

SPRING FRAMEWORK 4

---

# ESTRUCTURA GENERAL DE UNA APLICACIÓN WEB

### UN POCO DE HISTORIA...

- ▶ En las primeras aplicaciones cliente-servidor, cada programa cliente debía instalarse en cada ordenador personal atendiendo a su sistema operativo. Esto suponía un alto coste en desarrollo y una baja productividad.
- ▶ Las aplicaciones web actuales se basan en la generación de páginas dinámicas en un formato estándar (como HTML), por lo que son válidas en cualquier dispositivo con un navegador instalado en el sistema.
- ▶ Sólo es necesario crear un cliente que funcione en cualquier navegador, independientemente de su sistema operativo. La aplicación web se escribe una vez y se ejecuta igual en todas partes.

### VENTAJAS DE UNA APLICACIÓN WEB

- ▶ Ahorra tiempo en el desarrollo.
- ▶ No hay problemas de compatibilidad.
- ▶ No ocupa espacio en el disco duro.
- ▶ Actualizaciones inmediatas.
- ▶ Consumo bajo de recursos del sistema.
- ▶ Multiplataforma.
- ▶ Disponibilidad alta.

### EL MODELO DE TRES CAPAS

- ▶ Una aplicación web se estructura en tres capas principales:
  - ▶ Navegador → Interacción de usuario.
  - ▶ Servidor web → Lógica de la aplicación.
  - ▶ Base de datos → Almacén los datos.



## ESTRUCTURA GENERAL DE UNA APLICACIÓN WEB

---

### NAVEGADOR



- ▶ Es la capa visual con la que el usuario o cliente interactúa.
- ▶ El navegador simplemente se encarga de representar texto en pantalla mediante un lenguaje de marcas enriquecido, por ejemplo, con Javascript. Interpreta código y lo muestra como contenido.
- ▶ Puede hacer uso de hiperenlaces o hipervínculos que enlazan una porción de texto o una imagen a otro documento.
- ▶ El seguimiento de enlaces de una página a otra se denomina navegación.
- ▶ Tanto en este curso, como en los diferentes proyectos informáticos donde te ubiques en tu carrera profesional, al conjunto de páginas las denominaremos vistas.

### SERVIDOR WEB



- ▶ El servidor web es un software que permite desarrollar lógica compleja en la aplicación. Esta lógica es llamada negocio o lógica de negocio.
- ▶ Realiza conexiones generalmente bidireccionales, generando respuestas para la aplicación cliente.
- ▶ Para esta comunicación se usa el protocolo HTTP.
- ▶ En este curso aprenderemos a utilizar Spring Framework para manejar toda la lógica de la aplicación.

# SERVIDOR WEB - PROTOCOLO HTTP



- ▶ El protocolo HTTP es un conjunto de reglas que permiten una comunicación efectiva entre el cliente y el servidor.
- ▶ A éste protocolo se le suele asignar los puertos 80 u 8080.
- ▶ Todo comienza escribiendo una dirección web o URL en nuestro navegador preferido:  
<http://www.udemy.com/index.html>
  - ▶ Esto produce un socket a un servidor DNS que se encargará de transformar la dirección alfanumérica a numérica (una IP).
  - ▶ Una vez obtenida la dirección IP, abre un socket con ésta mediante TCP.
  - ▶ Prepara la petición para el servidor 'GET /index.html HTTP/1.1'.
  - ▶ Comprueba si se encuentra en la caché.
  - ▶ Se presenta el recurso en pantalla (si es preciso).

# SERVIDOR WEB - PETICIONES HTTP

- ▶ Las peticiones HTTP son las encargadas de usar este protocolo para comunicar a los clientes con el servidor.
- ▶ Cada petición HTTP va acompañada de:
  - ▶ Una dirección o URL, que indica el recurso al que se quiere acceder.
  - ▶ Unos datos llamados parámetros, necesarios en ciertas ocasiones.
  - ▶ Unas cabeceras, también llamadas headers, que amplían información a la petición (por ejemplo: el navegador utilizado o User Agent, el lenguaje, etc...).
- ▶ Existen diferentes tipos de petición, las más comunes y recomendadas son:
  - ▶ GET: Utilizada para solicitar recursos.
    - ▶ <http://www.servidor.com/buscarPersona?nombre=jorge&apellido=lopez>
  - ▶ POST: Utilizada para crear, modificar y eliminar recursos.
    - ▶ <http://www.servidor.com/altaPersona> → los datos van ocultos.

