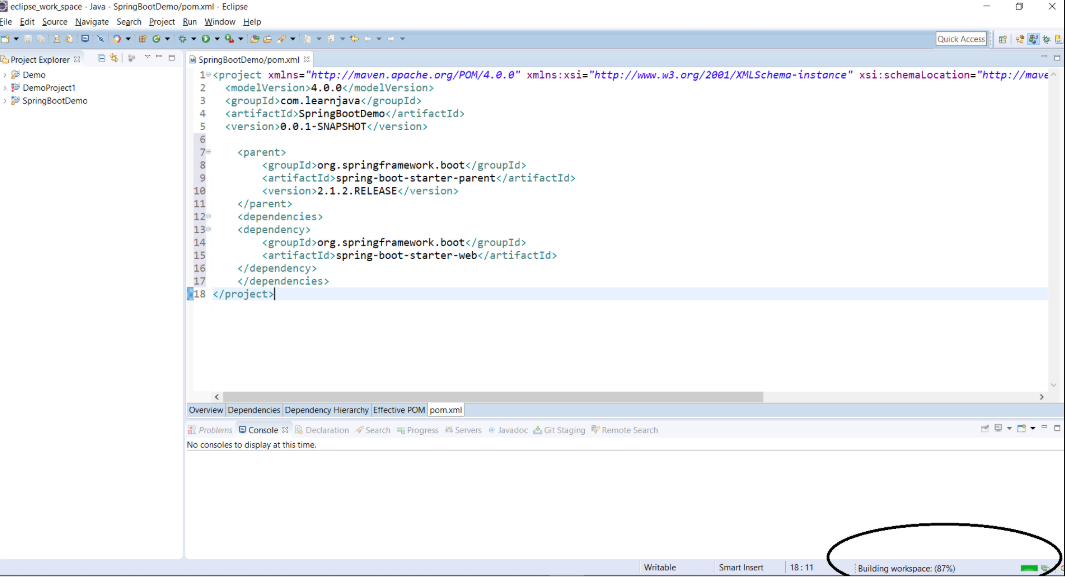
    </dependency>

    </dependencies>

Spring Boot proporciona una serie de 'iniciadores' que agregan los archivos jar necesarios al classpath. Aquí estamos utilizando los siguientes iniciadores:  
**spring-boot-starter-parent** - Esto proporciona valores predeterminados útiles de Maven.  
**spring-boot-starter-web** : esto le dice a Spring Boot que estamos desarrollando una aplicación web. Entonces SpringBoot quedará asi

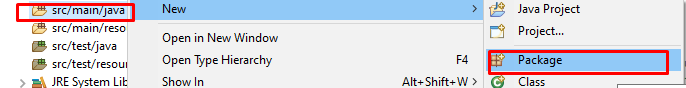


**Paso 8 - Guardar (Esto descargará los archivos jar necesarios)**

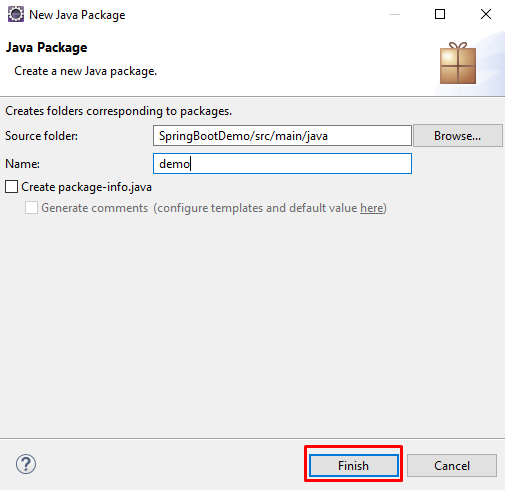


##### **Crear clase para el código de arranque de Spring Boot**

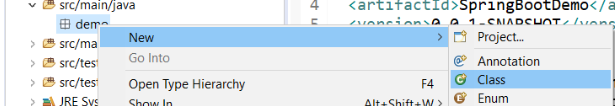
**Paso 9 - Crea un nuevo paquete (Haz clic derecho en src / main / java -> Nuevo -> paquete**



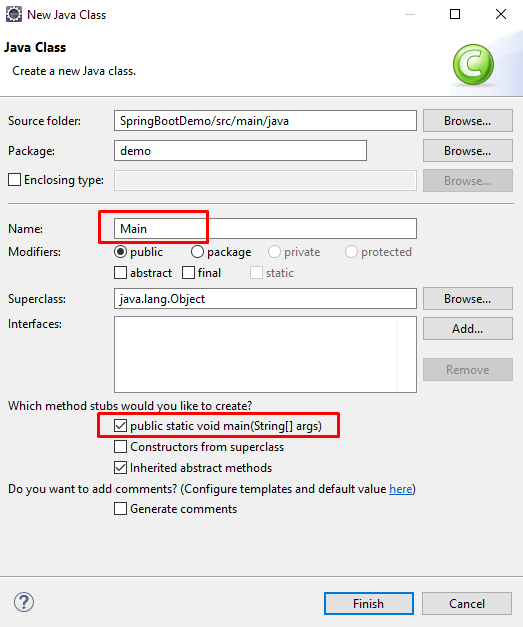
**Paso 10: ingrese el nombre del paquete como demostración y haga clic en Finalizar**



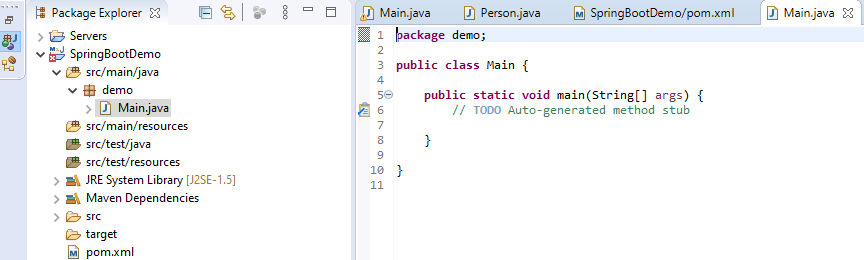
**Paso 11: Cree una nueva clase (haga clic derecho en el paquete de demostración -> Nuevo -> Clase**



