# Introducción a Spring Framework

Desarrollo Web en Entorno Servidor

Ana Viciano Fabregat
a.vicianofabregat@edu.gva.es

IES Álvaro Falomir
Curso 2023-2024

# ¿Qué es Spring?

Spring es el framework de desarrollo de aplicaciones de código abierto más popular para Java.

El objetivo de Spring es simplificar el desarrollo de aplicaciones ofreciendo el soporte de infraestructura a nivel de aplicación.

Con ello, por un lado las aplicaciones resultan más escalables y robustas y, por otro, los desarrolladores podemos centrarnos en la lógica empresarial que requiere la aplicación, haciendo el proceso más corto, rápido y eficaz.

# ¿Qué es Spring?

Spring es un ecosistema de proyectos. Los principales son:

- Spring Boot
- Spring Data
- Spring Cloud
- Spring Security
- Spring Batch
- Spring Web Services
- ...

# Spring Boot

Spring Boot es un proyecto de Spring que nos proporciona un conjunto de bibliotecas que simplifican enormemente el desarrollo de aplicaciones basadas en Spring.

Es decir, nos ofrece la opción de configurar los elementos que necesitamos de forma rápida y sencilla mediante starters, sin necesidad de realizar configuraciones ni administrar dependencias.

Podemos generar un proyecto de Spring Boot mediante Spring Initializr desde:

- Un IDE, como IntelliJ.
- Su propia web https://start.spring.io/

# Spring Boot

Algunos de esos starters son:

- spring-boot-starter-parent: define los valores predeterminados comunes, como la versión de Java, la codificación de fuente, o varias configuraciones de complementos de Maven
- spring-boot-starter-web: específica un grupo de dependencias para desarrollar aplicaciones web como Spring web MVC y el servidor Tomcat.
- spring-boot-maven-plugin: permite la creación automática de un archivo JAR ejecutable para una aplicación web Java independiente con un servidor Tomcat incorporado.
- spring-boot-starter-data-jpa: incluye todas las dependencias necesarias para usar Spring Data JPA, junto con las dependencias de la biblioteca Hibernate, la implementación de JPA más utilizada.

# El main de Spring Boot y su anotación

```java
@SpringBootApplication
public class DemoApplication {

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

}
```

Al iniciar nuestra aplicación, lo primero que observamos es que se crea una clase XXXXApplication con un método main.

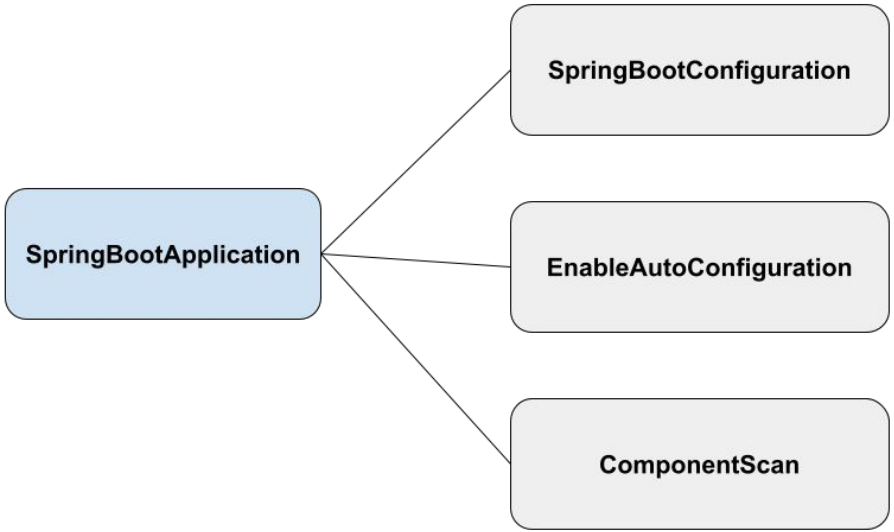
La clase está anotada con @SpringBootApplication y el método main tiene la instrucción SpringApplication.run(...) para indicarle a Spring que arranque.