# SERVIDOR WEB - RESPUESTAS HTTP

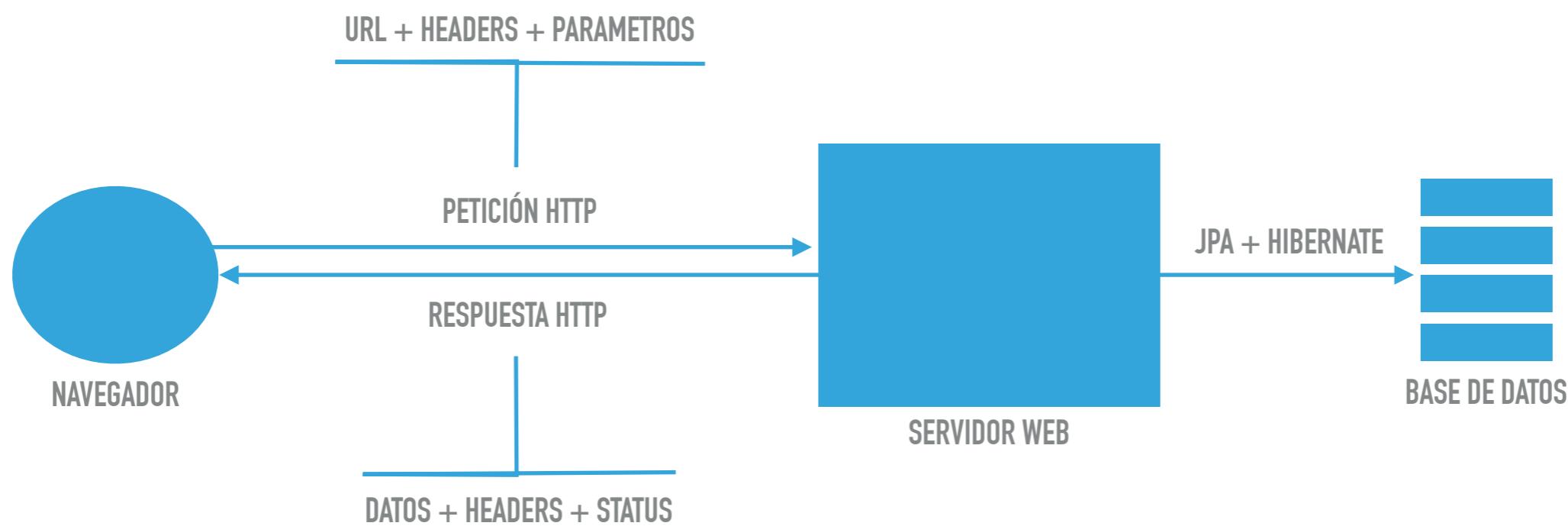
- ▶ El servidor nos devolverá una respuesta HTTP con el siguiente contenido:
  - ▶ La información solicitada, que puede ser el código HTML o datos en otros formatos.
  - ▶ Los headers, para ampliar información.
  - ▶ El estado de la respuesta, que va asociada a códigos de tres dígitos del 1XX al 5XX:
    - ▶ 1XX: Respuestas informativas.
    - ▶ 2XX: Peticiones incorrectas.
    - ▶ 3XX: Redirecciones.
    - ▶ 4XX: Errores del cliente.
    - ▶ 5XX: Errores del servidor.

### BASE DE DATOS



- ▶ Las bases de datos son sistemas que contienen información clasificada de distinta manera, relacionados entre sí mediante un tipo de unión.
- ▶ Es importante para el curso que tengas unos conocimientos básicos sobre el modelo entidad relación, dado a que trabajaremos con bases de datos relaciones como MySQL.
- ▶ Para que nuestro backend Spring pueda acceder a los datos utilizaremos Spring Data JPA e Hibernate, apoyándonos en QueryDSL para facilitar consultas complejas.

