

Tabla de Contenidos

- [Spring Boot](#)
- [String Boot Starter y simplificaciones](#)
- [Otros artículos relacionados](#)

Spring Boot es una de las tecnologías dentro del mundo de Spring de las que más se está hablando últimamente. ¿Qué es y cómo funciona Spring Boot? . Para entender el concepto primero debemos reflexionar sobre cómo construimos aplicaciones con Spring Framework



- 1 seleccionar jars con maven
- 2 crear la aplicación
- 3 desplegar en servidor

Fundamentalmente existen tres pasos a realizar . El primero es crear un proyecto Maven/Gradle y descargar las dependencias necesarias. En segundo lugar desarrollamos la aplicación y en tercer lugar la desplegamos en un servidor. Si nos ponemos a pensar un poco a detalle en el tema , únicamente el paso dos es una tarea de desarrollo. Los otros

pasos están más orientados a infraestructura. No deberíamos tener que estar eligiendo continuamente las dependencias y el servidor de despliegue.

Spring Boot

SpringBoot nace con la intención de simplificar los pasos 1 y 3 y que nos podamos centrar en el desarrollo de nuestra aplicación. ¿Cómo funciona?. El enfoque es sencillo y lo entenderemos realizando un ejemplo. Para ello nos vamos a conectarnos **al asistente de Boot** que se denomina Spring Initializer.

Generate a

Maven Project

with Spring Boot

1.3.3

Project Metadata

Artifact coordinates

Group

com.arquitecturajava

Artifact

HolaSpringBoot

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Web, Security, JPA, Actuator, Devtools...

Selected Starters

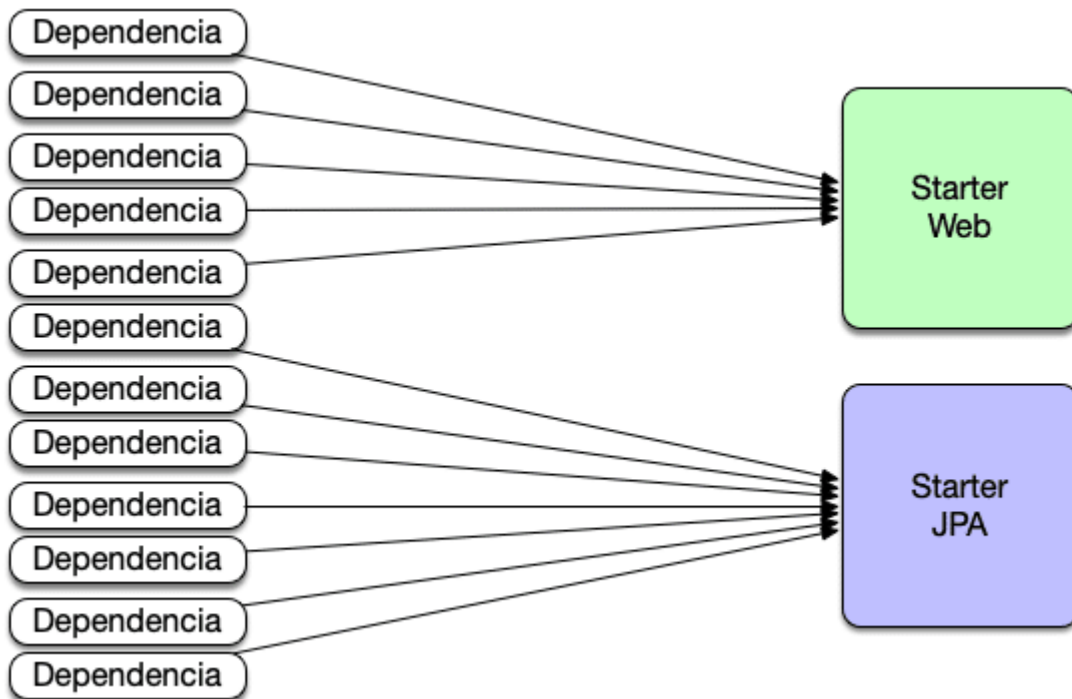
Web ✕

Generate Project ⌘ + ↵

String Boot Starter y simplificaciones

El asistente es intuitivo , elegimos el package al que queremos que nuestras clases pertenezcan , elegimos el nombre del proyecto y por último las dependencias. Eso sí ya no se trata de elegir jar por JAR sino por tipo de aplicación que necesitamos a este concepto [se](#)

















le denomina **Spring Starter**. Por lo tanto en vez de tener que elegir 10 o 20 dependencias es mucho más cómodo elegir 2 starters y Spring Boot se encarga del resto.