# ¿Qué hace @SpringBootApplication?

@SpringBootApplication es la suma de @SpringBootConfiguration, @EnableAutoConfiguration y @ComponentScan:

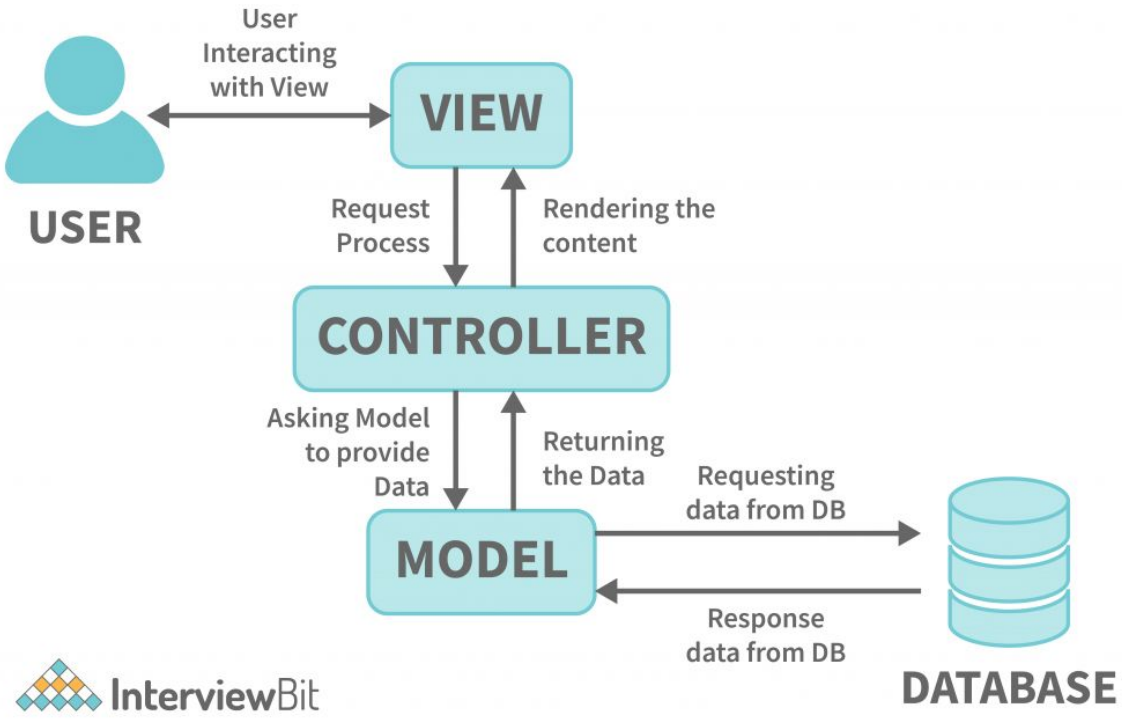
- @SpringBootConfiguration: indica que una clase proporciona la configuración de la aplicación.
- @EnableAutoConfiguration: se encarga de configurar automáticamente la aplicación y buscar las clases con @Entity y registrarlas con el proveedor de persistencia.
- @ComponentScan: le da la directiva a Spring para que haga una búsqueda de todos los componentes anotados como @Component (y sus clases derivadas) dentro de los paquetes, registrándolos en su contexto.