# CHECK-IN DE CONOCIMIENTOS

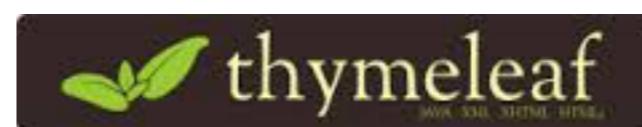


¿EN QUÉ CAPA NOS CENTRAREMOS EN ESTE CURSO?

---

## TECNOLOGÍAS QUE VAMOS A UTILIZAR

- ▶ Spring Boot.
- ▶ Spring Core.
- ▶ Spring MVC.
- ▶ Thymeleaf.
- ▶ Spring Rest.
- ▶ Spring Data JPA.
- ▶ Hibernate.
- ▶ QueryDSL.
- ▶ Log4j.
- ▶ Spring Security.
- ▶ Spring Loaded.
- ▶ Spring Batch.



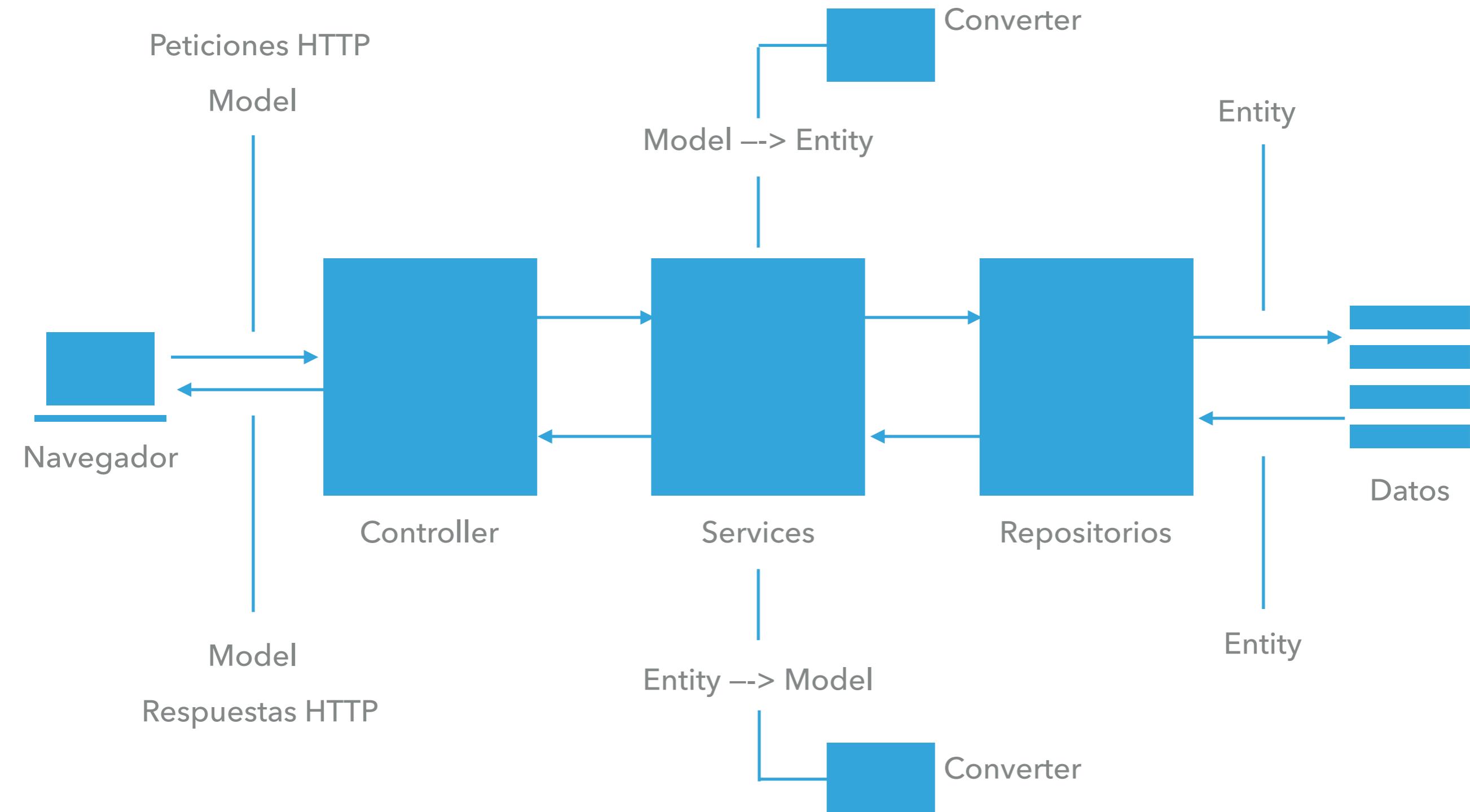
# DEFINICIÓN DE UN FRAMEWORK

- ▶ Un framework es un marco de trabajo que facilita el desarrollo de software.
- ▶ Proporciona un esqueleto, patrón que el programador debe seguir, preocupándose únicamente de la codificación.
- ▶ Spring es un framework basado en Java orientado a aplicaciones de gran magnitud.

### VENTAJAS

- ▶ Agiliza la codificación de aplicaciones.
- ▶ Es modular y estándar.
- ▶ Permite modificar o ampliar el software con mayor facilidad.
- ▶ Soporte constante por los desarrolladores y la comunidad.

# ARQUITECTURA DE UNA APLICACIÓN WEB SPRING



# CONFIGURACIÓN SPRING ANTIGUA

```
<beans xmlns="http://www.springframework.org/schema/beans">
    <bean id="currentDate" class="com.spring.app.revision.Watch"
        scope="prototype"/>
</beans>
```



+

Java Configuration

```
ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);
```

```
@Configuration
```

```
class AppConfig {
```



```
    @Scope("prototype")
```

```
    @Bean(name="currentDate")
```

```
    public Watch getWatch() {
```

```
        return new Watch();
```

```
}
```

```
}
```

