Spring Boot

DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Índice

1 Spring Boot Initializr

Spring Boot Initializr

Spring Initializr es una pequeña utilidad web que permite crear un esqueleto de un proyecto de Spring Boot con las opciones que queramos configurar.

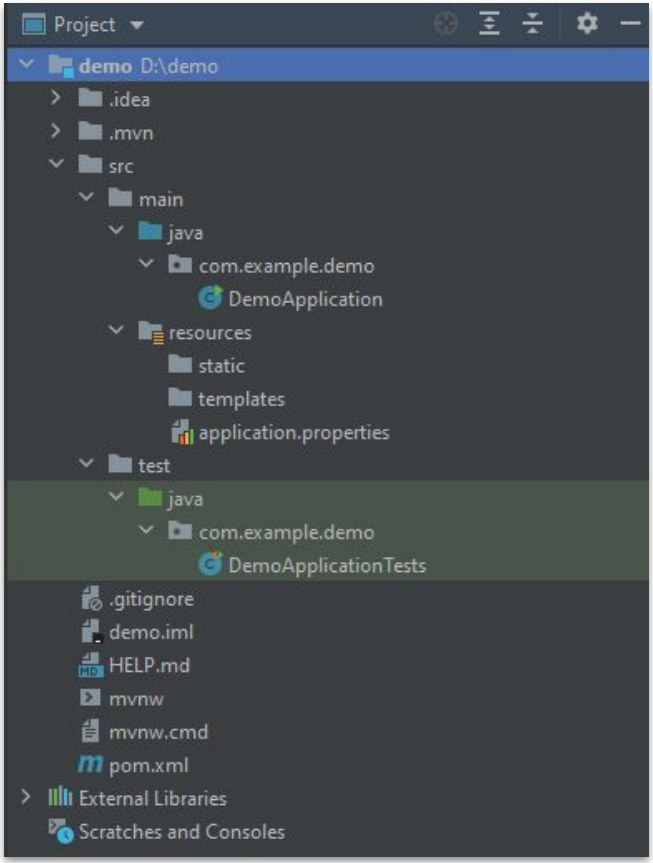
Permite elegir el proveedor de dependencias (Maven, Gradle, etc.), el lenguaje a utilizar (Java, Kotlin, etc.), el tipo de empaquetado (Jar o War), las dependencias/librerías que necesitamos, entre otras configuraciones iniciales.

Se puede utilizar desde su web oficial, o descargando un complemento para Intellij IDEA o el IDE que estemos utilizando.

Estructura del proyecto

Se crean los siguientes archivos y directorios. Entre los más importantes se encuentran:

- pom.xml.
- application.properties.
- carpeta target (luego de buildear el proyecto).
- profile (definir entorno de trabajo).



2 | @Annotations

¿Por qué nos interesan las @Annotations?

Con Spring Boot es posible poner en funcionamiento una aplicación de Spring con muy poca configuración, la cual está en forma de anotaciones.
Una anotación sirve para dar información al compilador o a las herramientas de desarrollo para hacer algo.

Una anotación siempre va delante de una clase, método, o declaración y se ve así:

```
@EstoEsUnaAnotation
```

¿Por qué nos interesan las @Annotations?

Puede tener elementos como parámetros con valor para otorgar más información, por ejemplo:

```
@Autor(nombre = “DH", fecha = “14/03/2019")
public class MiClaseEjemplo() {
  …
}
```

Podemos poner múltiples anotaciones:

```
@Autor(nombre = “DH", fecha = “14/03/2019")
@Deprecated
@OtraAnotacion("valor")
public class MiClaseEjemplo() {
  …
}
```

¿Cuáles son las más comunes?

@Deprecated

La usamos para indicar que un método es obsoleto. Imaginá que hay varios programadores trabajando en el mismo proyecto y tenemos un método que en un futuro será eliminado, entonces simplemente adicionás la anotación @Deprecated y listo.

@Override

Le informa al compilador que sobreescribirá el método de la clase padre. Esto es aplicable cuando existe herencia de clases. Si tu clase tiene un extends podés sobreescribir su código poniendo en tu método que está en la clase hijo con esta anotación.

@SuppressWarnings

Le dice al compilador que elimine las advertencias que se pueden mostrar, es decir, ya no te mostrará advertencias que pueda generar ese método. Será útil cuando ya no te interesen algunas advertencias que conoces.

3 | Definiendo la clase principal

SpringApplication

Como ya sabemos, toda aplicación en java debe contener una clase principal con un método main.

Dicho método, en caso de implementar una aplicación con Spring, deberá llamar al método run de la clase SpringApplication.

A continuación, definimos de una forma muy fácil la clase principal de nuestra aplicación y veamos como Spring Boot nos ayuda a simplificar las annotation que teníamos con Spring Framework.

@Configuration indica que la clase en la que se encuentra contiene la configuración principal del proyecto.

La anotación **@EnableAutoConfiguration** indica que se aplicará la configuración automática del starter que hemos utilizado. Solo debe añadirse en un sitio, y es muy frecuente situarla en la clase main.

En tercer lugar, la etiqueta **@ComponentScan** ayuda a localizar elementos etiquetados con otras anotaciones cuando sean necesarios.

```
@Configuration
@EnableAutoConfiguration
@ComponentScan
public class Application {

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

Para no llenar nuestra clase de anotaciones, a diferencia de usar solo Spring Framework, Spring Boot nos permite sustituir las etiquetas @Configuration, @EnableAutoConfiguration y @ComponentScan por @SpringBootApplication, que engloba al resto.

```
@SpringBootApplication = {

    @Configuration
    @EnableAutoConfiguration
    @ComponentScan
}
```

DigitalHouse>