**Paso 12 - Ingrese el nombre de la clase como Principal. Marque la casilla para crear un método Principal.**



**Paso 13 - Haz clic en Finalizar. Esto crea Main.java de la siguiente manera:**



##### **Escribe el código de Spring Boot**

**Paso 14: agrega el siguiente código en Main.java:**

package demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.EnableAutoConfiguration;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

@EnableAutoConfiguration

public class Main {

    @RequestMapping("/")

    String home() {

        return "Hello World!";

    }

      public static void main(String[] args) {

            SpringApplication.run(Main.class, args);

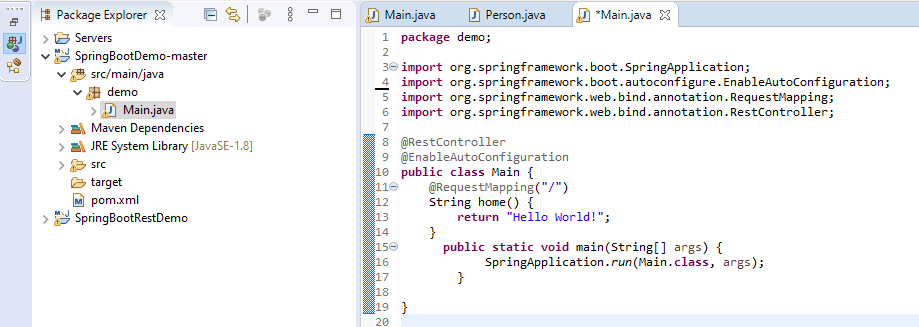
        }

}

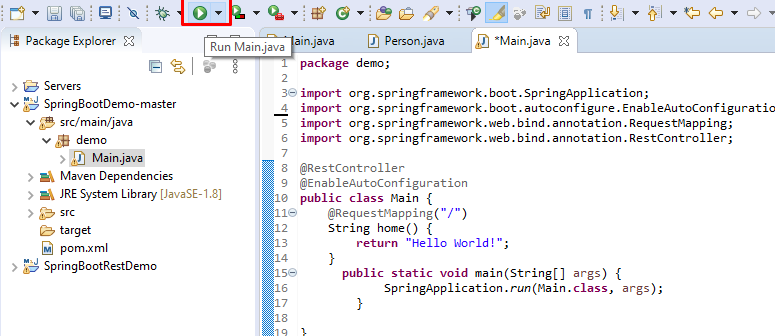
@[RestController](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/RestController.html) annotation 🡪 designa la clase como una clase **Controladora** para manejar las solicitudes entrantes.

@[EnableAutoConfiguration](https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/EnableAutoConfiguration.html) 🡪 está anotación le pide a Spring Boot que adivine y agregue los **beans** necesarios de acuerdo con el archivo de configuración.

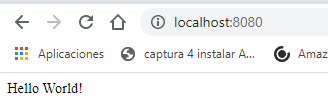
@[RequestMapping](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/RequestMapping.html) 🡪esto significa que cualquier solicitud a la página de inicio (designada por la URL "/") se asignará al método llamado "home".



**Paso 15: ejecuta el código**



**Paso 16 - Verifica en el navegador. Abra una ventana del navegador y escriba localhost: 8080**



¡Entonces tiene su aplicación Spring Boot en funcionamiento!

CÓMO CREAR EL SERVICIO SPRINGBOOT REST Y PROBARLO A TRAVÉS DE POSTMAN

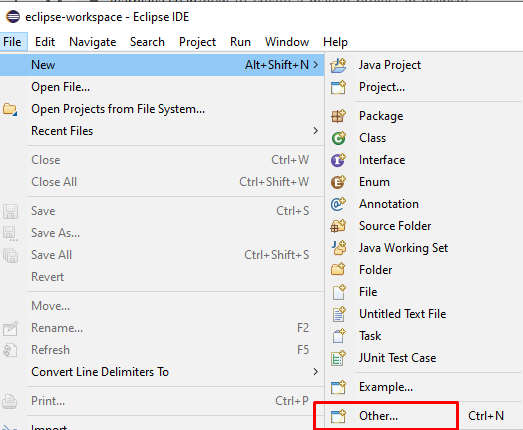
En esta publicación de blog, explicaré cómo puede crear un servicio REST utilizando Spring Boot.

**Paso 1: crea un nuevo proyecto Maven**  llámale **SpringBootRestDemo**

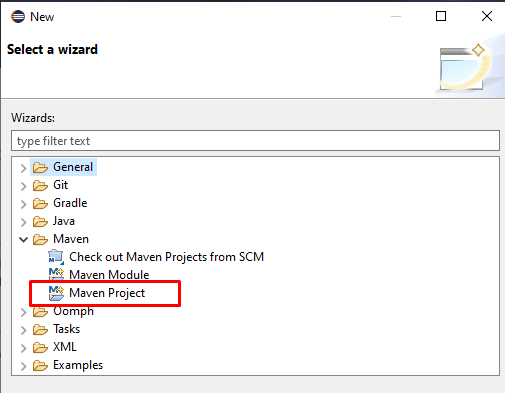
# CÓMO CREAR UN PROYECTO MAVEN EN ECLIPSE

En esta publicación de blog, explicaré cómo crear un proyecto Maven en Eclipse:

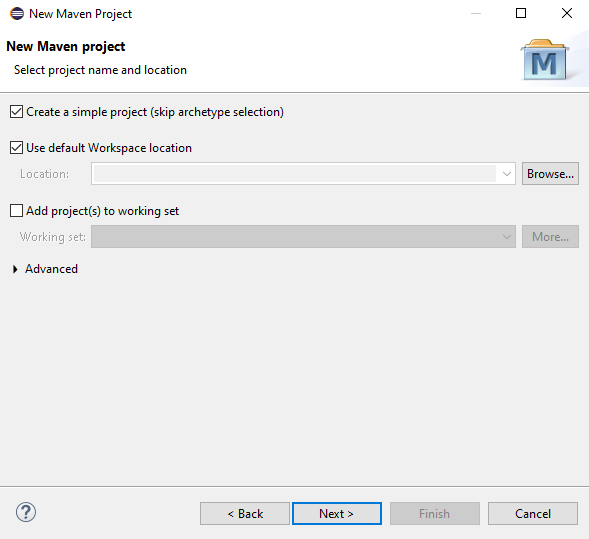
**Paso 1.1 - Abra Eclipse. Haga clic en Archivo -> Nuevo -> Otro**



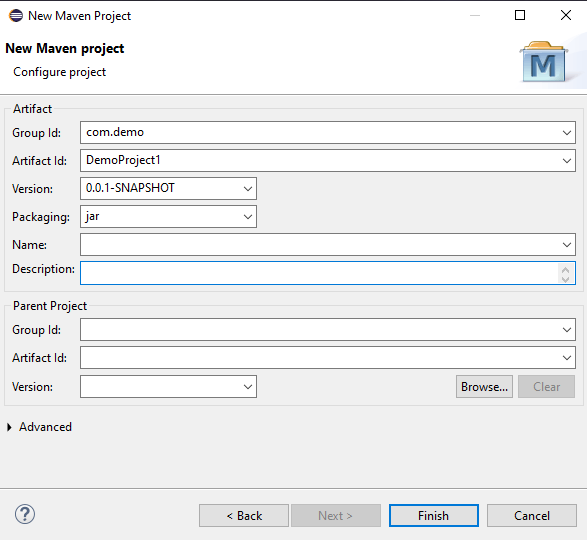
**Paso 1.2 - Haga clic en Proyecto Maven debajo de Maven:**



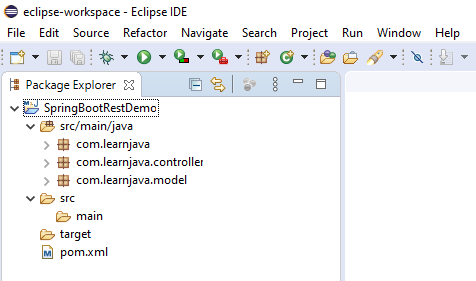
**Paso 1.3 - Haga clic en Crear un proyecto simple (Saltar selección de tipo de archivo). Haga clic en Siguiente:**



**Paso 1.4 Ingrese el ID del grupo y el ID del artefacto en la pantalla:**



**Paso 1.5 - Haga clic en Finalizar. Tu proyecto Maven está listo. Puede comenzar a agregar dependencias en su archivo pom y comenzar a escribir código Java en la carpeta src / main / java:**



**Paso 2: agregue las siguientes dependencias de SpringBoot al archivo POM y guárdelas.**

<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>com.learnjava</groupId>

  <artifactId>SpringBootRestDemo</artifactId>

  <version>0.0.1-SNAPSHOT</version>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>2.1.2.RELEASE</version>

    </parent>

    <dependencies>

    <dependency>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-web</artifactId>

    </dependency>

    </dependencies>

</project>

Escribir código

**Paso 3: crea la clase Persona de la siguiente manera**

package com.learnjava.model;

public class Person {

    private int id;

    private String name;

    private int age;

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

    public Person(){

    }

    public Person(int id,String name, int age) {

        super();

        this.id=id;

        this.name = name;

        this.age = age;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public int getAge() {

        return age;

    }

    public void setAge(int age) {

        this.age = age;

    }

}

Clase de persona

Esta es la clase de modelo. Representa a una persona. Tiene 3 campos, id, nombre y edad.

**Paso 4: cree una clase PersonController de la siguiente manera:**

package com.learnjava.controller;

import java.util.ArrayList;

import java.util.List;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.learnjava.model.Person;

@RestController

public class PersonController {

    @RequestMapping("/")

    String home() {

        return "Hello World!";

    }

    @GetMapping("/person")

    public List<Person> getAllPersons(){

        //Returns hardcoded data, a real world application would return from the database

        List<Person> personList = new ArrayList<Person>();

        personList.add(new Person(1,"Mickey Mouse", 30));

        personList.add(new Person(2,"Donald Duck", 35));

        personList.add(new Person(3,"Peppa Pig", 15));

        return personList;

    }

    @GetMapping("/person/{personId}")

    public Person getPersonWithId(@PathVariable Integer personId){

        //Returns hardcoded data, a real world application would return from the database

        return new Person(3,"Peppa Pig", 15);

    }

Clase PersonController

Esta es una clase de controlador. Especifica el método que manejará las solicitudes del cliente. Tiene los siguientes métodos.

Métodos

**getAllPersons**

Esto devuelve todos los objetos Persona. Una aplicación del mundo real consultaría una base de datos y devolvería los objetos Persona. Este código simplemente devuelve una lista de objetos Persona.

**getPersonWithId**  
Esto devuelve un objeto Person correspondiente a la identificación especificada. Una aplicación del mundo real consultaría una base de datos y devolvería los objetos Persona correspondientes a la identificación especificada. Este código simplemente devuelve un objeto Person codificado

**addPerson**

Esto guarda el objeto Person que se pasa. Una aplicación del mundo real guardaría el objeto Person en la base de datos. Este código simplemente ejecuta una declaración Sysout.

Anotaciones

**@RestController anotación**

Esto se especifica a nivel de clase. Esto designa la clase como un controlador REST, es decir, un controlador que maneja las solicitudes REST. Incluye internamente la anotación @ResponseBody que convierte el valor de Java devuelto por el método en JSON. Esta anotación se agregó desde Spring 4.0. Es una anotación de conveniencia que combina las anotaciones @Controller y @ResponseBody.