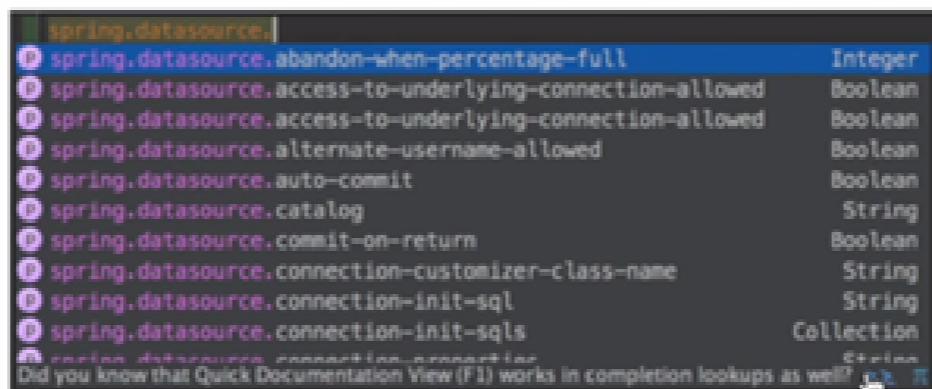
# CONFIGURACIÓN SPRING ACTUAL

```
package hello;

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

@SpringBootApplication
public class Application {

