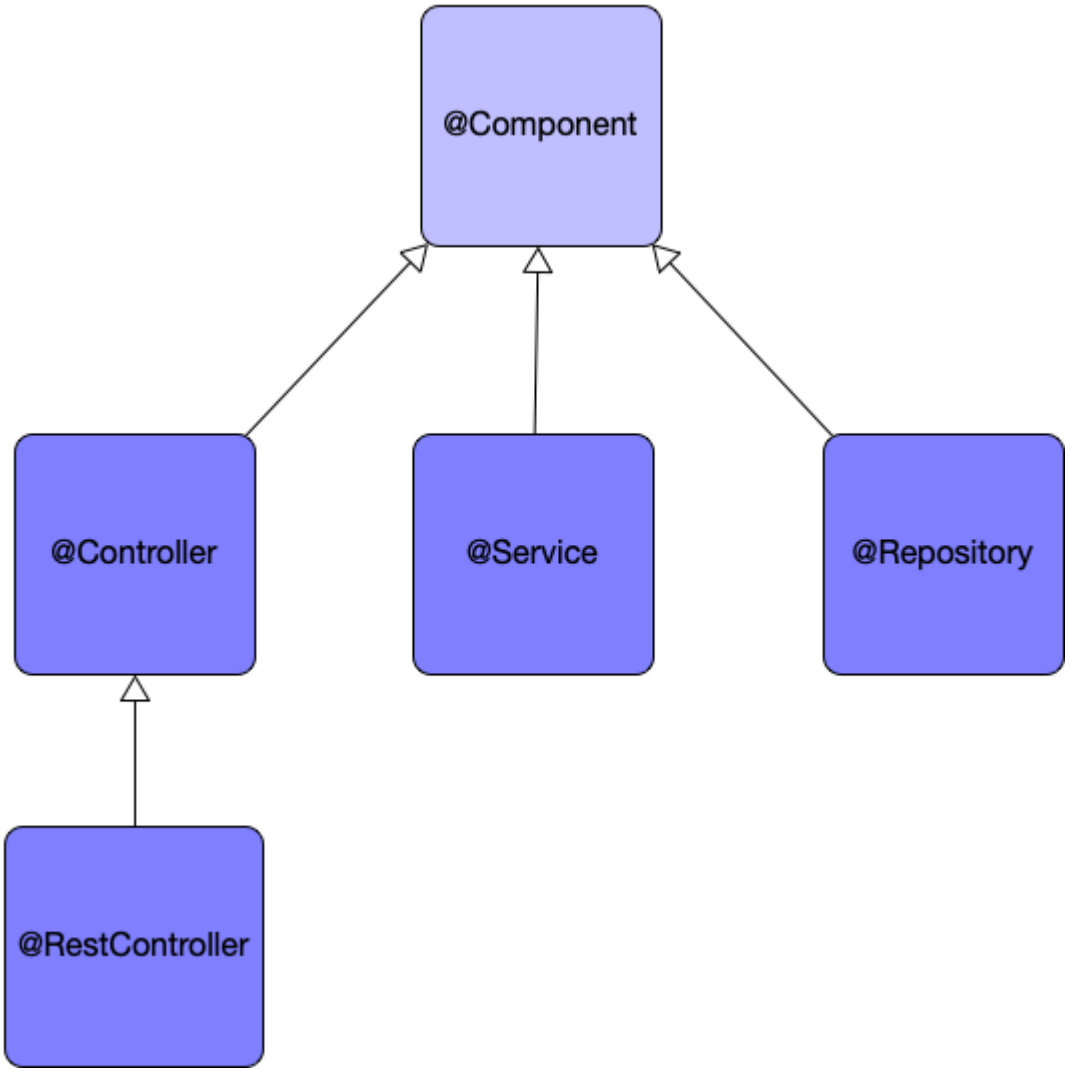
Spring @Bean vs @Component es una de las preguntas más habituales cuando usamos Spring Framework ya sea el de forma directa o con Spring Boot. Que diferencia hay entre estas dos opciones . La realidad es que tienen fuertes similitudes pero vamos a hablar un poco mas a detalle de cada una de de ellas.

Spring y Componentes (Objetos)

Spring es un framework que nos permite simplificar y reducir el código que nosotros desarrollados a través del registro de componentes y el uso de helpers que reducen drasticamente el código . La forma mas sencilla de registrar un componente con Spring es añadir a la clase @Component de esta forma Spring instancia un objeto de este clase como Singleton por defecto y lo registra como un objeto bajo su control . Hay que recordar que @Component no es la forma más habitual de registrar los objetos sino que es mucho más típico usar @Controller @Service y @Repository para realizar estas tareas que son anotaciones que heredan de @Component.



De igual manera @RestController hereda de @Controller . Con estas operativas normalmente podemos configurar Spring Framework y sus componentes sin problemas. Pero eso aborda una configuración por defecto de ellos . En donde se inyectan sus dependencias etc.

Spring @Bean vs @Component

Hay situaciones en las cuales nosotros tenemos que configurar a detalle el Objeto que se va a crear . En este caso las anotaciones de @Component y sus hijas no aportan la misma flexibilidad . Para abordar esta punto en el que necesitamos un control exhahustivo de la inicialización del objeto usamos @Bean lo cual nos permitirá entrar mucho más a detalle.

```
package com.arquitecturajava.bean;

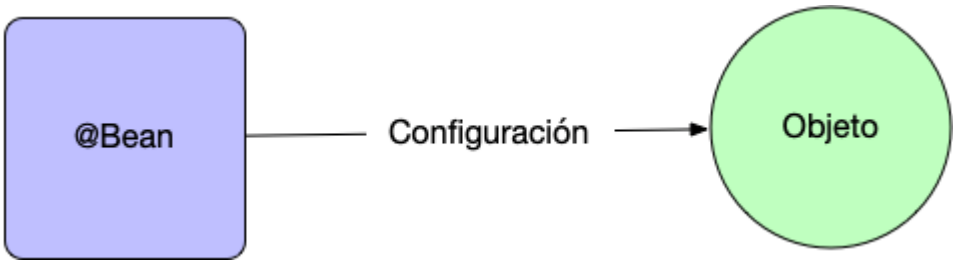
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class BeanApplication {

  public static void main(String[] args) {
    SpringApplication.run(BeanApplication.class, args);
  }
  @Bean
  public GestorCarpeta crearHilo() {
    return new GestorCarpeta("micarpeta", "mifiltro");
  }

}
```

Por ejemplo en este caso inicializamos con Spring Boot un Bean que se denomina GestorCarpeta y que necesita dos parámetros de inicialización uno el nombre de carpeta y otro el filtro a aplicar a la carpeta .



Usamos @Bean e instanciamos totalmente a medida nuestro objeto no solo eso sino que nos podemos ayoyar en otras clases o métodos que tendra Spring para finalizar la inicialización correcta. Aquí podemos ver un ejemplo sencillo de dar de alta un dataSource.

```
@Bean
public DataSource getDataSource() {
    DataSourceBuilder dataSourceBuilder = DataSourceBuilder.create();
    dataSourceBuilder.username("cecilio");
    dataSourceBuilder.password("cecilio");
    return dataSourceBuilder.build();
}
```

Tengamos siempre en cuenta esta anotación para particularizar la inicialización de los diferentes componentes de Spring. Esa es la diferencia entre Spring @Bean vs @Component

- Spring @Component , anotaciones y jerarquía
- JavaScript reduce y su flexibilidad
- Spring @Bean y su uso
- Spring Stereotypes y anotaciones
- Spring Batch Hello World y su configuración
- Spring @Component