# ¿Qué hace @SpringBootApplication?



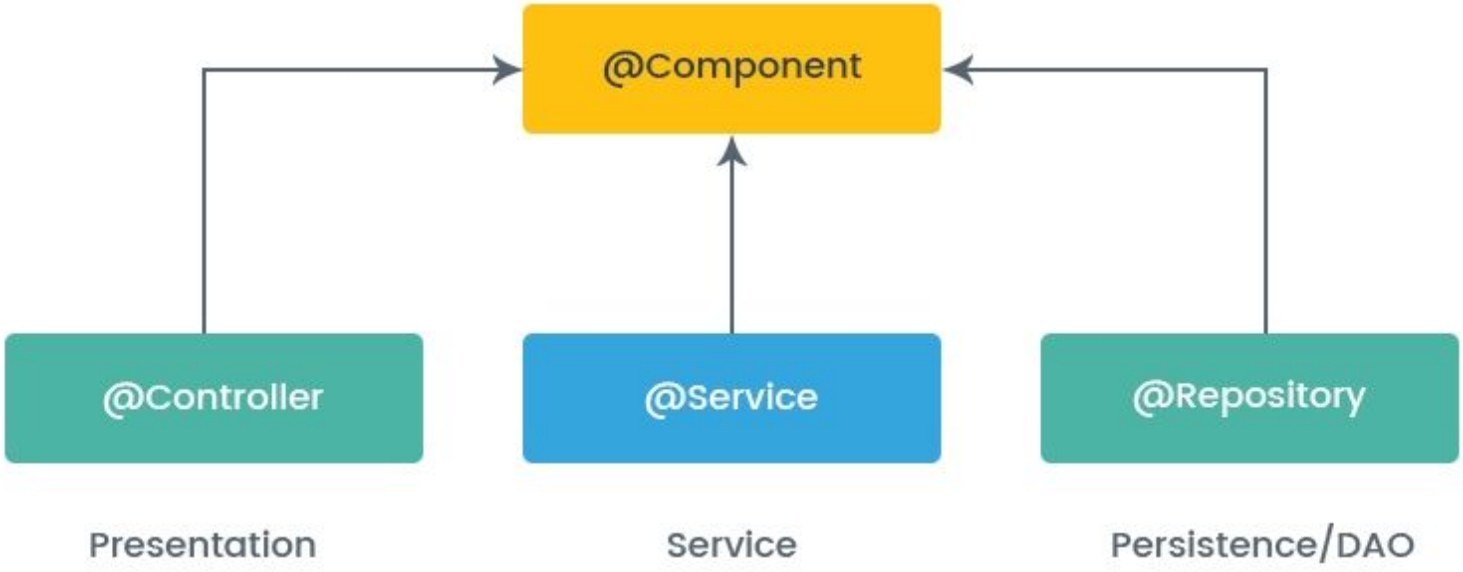

En resumen, @SpringBootApplication se utiliza para que Spring configure la aplicación y busque los componentes que necesitamos

# Arquitectura MVC



# Componentes Spring

Spring categoriza componentes mediante un conjunto de anotaciones. Cada componente tiene una responsabilidad concreta:



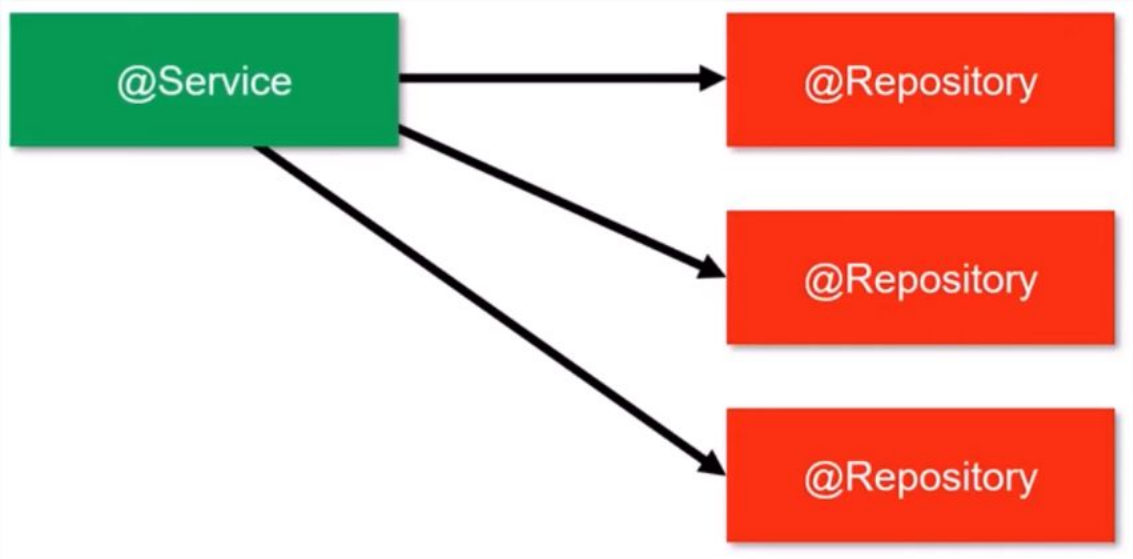
# @Repository

Los componentes @Repository implementa la interfaz y operaciones de persistencia de la información.

Por lo tanto, se encarga de almacenar datos en una base de datos, API externa o repositorio de información que se necesite.