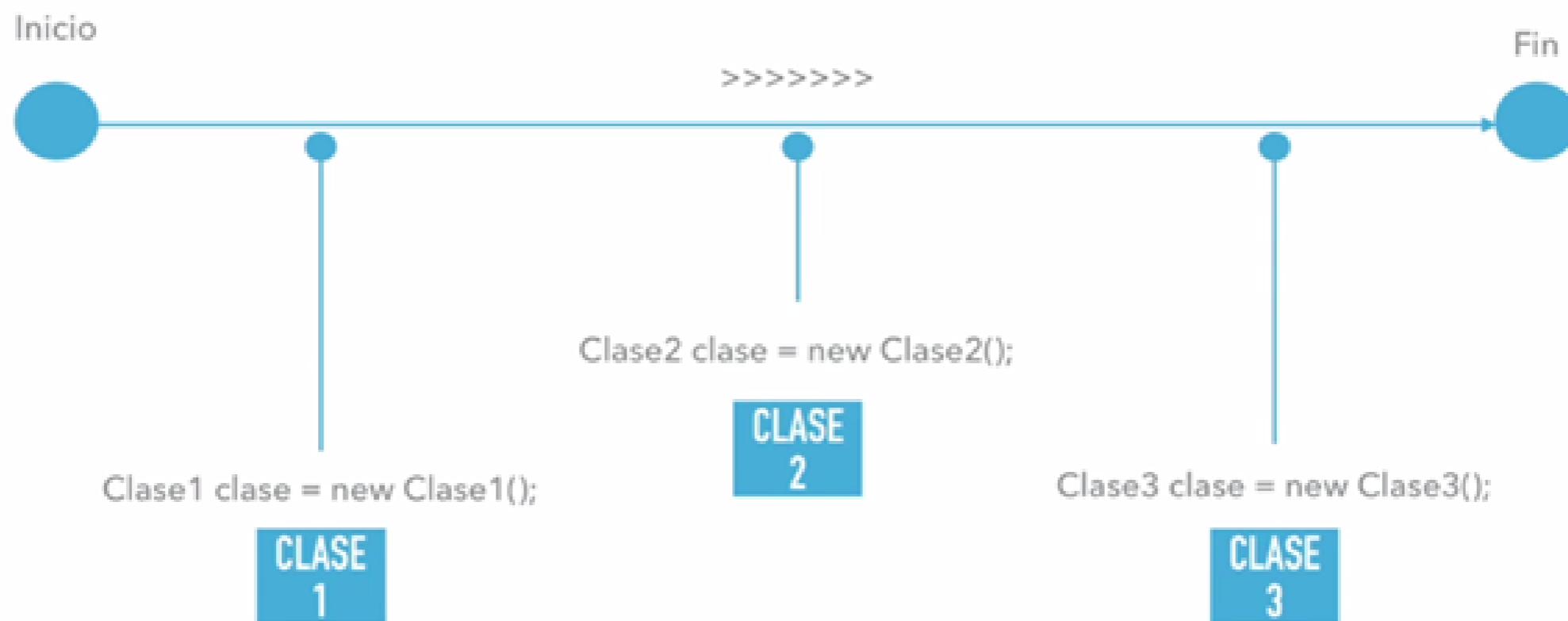
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```



## INTRODUCCIÓN A LA INYECCIÓN DE DEPENDENCIAS

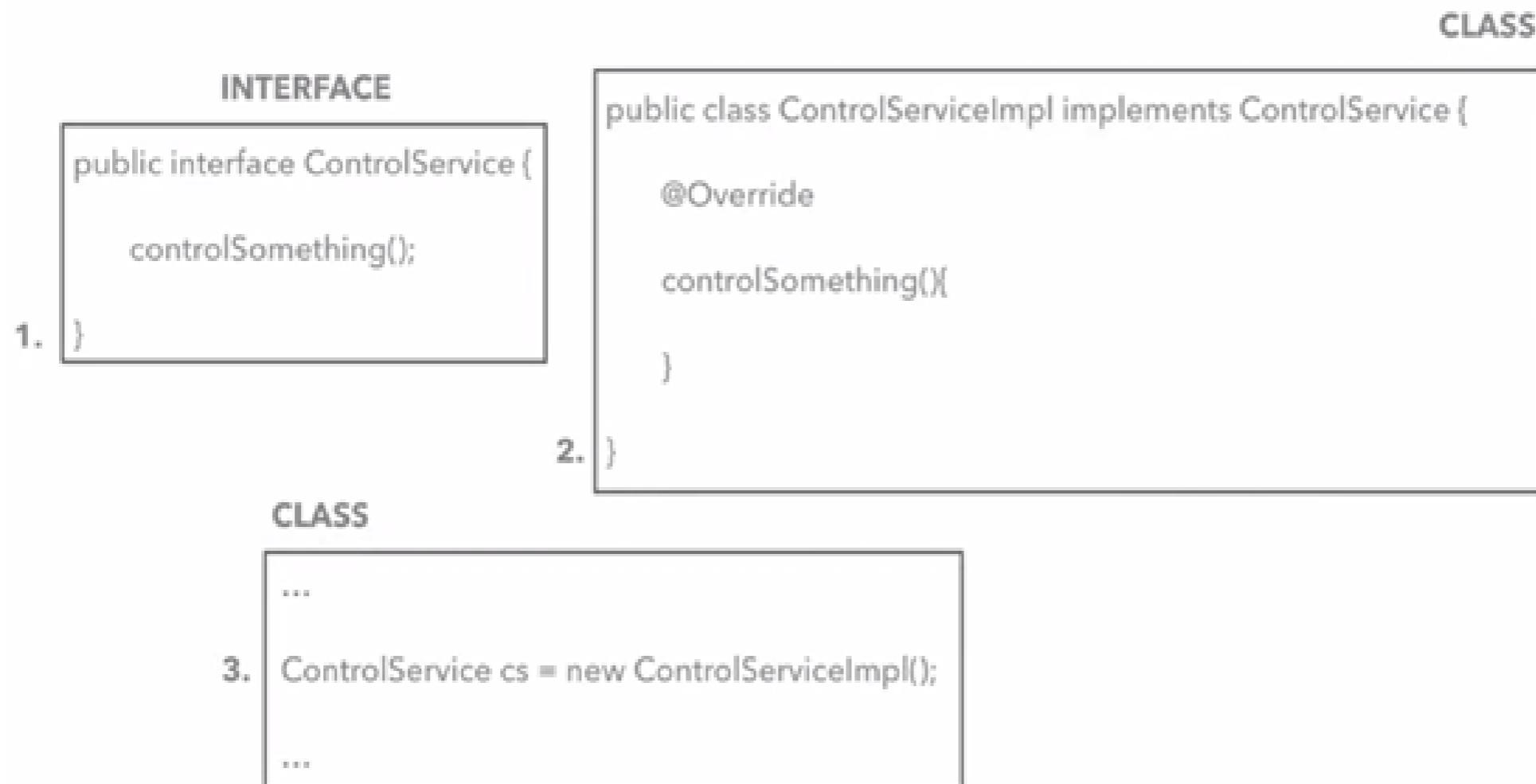