**Anotación @GetMapping**

Esta anotación especifica la URL a la que se asigna este método y también especifica que es una solicitud HTTP GET. Es una anotación compuesta que actúa como un acceso directo para @RequestMapping (method = RequestMethod.GET). Esto se especifica para los métodos getAllPersons y getPersonWithId.

**Anotación @PostMapping**

Esta anotación especifica la URL a la que se asigna este método y también especifica que es una solicitud HTTP POST. Es una anotación compuesta que actúa como un acceso directo para @RequestMapping (method = RequestMethod.POST). Esto se especifica en el método addPerson

**@RequestBody Anotación**

Esta anotación debe especificarse cuando se pasa un objeto a un método. Deserializa automáticamente el objeto JSON en un objeto Java.

**Paso 5 - Finalmente, crea un Main.java de la siguiente manera:**

package com.learnjava;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Main {

      public static void main(String[] args) {

            SpringApplication.run(Main.class, args);

        }

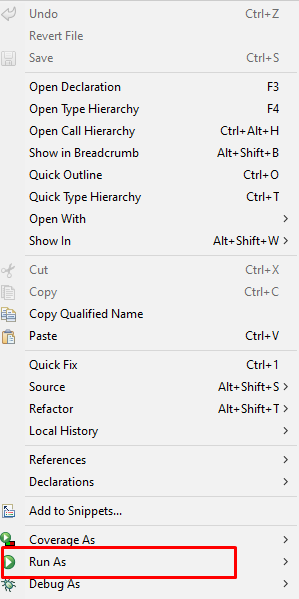
}

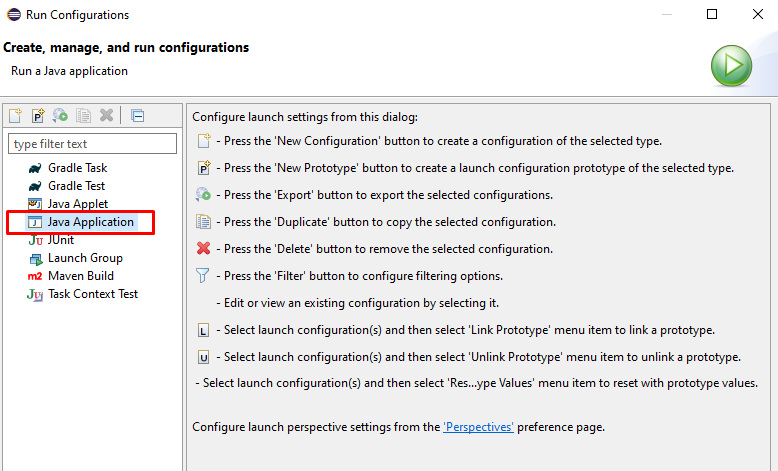
Principal

Este es el punto de partida de la aplicación.

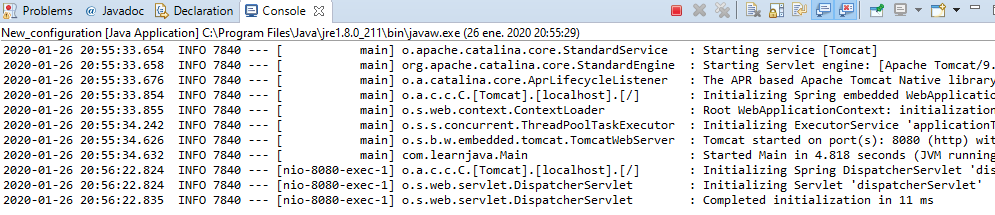
Ejecutando la aplicación

**Paso 6 - Ejecute Main.java como una aplicación Java de la siguiente manera:**





Esto mostrará lo siguiente en la consola:



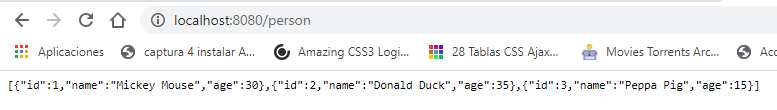
Prueba a través del navegador

Consigue todas las personas

Abra una ventana del navegador y escriba la siguiente URL:

<http://localhost:8080/person>

Esto mostrará lo siguiente:

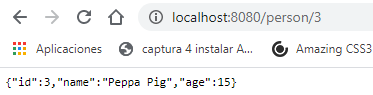


Obtener persona por id

Abra una ventana del navegador y escriba la siguiente URL:

<http://localhost:8080/person/3>

Esto mostrará lo siguiente:



Agregar persona

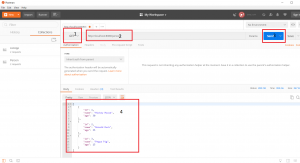
Esto no se puede probar en un navegador, ya que es una solicitud POST

Prueba vía Postman

Postman es una aplicación del lado del cliente que le permite probar los servicios REST fácilmente. Puede descargar Postman desde Postman como [aquí](https://www.getpostman.com/downloads/) . Una vez completada la descarga, puede hacer clic en el acceso directo del escritorio para iniciar Postman.

