**Comparing @Component vs @Bean**

|  |  |  |
| --- | --- | --- |
| **Heading** | **@Component** | **@Bean** |
| Where? | Can be used in any java class | Typically used on methods in spring Configuration classes. |
| Ease of use | Very easy. Just add an annotation | You write all the code. |
| Autowiring | Field, Setter or Constructor injection | Method calls or parameters |
| Who creates beans? | Spring Framework | You write bean creation code |
| Recommended for | Instantiating Beans for your own application code | 1. Custom Business logic. 2. Instantiating Beans for 3rd party libraries |
| Purpose | **@Component** is used to mark a class as a Spring-managed component, indicating that it should be automatically detected and registered as a bean in the application context. | **@Bean**, on the other hand, is used to explicitly declare a bean in the configuration class. |
| Usage | @Component can be used on a class. | @Bean is used on a method within a configuration class that creates and returns a bean. |
| Naming | When using **@Component**, Spring will generate a bean name by default, based on the class name (with the first letter in lower case). | With **@Bean**, the name of the bean is specified by the method name by default, but it can be customized using the **name** attribute. |
| Scope | **@Component** does not allow specifying a custom scope for a bean. It uses the default scope of singleton. | **@Bean** allows customizing the scope of a bean by specifying the **scope** attribute. |

**Singleton Scope**: the singleton scope is one of the bean scopes that can be used to control the lifecycle of Spring-managed beans. When a bean is defined with singleton scope, Spring creates a single instance of the bean for the entire application context and shares that single instance among all the clients requesting that bean.