Desarrollo web en entorno servidor

UT1.2 – Programación web

4 – Aplicaciones de consola en Spring Boot

IES Clara del Rey – Madrid

Aplicación mínima Spring Boot

Si se crea una aplicación Spring Boot con Spring Initializr o con el asistente de IntelliJ o STS, con Maven, sin añadir ninguna clase de dependencias adicionales, se crea un proyecto:

- Con las dependencias de org.springframework.boot:
  - spring-boot-starter – La base de Spring Boot
  - spring-boot-starter-test – Soporte para test
- Con un plugin de Maven para dar soporte en el entorno para algunas tareas: arrancar la aplicación, empaquetarla en JAR o WAR, etc.
- Con una clase que lanza la aplicación Spring Boot.

Clase de la aplicación mínima Spring Boot

La clase de una aplicación mínima Spring Boot es una clase anotada con @SpringBootApplication, con el siguiente código:

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

El nombre de la clase cambiará en función del nombre del proyecto creado. Se puede cambiar sin problemas por el nombre que deseemos. Es muy habitual llamar a esta clase “App” o similar.

La anotación @SpringBootApplication hace que se lance la aplicación.

La anotación @SpringBootApplication

Es una anotación compuesta que simplifica la configuración.

Combinar tres anotaciones de Spring:

- @Configuration: Permite definir beans mediante métodos anotados con @Bean dentro de la clase.
- @EnableAutoConfiguration: Habilita la configuración automática. Spring intentará configurar automáticamente los beans necesarios en función de las dependencias agregadas en el POM.XLM.
- @ComponentScan: Activa el escaneo de componentes, buscando otras clases anotadas con @Component, @Service, @Repository, o @Controller dentro del mismo paquete o subpaquetes.

Volveremos a esto cuando veamos qué son y como funcionan los beans.

Código en inicio

Para ejecutar código al inicio de una aplicación Spring, para realizar alguna tarea inicial, hay dos opciones:

- Modificar el método main() de la aplicación
- Usar clases que implemente CommandLineRunner

Aunque las dos son válidas, la diferencia entre una y otra es el momento en que se ejecutan:

- Cuando se ejecuta el método main puede que no se haya completado la inicialización del contexto de Spring, los distintos componentes de la aplicación.
- Cuando se ejecutan los commandLineRunners, ya se ha inicializado el contexto de Spring, todos los componentes.

Código en inicio – CommandLineRunner

Un command line runner es una clase que:

- Implementa la interfaz CommandLineRunner. Esta interfaz tiene un método abstracto “run” que sería, por así decirlo, el equivalente de un main() de un programa Java “clásico”
- Está anotada con @Component. Esta anotación, y otras que veremos más adelante, indican a Spring que debe crear un objeto de esa clase.

Cuando se lanza la aplicación, Spring localiza todas las clases que implementan CommandLineRunner, que tengan una anotación @Component o derivada, y lanza sus métodos run.

Esto se hace después de la inicialización del contexto de Spring.

CommandLineRunner – Orden de ejecución

Cuando hay varios Command Line Runners en una aplicación, Spring elige el orden en que se ejecutan los métodos de estas clases. No se garantiza un orden específico.

Para establecer el orden en que se ejecuten los CommandLineRunners hay dos opciones:

- Usar la anotación @Order. Se indica como parámetro el orden de ejecución que deseamos.
- Implementar en el CommandLineRunner la interfaz Ordered. Esta interfaz tiene un método abstracto getOrder que tiene que devolver el orden en que se desea ejecutar el CommandLineRunner.