arquitectura**java**

Curso **GRATIS** **MAVEN** **Apúntate !!**

En este caso voy a construir una aplicación Spring MVC y elijo la dependencia web o Starter Web. Pulsamos generar proyecto y nos descargará un proyecto Maven en formato zip . Descomprimos el proyecto y este es su contenido.

 mvnw	✓	hoy 7:29
 mvnw.cmd	✓	hoy 7:29
 pom.xml	✓	hoy 7:29
▼  src	✓	hoy 8:31
▼  main	✓	hoy 8:31
▼  java	✓	hoy 8:31
▼  com	✓	hoy 8:31
▼  arquitecturajava	✓	hoy 8:31
 HolaSpringBootApplication.java	✓	hoy 7:29
▼  resources	✓	hoy 8:31
 application.properties	✓	hoy 7:29
▼  static	✓	hoy 7:29
▼  templates	✓	hoy 7:29
▼  test	✓	hoy 8:31
▼  java	✓	hoy 7:29
▶  com	✓	hoy 7:29

Una aplicación de Spring con estructura Maven totalmente configurada. El siguiente paso es usar Maven y escribir en línea de comandos.

```
mvn eclipse:eclipse
```

Esto convertirá el proyecto de Maven en un proyecto para Eclipse que podamos abrir con el editor. Si tenemos una versión moderna de Eclipse in siquiera tendremos que hacer ese paso nos valdrá con importar el proyecto de forma directa como Maven Project .Vamos a ver el contenido de la clase HolaSpringBootApplication

```
package com.arquitecturajava;
```

```
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication
public class HolaSpringBootApplication {

    public static void main(String[] args) {
        SpringApplication.run(HolaSpringBootApplication.class, args);
    }
}
```

Esta clase es la encargada de arrancar nuestra aplicación de Spring a diferencia de un enfoque clásico no hace falta desplegarla en un servidor web ya que Spring Boot provee de uno. Vamos a modificarla y añadir una anotación.

```
package com.arquitecturajava;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan(basePackages = "com.arquitecturajava")
public class HolaSpringBootApplication {

    public static void main(String[] args) {
        SpringApplication.run(HolaSpringBootApplication.class, args);
    }
}
```

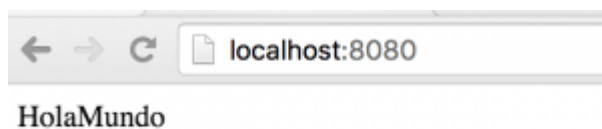
Recordemos que la anotación `@ComponentScan` escanea los packages buscando clases que SpringFramework pueda inyectar. En este caso vamos a construir un controlador de `HolaMundo`.

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class ControladorHola {

    @RequestMapping("/")
    @ResponseBody
    String home() {
        return "HolaMundo";
    }
}
```

Es momento de ejecutar nuestra aplicación como una aplicación de consola utilizando botón derecho run as Java Application Esto abrirá un servidor web y accederemos a la url.



SpringBoot nos ha simplificado toda la operativa a la hora de construir la aplicación prácticamente no hemos tenido que seleccionar dependencias de Spring y no ha hecho falta definir ningún servidor Tomcat en nuestro entorno de desarrollo ya que Spring Boot trae uno integrado.

Otros artículos relacionados

1. [Spring Boot JPA](#)
2. [Spring Boot Thymeleaf](#)
3. [PropertiesPlaceholder](#)

4. SpringConfiguration
5. Spring MVC