# @Service

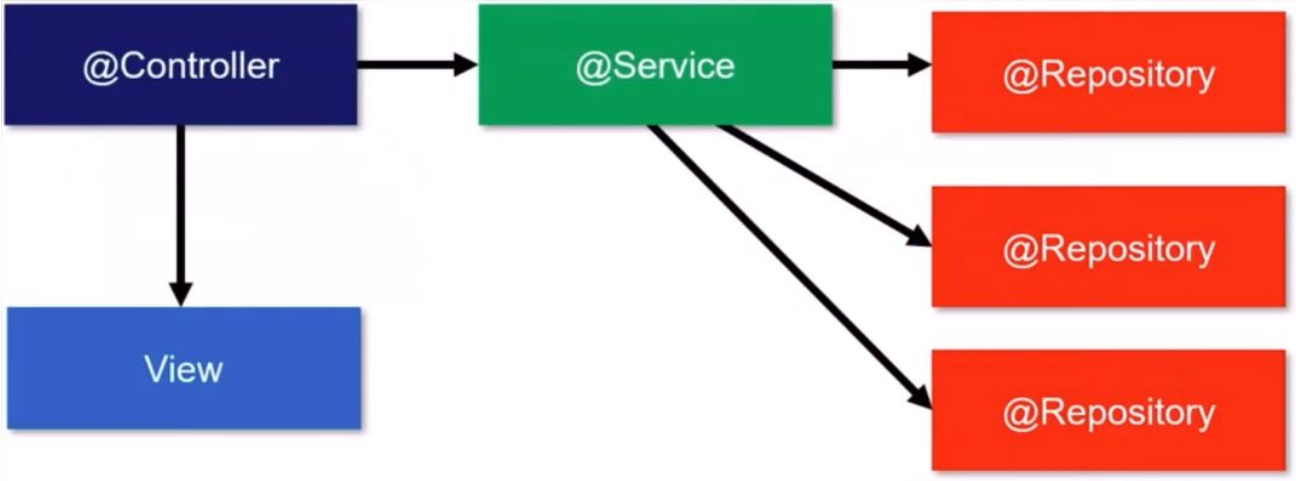
Los componentes @Service implementan la parte de negocio y aglutinan varios repositorios y llamadas a otros servicios.



# @Controller

Los componentes @Controller (o @RestController con API REST) se encargan de recibir las peticiones de los usuarios y devolver respuestas.

Para ello, se apoya habitualmente en algún motor de plantillas.



# Spring Data

Spring Data es un proyecto de persistencia que nos permite acceder a bases de datos de forma sencilla y realizar, entre otras, operaciones CRUD, de paginación, ordenación o búsquedas.

Actualmente, ofrece funcionalidad tanto para bases de datos relacionales como no relacionales, como MongoDB, Redis y más.

Spring Data JPA es una subparte de Spring Data que se centra en facilitar el uso de JPA (Java Persistance API).

# Inversión de Control

Inversión de Control (IoC) es un principio de diseño de software que invierte el control del flujo de la aplicación.

En un programa tradicional, el flujo de control está dictado por el propio programa, lo que significa que el programa controla la creación y gestión de los objetos.

Sin embargo, en un programa que utiliza IoC, este control se invierte, es decir, el framework se encarga de la creación y gestión de los objetos, reduciendo el acoplamiento entre las clases y permitiendo mayor flexibilidad y modularidad.

# Inyección de dependencias

La Inyección de Dependencias (DI) es una técnica que implementa el principio de IoC para la gestión de dependencias entre objetos.

En lugar de que los objetos creen o busquen sus dependencias, estas se "inyectan" en ellos por el framework.

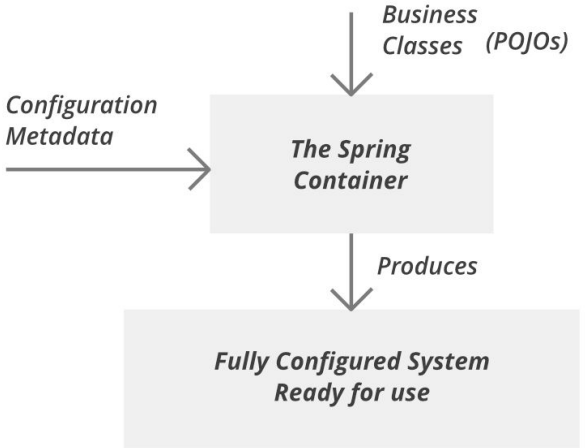
En Spring, esto se puede hacer a través de constructores, métodos setter o campos directamente.

