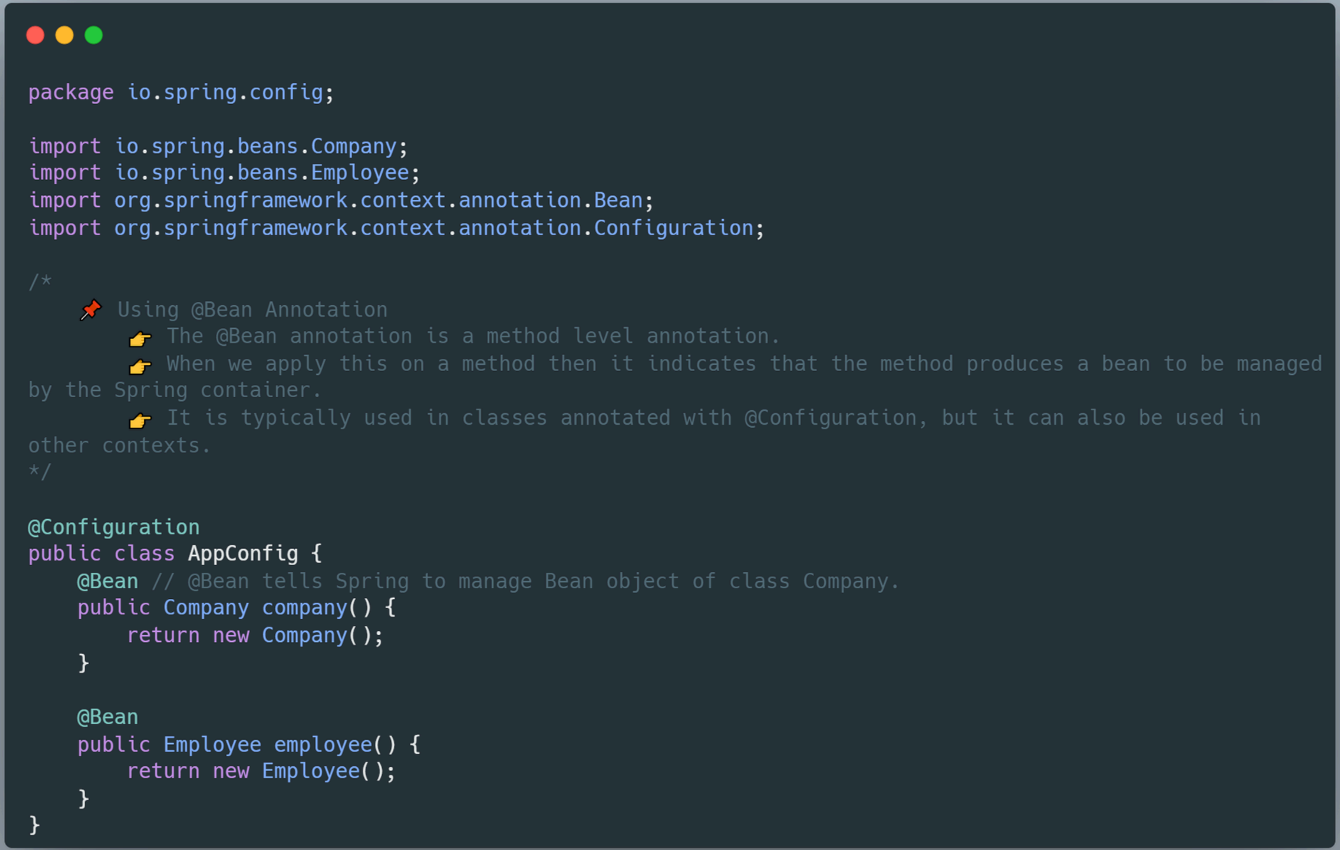
Comparison between @Bean and @Component in Spring Framework

We will explore the differences and use cases of the @Bean and @Component annotations in the Spring Framework.

What is @Bean?

The @Bean annotation is a powerful feature in the Spring framework that allows developers to define and manage beans, which are the fundamental building blocks of a Spring-based application. When applied to a method, the @Bean annotation instructs the Spring container to treat the method as a factory for creating and managing a specific bean instance.

