Difference between @Component vs @Bean annotations in Spring Framework?

JAVINPAUL AND SOMA
DEC 05, 2023 · PAID

Hello folks, if you are preparing for Java Developer and Spring Developer interviews then you must prepare for questions like the difference between X and Y like **@bean vs @component**.

In Spring, both `@Component` and `@Bean` annotations are used to register a bean in the Spring container, but they differ in the way they are used which you will find in this article.

Now, coming back to the question, `@Component` is a class-level annotation that is used to mark a class as a Spring component, indicating that it should be automatically detected and registered as a bean in the Spring application context. The `@Component` **annotation is a generic stereotype annotation that can be used to annotate any class.**

On the other hand, `@Bean` is a method-level annotation that is used to explicitly declare a bean definition in a configuration class. It is typically used when a class does not have the `@Component` annotation, or when you want more fine-grained control over the configuration of the bean.

While `@Component` is used for automatic scanning and registering of beans, `@Bean` and is used for explicit manual registration of beans. Another difference is that `@Component` is used for general classes while `@Bean` is used for creating a specific instance of a class and providing it to the Spring container.

In short, `@Component` is used to annotate a class and automatically detect and register it as a bean, whereas `@Bean` is used to declare a method that provides an instance of a bean to the Spring container.

Now, that you know the basic difference, it's time for a deep dive.

What is @Component in Spring?

in Spring, `@Component` is a generic annotation that can be used to indicate any Spring-managed component or bean. It serves as a base annotation for more specific annotations such as `@Service`, `@Repository`, and `@Controller`.

Here’s an example of how `@Component` can be used to mark a class as a Spring-managed component:

```
@Component
public class Order {
  // class implementation
}
```

By adding the `@Component` annotation to the `Order` class, Spring will automatically detect and manage this class as a bean within the application context. This allows it to be easily injected into other Spring-managed components using dependency injection.

Here’s an example of how a `@Component` can be used in conjunction with dependency injection:

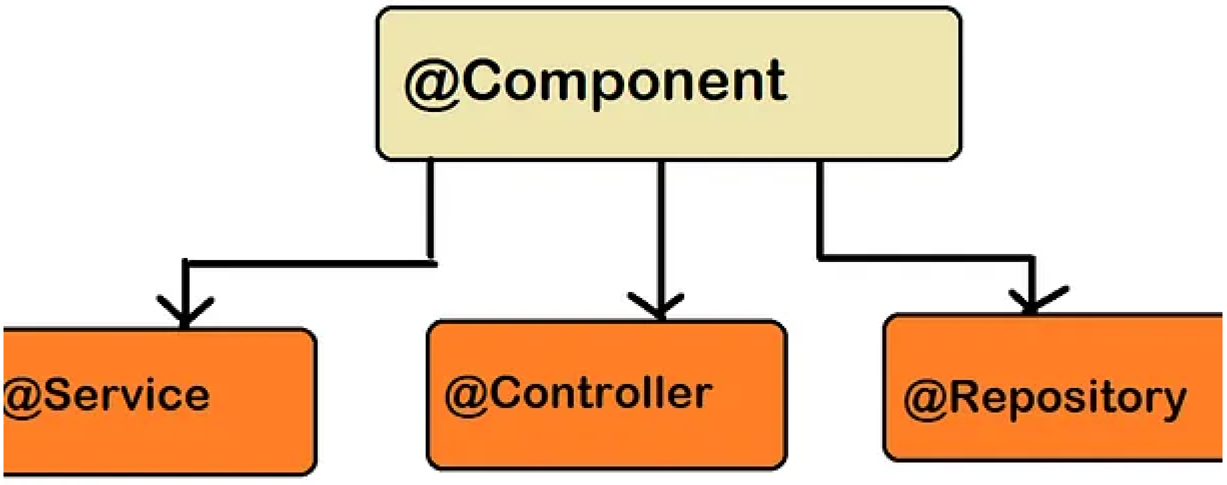
```
@Component
public class Invoic{
  private final Order order;

  @Autowired
  public Invoice(Order order) {
    this.order = order;
  }

  // class implementation
}
```

In this example, `Invoice` is also a Spring-managed component that has a dependency on `Order` . By using the `@Autowired` annotation on the constructor, Spring will automatically inject an instance of `Order` into `Invoice`at runtime.

