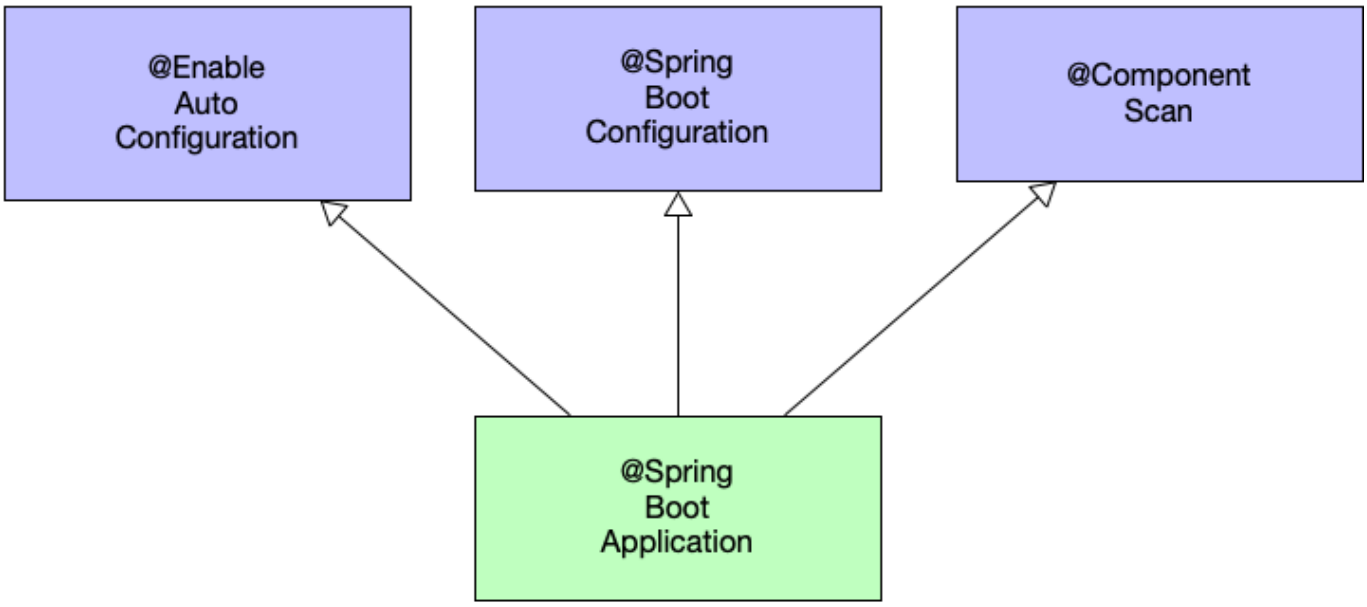
@SpringBootApplication es la anotación que aparece en la función main de todo proyecto que definamos con Spring Boot .Ahora bien la mayor parte de los desarrolladores no saben exactamente para que sirve esta anotación y como funciona. Vamos a explicarlo

¿Para que sirve?



Cuando nosotros usamos la anotación @SpringBootApplication esta anotación herede el comportamiento de un conjunto de anotaciones amplio que tenemos que explicar:



Herencia entre anotaciones

@EnableAutoConfiguration : Esta es una anotación clásica de Spring que se encarga de forma inteligente de intentar configurar Spring de forma automática . Es la anotación encargada de buscar en el Classpath todas las clases con @Entity y registrarlas con el proveedor de persistencia que tengamos. Por lo tanto por eso con Spring Boot es suficiente

configurar simplemente el dataSource a nivel application.roperties ya que Spring buscará todos las clases.

@SpringConfiguration : Es la anotación que define que el fichero es un fichero de Configuración de Spring .Normalmente esto se solía hacer antiguamente con @Configurátión . La particularidad que tiene @SpringConfiguration es que solo puede haber una en la aplicación

@ConponentScan : Se encarga de revisar los paquetes actuales y registrar de forma automática cualquier @Servie @Repository @Controller etc que la aplicación tenga de forma totalmente transparente para Spring Framework.

Así pues podemos tener un c´ódigo como el siguiente y Spring Boot será capaz de inicializarlo todo de forma bastante rápida. En primer lugar escribimos un bean

```
package com.arquitecturajava.springboot;

public class Persona {

	private String nombre;
	private String apellidos;
	private int edad;
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getApellidos() {
		return apellidos;
	}
```

```
	public void setApellidos(String apellidos) {
		this.apellidos = apellidos;
	}
	public int getEdad() {
		return edad;
	}
	public void setEdad(int edad) {
		this.edad = edad;
	}
	public Persona(String nombre, String apellidos, int edad) {
		super();
		this.nombre = nombre;
		this.apellidos = apellidos;
		this.edad = edad;
	}
	@Override
	public String toString() {
		return "Persona [nombre=" + nombre + ", apellidos=" +
apellidos + ", edad=" + edad + "]";
	}
}
```

Ahora creamos un servicio que automaticamente será registrado por @ComponentScan ya que dispone de la anotación @Service

```
package com.arquitecturajava.springboot;

import java.util.Arrays;
import java.util.List;

import org.springframework.stereotype.Service;
```

```
@Service
public class PersonaService {

	public List<Persona> buscarTodos() {
		return Arrays.asList(new Persona ("pepe","perez",20));
	}
}
```

Por último registramos un Bean en el fichero de SpringBoot ya que es un fichero de configuración y permite dar de alta Beans a medida:

```
package com.arquitecturajava.springboot;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class SpringbootApplication implements CommandLineRunner {

	@Autowired
	PersonaService servicio;
	@Autowired
	Persona persona;
	public static void main(String[] args) {
		SpringApplication.run(SpringbootApplication.class,
args);
	}

	@Override
```

```
        public void run(String... args) throws Exception {
servicio.buscarTodos().stream().forEach(System.out::println);
                System.out.println(persona);
        }
        @Bean
        public Persona Persona() {
                return new Persona("ana","gomez",30);
        }
}
```

```
package com.arquitecturajava.springboot;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class SpringbootApplication implements CommandLineRunner {

        @Autowired
        PersonaService servicio;
        @Autowired
        Persona persona;
        public static void main(String[] args) {
                SpringApplication.run(SpringbootApplication.class,
args);
        }
```

```
        @Override
        public void run(String... args) throws Exception {
servicio.buscarTodos().stream().forEach(System.out::println);
                System.out.println(persona);
        }
        @Bean
        public Persona Persona() {
                return new Persona("ana","gomez",30);
        }
}
```

Spring Boot ha podido inyectar de forma automática el PersonaService ya que dispone de la anotación service y por lo tanto con @ComponentScan será localizado una anotación de la que hereda @SpringBootApplication de la misma manera sucederá con el Objeto Persona (“ana”,”gomez”,30) ya que esta dado de alta en el propio fichero de SpringBootApplication y al heredar de @Configuration , lo tendremos a nuestra disposición . Únicamente nos queda ejecutar la aplicación como linea de comandos y ver como Spring Boot imprime los datos por la consola.



Esta es la forma en que Spring Boot funciona a nivel de Aplicaciones con la anotación de @SpringBootApplication

Otros artículos relacionados

- Spring Boot REST JPA y JSON
- Spring Boot JDBC y su configuración
- Spring Boot Load Data y testing
- Spring Boot Starter ,un concepto fundamental

- Spring Initializer