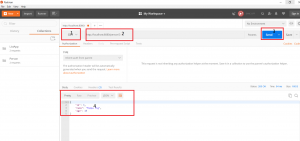
Consigue todas las personas

1. Seleccione el tipo de solicitud como "OBTENER"
2. Ingrese la URL como "http: // localhost: 8080 / person /"
3. Haga clic en "Enviar".
4. Esto mostrará la respuesta "JSON" enviada por el servidor en el cuerpo a continuación de la siguiente manera:

[](https://learnjava.co.in/wp-content/uploads/2019/05/7.png)

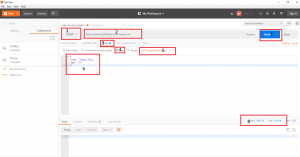
Obtener persona por id

1. Seleccione el tipo de solicitud como "OBTENER"
2. Ingrese la URL como "http: // localhost: 8080 / person / 3",
3. Haga clic en "Enviar".
4. Esto mostrará la respuesta "JSON" enviada por el servidor en el cuerpo a continuación de la siguiente manera:

[](https://learnjava.co.in/wp-content/uploads/2019/05/8.png)

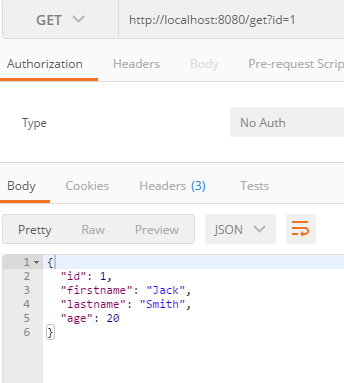
Agregar persona

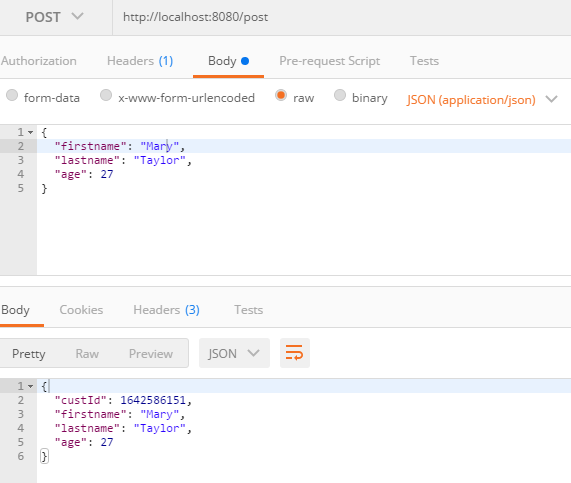
1. Seleccione el tipo de solicitud como "POST"
2. Ingrese la URL como "http: // localhost: 8080 / person / newPerson".
3. Click Body
4. Seleccione "Raw"
5. Escriba como "JSON (Aplicación / JSON"),
6. Ingrese los datos que se guardarán en formato JSON como se muestra a continuación.
7. Haga clic en "Enviar".
8. Esto mostrará el estado HTTP 200 enviado por el servidor.

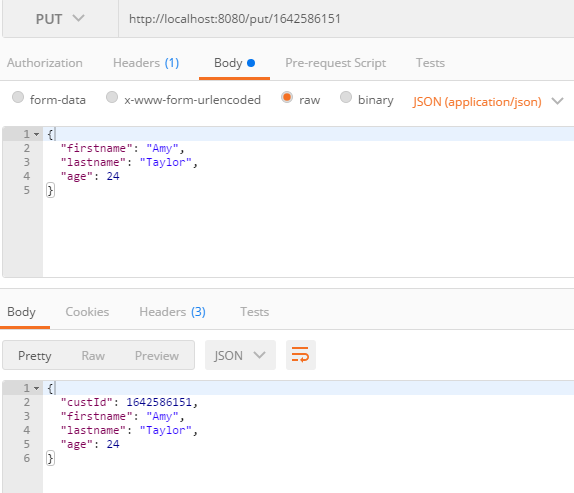
[](https://learnjava.co.in/wp-content/uploads/2019/05/9.png)

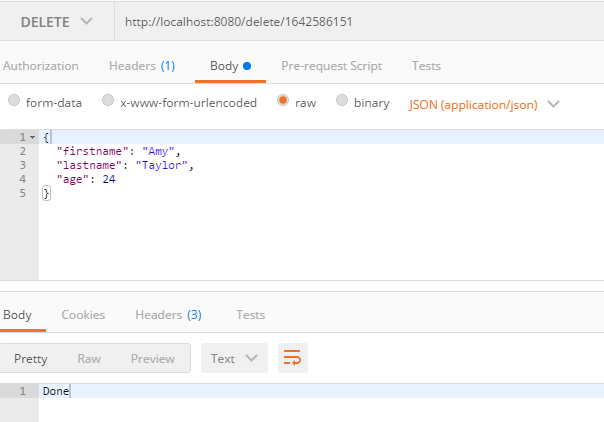
###### 4. Run and check result

Build & Run SpringBoot project then check result:

**– Get RequestMapping**:  


**– POST RequestMapping**:  


**– PUT RequestMapping**:  


**– DELETE RequestMapping**:  


Puede descargar el código fuente de este proyecto a través del repositorio de Github [aquí](https://github.com/learnjavawithreshma/SpringBootRestDemo) .

# [Creando un proyecto de Spring Boot](http://javadesde0.com/creando-un-proyecto-de-spring-boot/)

Para crear un proyecto de Spring, tenemos **varias opciones**. Las **2 más populares** son:

* Crear un proyecto desde **Spring Boot Initializer** la herramienta web que nos proporciona Pivotal.
* Crear el proyecto desde el **IDE mediante al Command Line Tool** (CLI), en nuestro caso el IDE elegido será Spring Tool Suite 4.

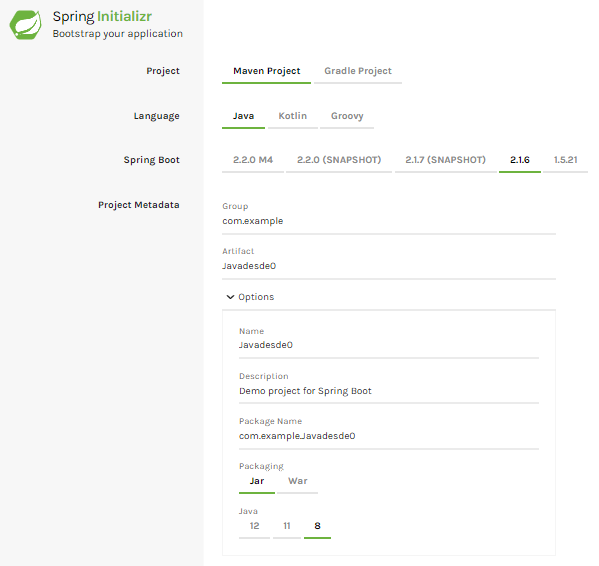
En mi opinión, la más sencilla es desde el propio Spring Tool Suite 4, ya que nos carga el proyecto automáticamente en el explorador de proyectos del IDE. Y así, ya podemos trabajar directamente con él.