Using @Bean Annotation

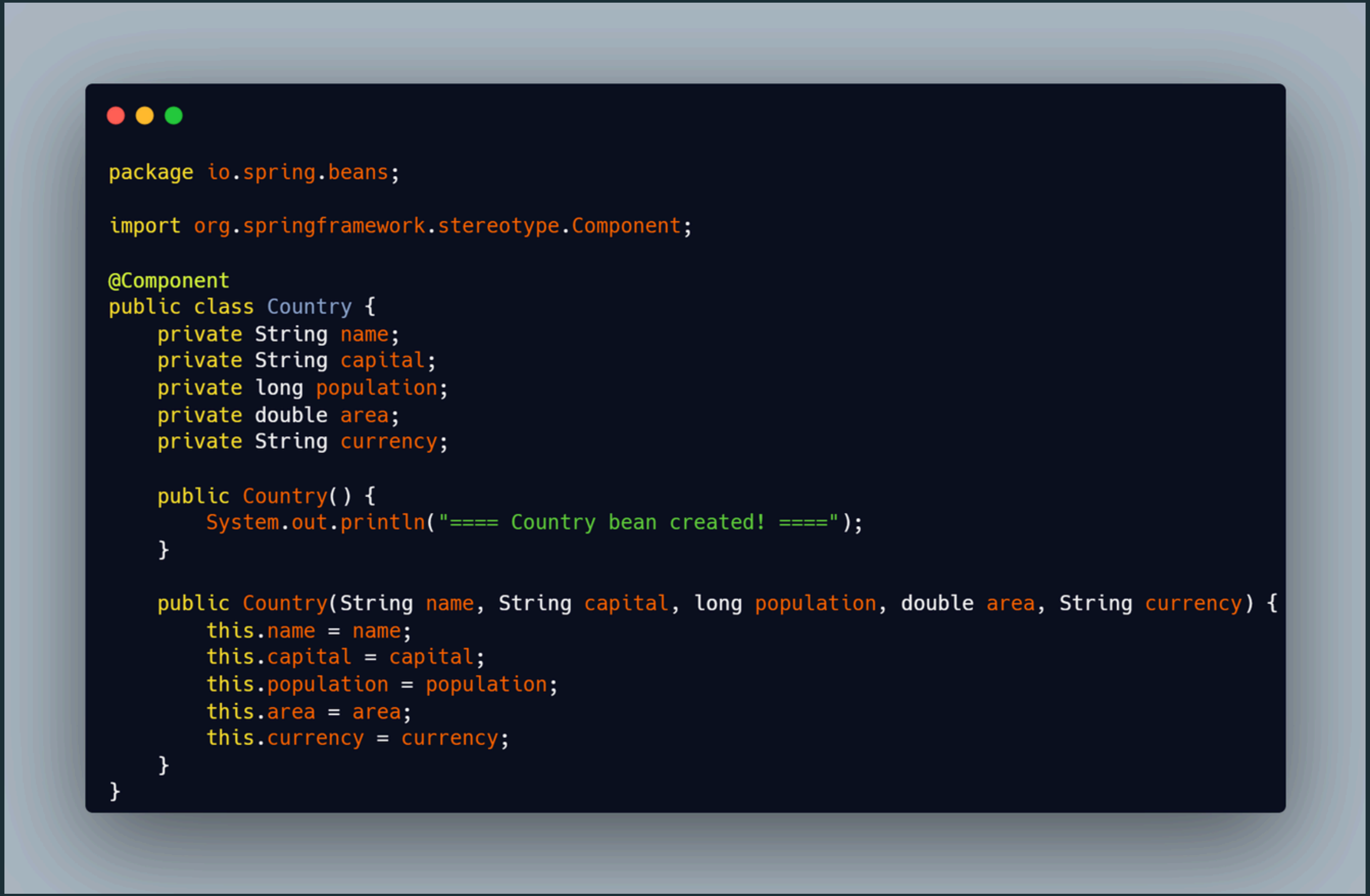


```java
package io.spring.config;

import io.spring.beans.Company;
import io.spring.beans.Employee;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/*
    📌 Using @Bean Annotation
        👉 The @Bean annotation is a method level annotation.
        👉 When we apply this on a method then it indicates that the method produces a bean to be managed
by the Spring container.
        👉 It is typically used in classes annotated with @Configuration, but it can also be used in
other contexts.
*/

@Configuration
public class AppConfig {
    @Bean // @Bean tells Spring to manage Bean object of class Company.
    public Company company() {
        return new Company();
    }

    @Bean
    public Employee employee() {
        return new Employee();
    }
}
```

What is @Component?

The @Component annotation is used to mark a class as a Spring-managed component, indicating that the Spring container is responsible for managing its lifecycle. When a class is annotated with @Component, the Spring runtime automatically detects and registers it, allowing other components to be injected into it and the component to be used throughout the application.

Using @Component Annotation



```java
package io.spring.beans;

import org.springframework.stereotype.Component;

@Component
public class Country {
    private String name;
    private String capital;
    private long population;
    private double area;
    private String currency;

    public Country() {
        System.out.println("==== Country bean created! ====");
    }

    public Country(String name, String capital, long population, double area, String currency) {
        this.name = name;
        this.capital = capital;
        this.population = population;
        this.area = area;
        this.currency = currency;
    }
}
```

Differences

@Bean vs @Component

@Bean is usually declared in configuration classes (classes annotated with @Configuration), while @Component is a class-level annotation.