Overall, `@Component` is a powerful annotation that allows for easy management and dependency injection of Spring components.



What is @Bean in a Spring Framework?

In Spring, `@Bean` is an annotation that is used to declare a single bean. It is generally used to configure third-party beans or beans that are not defined in the application context.

Here's an example of using `@Bean` to declare a single bean:

```
@Configuration
public class AppConfig {

    @Bean
    public MyService myService() {
        return new MyServiceImpl();
    }
}
```

In this example, `@Bean` is used to create a bean for the `MyService` interface. The method `myService()` returns a new instance of `MyServiceImpl`, which is then registered as a bean in the application context.

`@Bean` can also be used to specify bean dependencies. Here's an example:

```
@Configuration
public class AppConfig {

    @Bean
    public MyService myService() {
        return new MyServiceImpl();
    }
    @Bean
    public MyController myController(MyService myService) {
        return new MyControllerImpl(myService);
    }
}
```

In this example, `@Bean` is used to declare a bean for the `MyController` interface, which has a dependency on `MyService`. The `myController()` the method takes a parameter of `MyService` and uses it to create a new instance of `MyControllerImpl`, which is then registered as a bean in the application context.

The `MyService` bean is automatically injected into the `MyController` bean because of its dependency declaration in the method signature.

Difference between @Bean and @Component annotation in Spring Framework?

Here are the key differences between `@Bean` and `@Component` annotations in Spring:

1. **Functionality**

- `@Bean` is used to explicitly declare a bean in Spring's application context.
- `@Component` is a general-purpose annotation used to indicate a Spring-managed component.

2. Scope

- `@Bean` is used to define custom bean instantiation logic, and can be used to configure the scope of a bean.
- `@Component` doesn't define the scope of the bean, it uses the default scope of `Singleton`.

3. Usage:

- `@Bean` is typically used in configuration classes that are annotated with `@Configuration` or `@Import`.
- `@Component` is typically used to mark classes as Spring-managed components that are picked up automatically by component scanning.

4. Return type

- `@Bean` methods can return any object, whereas `@Component` is typically used to annotate classes that should be instantiated as beans.

5. **Configuration**

- `@Bean` is used to configure specific instances of a bean or to inject dependencies between beans.
- `@Component` is used to mark a class as a Spring-managed component, and is used to auto-detect and configure beans in the application context.

In summary, `@Bean` **is more flexible and is used to create custom bean instantiation logic** and configure specific instances of a bean, whereas `@Component` is used to mark classes as Spring-managed components and auto-detect and configure beans in the application context.

That's all about the **difference between @Bean and @Component annotation in Spring Framework**. Even though, both `@Bean` and `@Component` are important annotations in Spring and are used to manage beans in a Spring application context.

The key difference between the two is that `@Component` **is used to auto-detect and configure beans while** `@Bean` **is used to explicitly declare beans.**

If you have a third-party class that you cannot modify and would like to declare it as a bean, you can use the `@Bean` annotation. On the other hand, if you have control over the class and want to make it a Spring bean, you can use the `@Component` annotation.

In general, the choice of which annotation to use depends on the situation and the level of control you need over the bean's creation and management. It is important to understand the difference between @Bean and `@Component` to make the best use of them in your Spring applications.

By understanding the differences between these annotations, you will be better prepared for your Java and Spring interview and have a deeper understanding of the Spring framework.

And, if you are preparing for Java interviews you can also check my **Java + Spring Interview + SQL Bundle on GUmroad**, use code **friends20** to get a 20% discount also

Java and React Developer

Comments

Write a comment...