Creando un proyecto de Spring mediante a **Spring Boot Initializer**

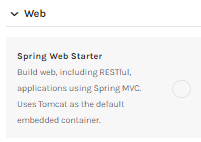
Para crear un proyecto desde la web, visitaremos la web [**https://start.spring.io/**](https://start.spring.io/)

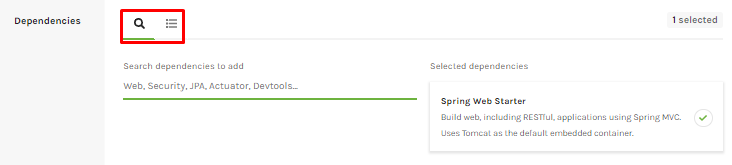
La configuración que utilizaremos será mediante a Maven, utilizando Java, y Spring Boot 2.1.6 (quizás esta versión cambie estés realizando este tutorial).

Si desplegamos el Options, podemos ver que podemos seleccionar la versión de Java. Nosotros utilizaremos la 8.



Si bajamos un poco más, podemos ver el apartado de **dependencias**. En nuestro caso, vamos a añadir **spring Web**. La podemos buscar desde el buscador, o bien ir al icono de alado, donde nos aparecerá un listado con dependencias y seleccionarla.





Y finalmente, pulsamos en generar proyecto.

http://javadesde0.com/wp-content/uploads/Screenshot_3-3.png

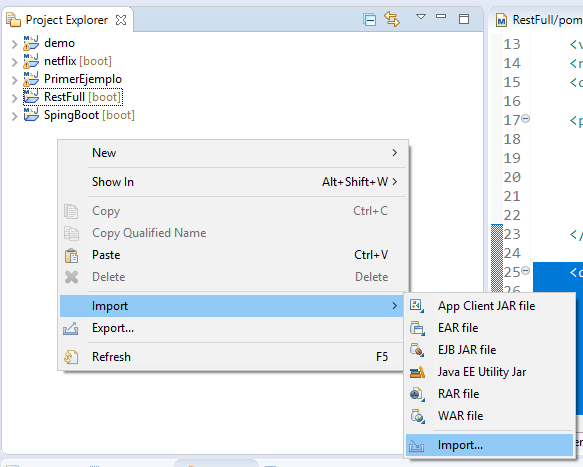
Finalmente, se nos descargará el proyecto en archivo .zip,

http://javadesde0.com/wp-content/uploads/Screenshot_7-2.png

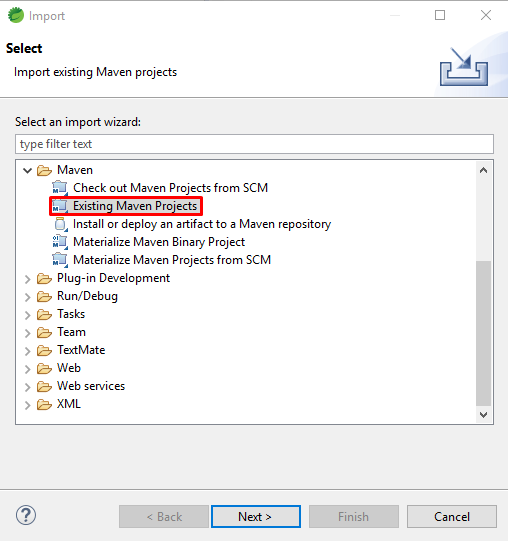
El cual vamos a descomprimir en el mismo directorio de descargas donde se nos ha descargado el proyecto

### ****Cargando un proyecto de Spring Boot****

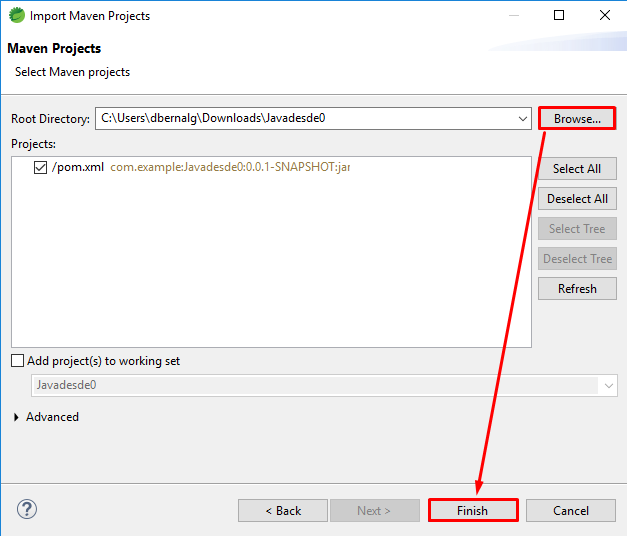
Ahora, vamos a cargar un proyecto de maven, esto nos vale para cualquier proyecto, no solamente para los que creamos desde [**https://start.spring.io/**](https://start.spring.io/) . Para realizar la carga, vamos a Project Explorer > Botón derecho > Import…



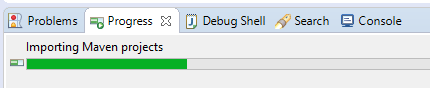
Seleccionamos la opción de proyecto de Maven existente:



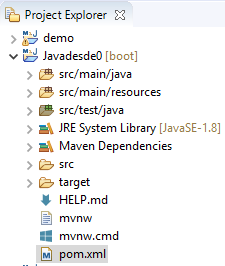
Seleccionamos el directorio donde se encuentra (si habéis seguido las indicaciones estará en descargas).