---

# FLUJO NORMAL



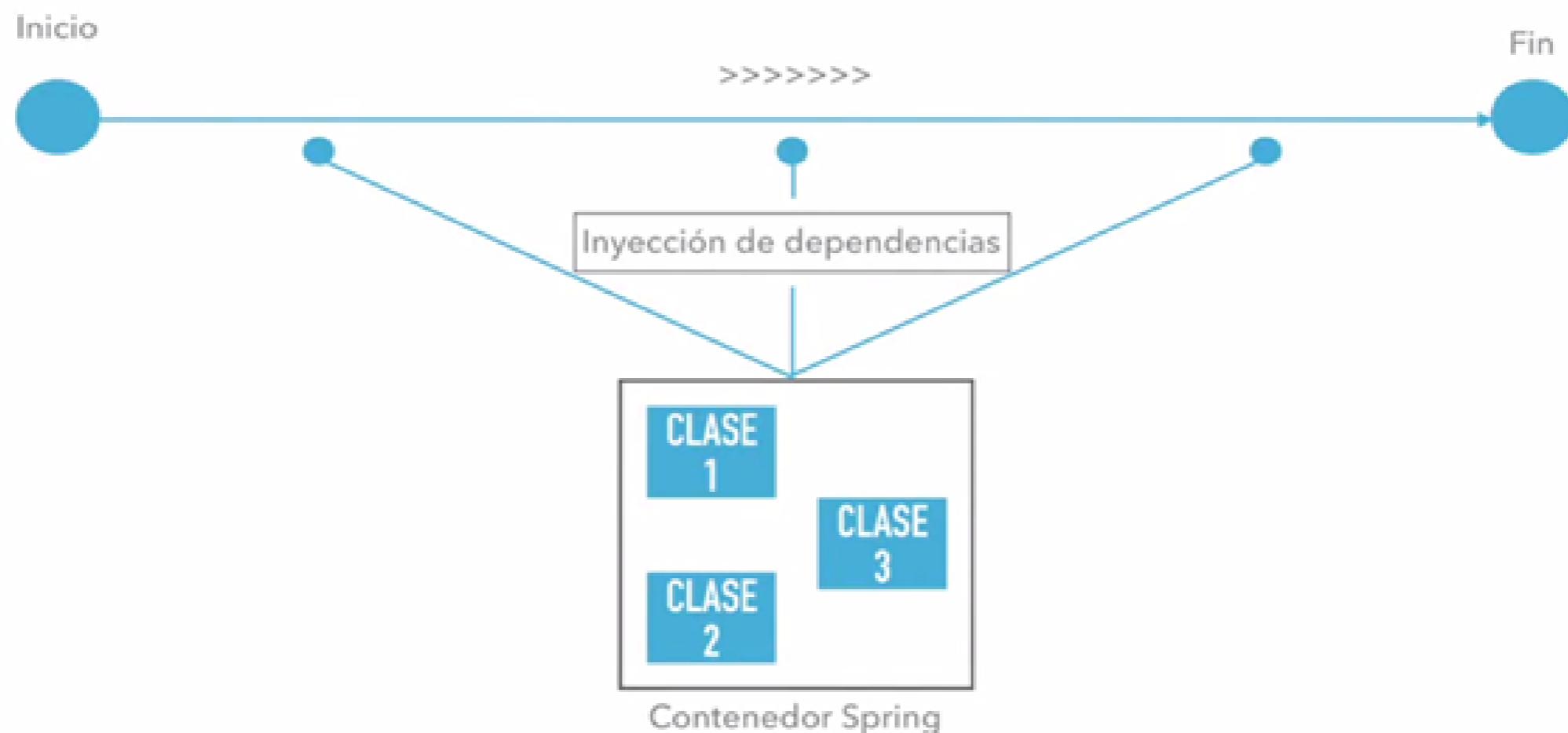
## INTRODUCCIÓN A LA INYECCIÓN DE DEPENDENCIAS

### CREACIÓN DE OBJETOS MEDIANTE NEW



## INTRODUCCIÓN A LA