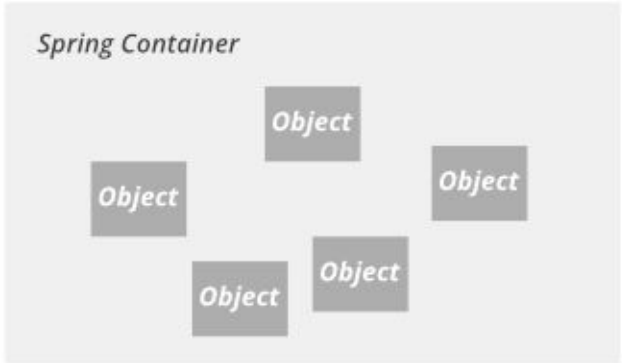
La DI facilita la prueba unitaria, ya que las dependencias pueden ser fácilmente sustituidas por mocks.

# Contenedor Spring

En Spring, los conceptos IoC y DI se implementan a través del contenedor Spring, que crea y gestiona los objetos de la aplicación, que se conocen como beans.

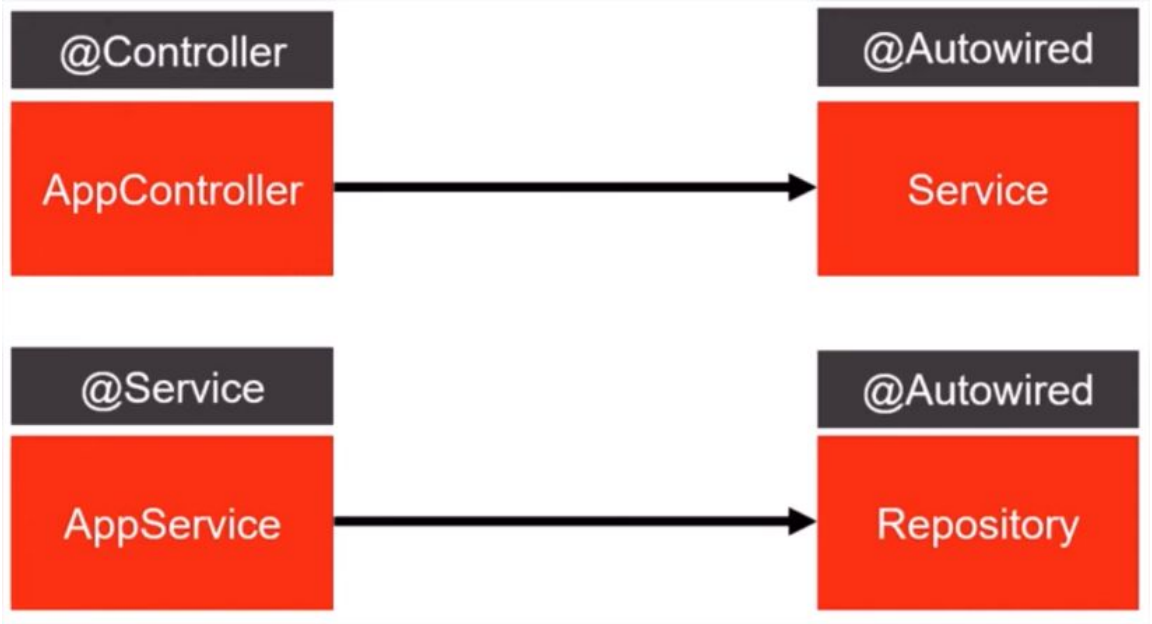
Los beans y sus dependencias se configuran en archivos de configuración XML o mediante anotaciones en el código.



# Anotación @Autowired

@Autowired es una anotación que permite inyectar unas dependencias con otras dentro de Spring.



# Ejemplo con @Autowired

Si una clase A depende de una clase B, en lugar de crear un objeto B dentro de A con new B(), podemos declarar la dependencia con la anotación @Autowired y Spring se encarga de inyectarla.

```
public class A {
    private B b;

   @Autowired
    public A(B b) {
        this.b = b;
    }
}
```

En este ejemplo, Spring crea un bean de B y lo inyecta en A cuando cree un bean de A, sin que te preocupes por crear y gestionar B.