Y una vez seleccionado, podemos ver como se realiza la importación y la descarga de dependencias.

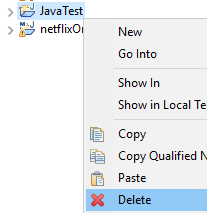


Finalmente, y una vez concluida la descarga de dependencias, ya tenemos nuestro proyecto cargado en Spring Tool Suite con nuestro fichero pom.xml



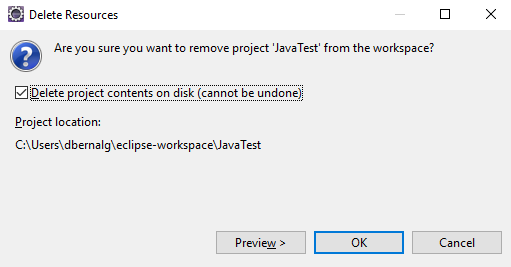
**Creando un proyecto Maven mediante a Spring Tool Suite 4**

Anteriormente si habéis seguido el tutorial paso a paso hasta aquí, hemos creado un proyecto, por lo que no podemos repetir el mismo nombre. Ya que nos generaría un error. Llegados a este punto, tenemos dos opciones, eliminar el proyecto o ponerle otro nombre diferente. En mi caso, voy a borrar el proyecto.



### ****Eliminando un proyecto de Spring Boot****

Si no marcamos el delete project contents on disk, realmente estamos únicamente eliminando el proyecto del explorador de proyectos del IDE. Pero aún existiría el directorio.



Bien, tras ver como realizamos un proyecto desde la web de[**https://start.spring.io/**](https://start.spring.io/) vamos a ver cómo realizarlo con **Spring Tool Suite**.

Esta manera, para mí es la más rápida y sencilla, aunque mejor que decirlo por aquí es que lo comprobareis vosotros mismos.

## Instalación Spring Tool Suite 4 (la distribución basada en Eclipse)

### Paquete de distribución Spring Tools 4 listo para usar

Para descargar la distribución completa de Spring Tool Suite 4 que tiene todo preinstalado, vaya a:

* [Spring Tools 4 Página principal](https://spring.io/tools)

**Nota:** Para Windows, la descarga es un archivo JAR autoextraíble. Haga doble clic y se expandirá en el disco en la misma carpeta donde te has descargado el .jar

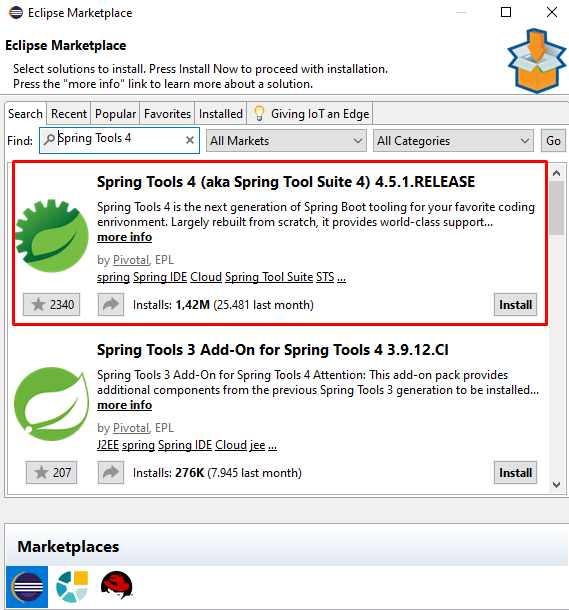
.

### Instale Spring Tools 4 en un Eclipse existente (a través del Marketplace)

Para instalar Spring Tools 4 en una instalación existente de Eclipse, necesita la **última versión de Eclipse** como requisito previo (descargue el IDE de Eclipse desde aquí: <https://www.eclipse.org/downloads/packages> ).

Puede instalar Spring Tools 4 en su instalación existente de Eclipse utilizando Eclipse Marketplace. Busque "Spring Tools 4" e instale:

# 

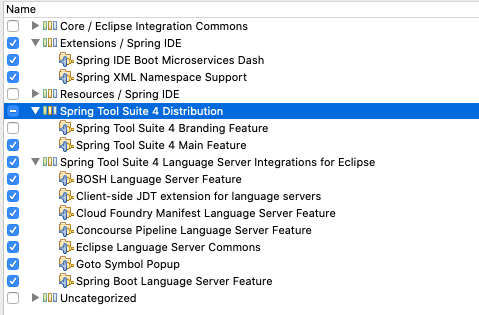


### Instale Spring Tools 4 en un Eclipse existente (a través de un repositorio p2)

También puede usar un repositorio p2 para instalar componentes individuales de Spring Tools 4 en una instalación existente de Eclipse. Elija Install New Softwareen Eclipse y use esta URL del repositorio p2:

* Eclipse 4.14: <https://download.springsource.com/release/TOOLS/sts4/update/e4.14/>
* Eclipse 4.13: <https://download.springsource.com/release/TOOLS/sts4/update/e4.13/>
* Eclipse 4.12: <https://download.springsource.com/release/TOOLS/sts4/update/e4.12/>

Seleccione las características principales que desea instalar y continúe. El conjunto estándar de características es este:



Para la referencia, las ID de funciones son:

* org.springframework.boot.ide.main.feature
* org.springframework.tooling.boot.ls.feature
* org.springframework.tooling.cloudfoundry.manifest.ls.feature
* org.springframework.tooling.bosh.ls.feature
* org.springframework.tooling.concourse.ls.feature
* org.springframework.ide.eclipse.boot.dash.feature
* org.springframework.ide.eclipse.xml.namespaces.feature

Además de esos repositorios p2, también proporcionamos uno adicional:

* <https://download.springsource.com/release/TOOLS/sts4/update/latest/>

Esto le proporcionará las últimas Spring Tools 4 junto con la última versión de Eclipse y puede usarse para estar siempre al tanto de las últimas Spring Tools 4 y la última versión de Eclipse. Como combina Spring Tools 4 y Eclipse, se recomienda no instalar manualmente Spring Tools 4.