INYECCIÓN DE DEPENDENCIAS

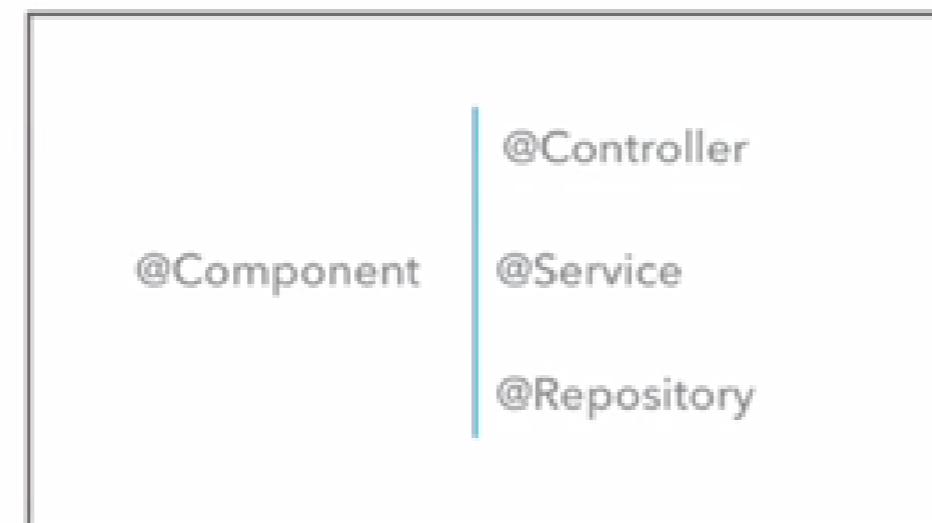
# FLUJO CON EL CONTENEDOR SPRING



## INTRODUCCIÓN A LA INYECCIÓN DE DEPENDENCIAS

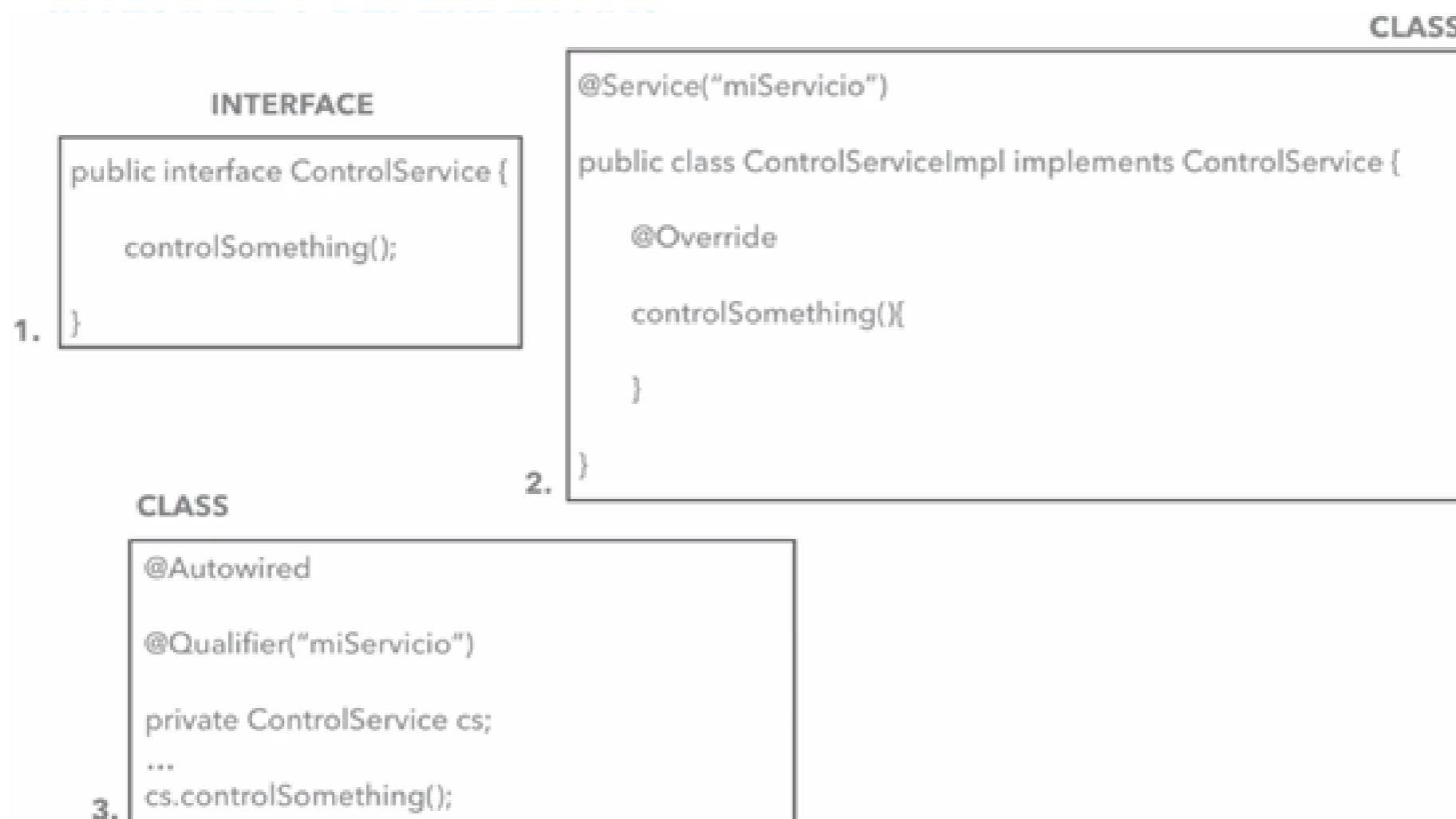
---

# TIPOS DE ANOTACIONES



## INTRODUCCIÓN A LA INYECCIÓN DE DEPENDENCIAS

# INJECTANDO DEPENDENCIAS



## INTRODUCCIÓN A LA INYECCIÓN DE DEPENDENCIAS

# INYECTANDO DEPENDENCIAS