Use Cases

@Bean is ideal for external libraries or classes that are not annotated with Spring annotations, while @Component is ideal for defining our own classes as Spring-managed components.

In summary, the main differences between @Bean and @Component are where they are declared and their intended use cases. @Bean is more suited for external dependencies, while @Component is better suited for defining your own Spring-managed components.

Pros and Cons

More Code, More Control

@Bean requires more manual configuration and boilerplate code, but provides granular control over bean instantiation and configuration.

Simplified Configuration

@Component leverages component scanning to automatically discover and register beans, reducing the amount of boilerplate code required.

Use Cases

Complex Beans with Method Calls and Dependencies

Use @Bean annotation for beans that require complex configuration, such as making method calls or depending on other beans.

Simple Beans without Special Configuration

Use @Component annotation for simpler beans that don't need any special initialization or configuration.

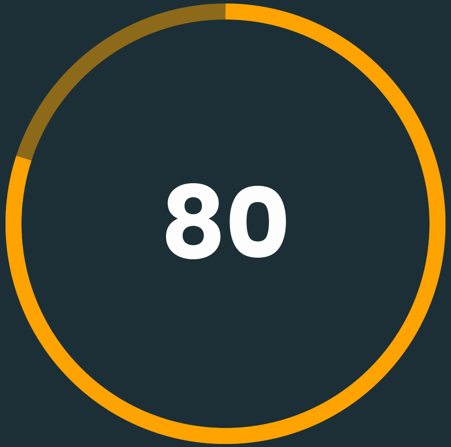
In summary, use @Bean for complex beans that require special setup, and @Component for simpler beans that don't need custom configuration.

Summary

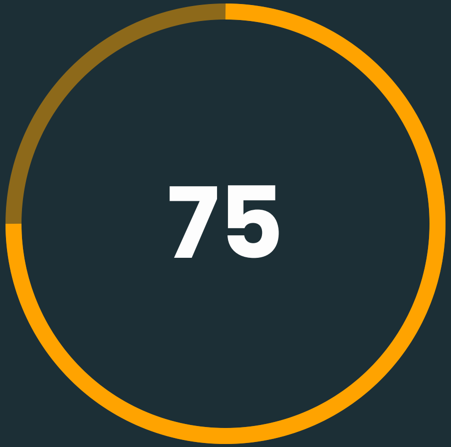
The use of @Bean and @Component annotations in Spring Framework is a crucial concept for managing the configuration of beans. @Bean is primarily used for explicit bean configuration, allowing developers to have more control over the creation and setup of beans. In contrast, @Component is used for automatic detection and configuration of beans, simplifying the overall configuration process.

Conclusion

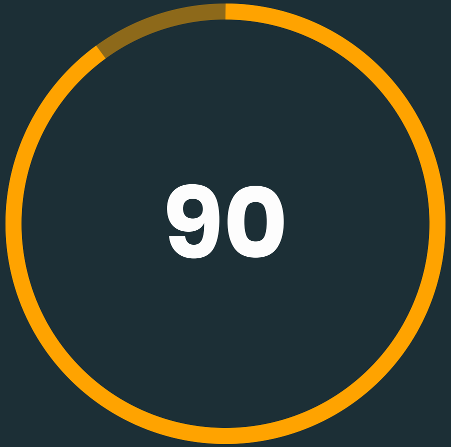
Comparison of @Bean and @Component in terms of complexity and level of control required



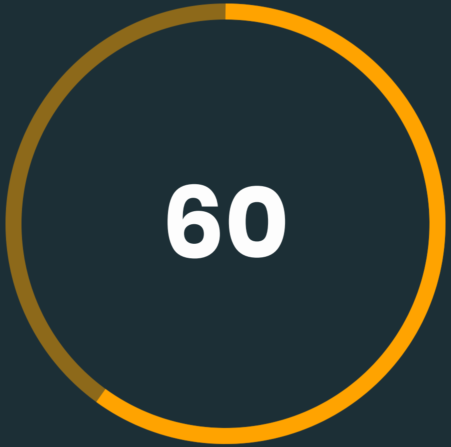
Complexity of the Bean



Level of Control Required



@Bean



@Component

Stay Tuned for More Coding Magic!

⚡ “Level up your coding journey! Follow, learn, and let’s build amazing things together!”

LinkedIn: https://www.linkedin.com/in/eyeganeshgupta/

GitHub: https://github.com/eyeganeshgupta

Portfolio: https://about-ganesh.vercel.app/