## Visual Studio Code

Descargue e instale Visual Studio Code:

* [Visual Studio Code](https://code.visualstudio.com/)

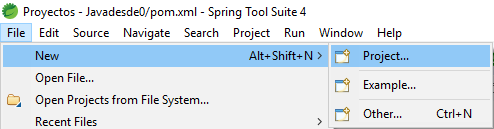
Instalar desde Visual Studio Marketplace:

* [Paquete de extensión de Java](https://marketplace.visualstudio.com/items?itemName=vscjava.vscode-java-pack)
* [Paquete de extensión de arranque de primavera](https://marketplace.visualstudio.com/items?itemName=Pivotal.vscode-boot-dev-pack)

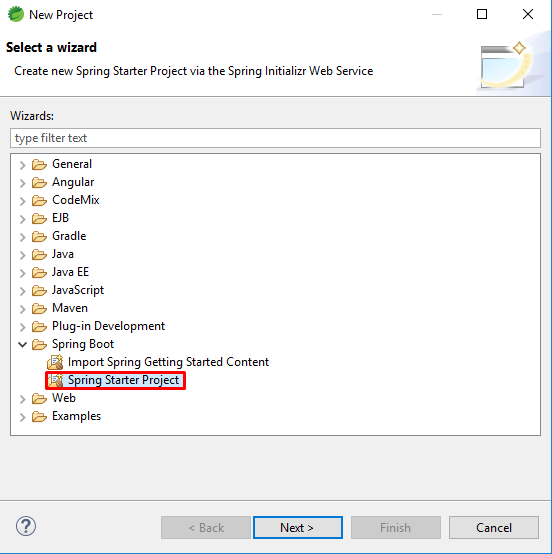
Puede encontrar instrucciones más detalladas sobre cómo instalar la extensión desde Visual Studio Marketplace [aquí](https://code.visualstudio.com/docs/editor/extension-gallery)

Una vez instalado vamos a **crear un proyecto:**

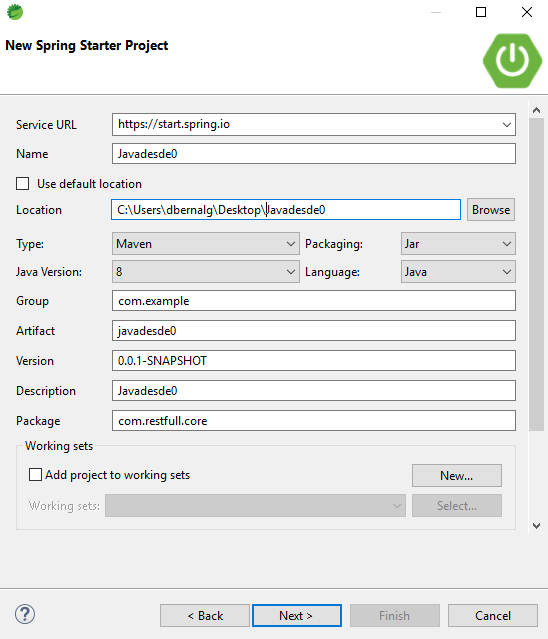
Abrimos nuestro IDE y vamos a File > New > Project



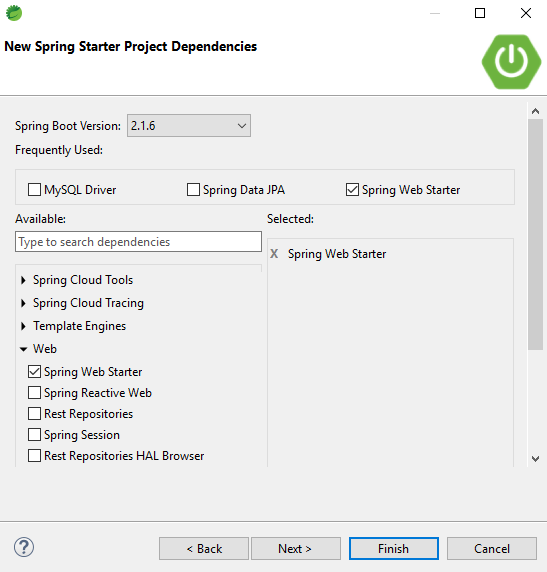
Seleccionamos Spring Starter Project



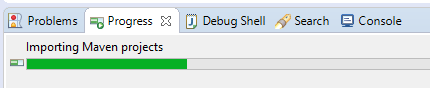
Y añadimos la información del proyecto:



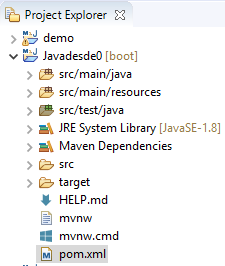
Y finalmente, añadimos las dependencias y pulsamos sobre finalizar.



Una vez finalizada la creación, como en la creación anterior, realizará la importación y descargará dependencias.



Y ya tenemos el proyecto listo para trabajar sobre el:



## ****Pom resultante de ambos proyectos****

Si observamos el pom que se nos ha creado en el proyecto, destaca principalmente el ver la versión de Java con al que estamos trabajando, en nuestro caso la 1.8.

También podemos apreciar que pese a que hemos marcado una sola dependencia, tenemos dos. Esto es debido a que automáticamente, al incluir la dependencia spring-boot-starter-web , también se nos incluye la dependencia de testing (spring-boot-starter-test).

<?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>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>2.1.6.RELEASE</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <groupId>com.example</groupId>

    <artifactId>javadesde0</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>Javadesde0</name>

    <description>Javadesde0</description>

    <properties>

        <java.version>1.8</java.version>

    </properties>

    <dependencies>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-test</artifactId>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

        </plugins>

    </build>

</project>