Spring @ComponentScan Annotation with Example

Last Updated : 23 Jul, 2025

Spring is one of the most popular frameworks for building enterprise-level Java applications. It is an open-source, lightweight framework that simplifies the development of robust, scalable, and maintainable applications. Spring provides various features such as Dependency Injection (DI), Aspect-Oriented Programming (AOP), and support for Plain Old Java Objects (POJOs), making it a preferred choice for Java developers.

In this article, we will focus on the **@ComponentScan annotation** in Spring.

@ComponentScan Annotation in Spring

The **@ComponentScan** annotation is used to specify the package that the framework should scan for Spring-managed components. These components are classes annotated with **@Component, @Service, @Repository, or @Controller.** When Spring finds such classes, it automatically registers them as beans in the Spring application context.

The **@ComponentScan annotation is used along with the @Configuration annotation**. It eliminates the need for manual bean registration in XML files, making the configuration more concise and easier to manage.

Key Points about @ComponentScan:

- The @ComponentScan annotation tells Spring to scan specific packages for components annotated with @Component, @Service, @Repository, or @Controller.
- Without arguments, the @ComponentScan annotation scans the package of the annotated class and its sub-packages by default.

- The @ComponentScan annotation is used alongside @Configuration to define package scanning in a configuration class.
- The @ComponentScan annotation helps spring automatically detect and register beans without needing manual bean declaration.

Steps to Use the @ComponentScan Annotation

We will create a simple Spring application with a College class and configure it using @ComponentScan.

Step 1: Create the College Class

Let's create a simple Collge Class with a method test() that prints a message.

```
// Creating a class
package ComponentAnnotation;

public class College {
    public void test() {
        System.out.println("Test College Method");
    }
}
```

Now let’s create a Bean for this class inside the beans.xml file.

Bean.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans/"
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context/"
    xsi:schemaLocation="http://www.springframework.org/schema/beans/
        https://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context/
        https://www.springframework.org/schema/context/spring-context.xsd">

    <bean id="collegeBean" class="ComponentAnnotation.College"></bean>
</beans>
```

Step 2: Add @Component

Instead of defining the College bean in an XML file, we can use the **@Component annotation** to mark the class as a Spring-managed component.

```
// Java Program to Illustrate Component Annotation
// Indulgence in College Class

package ComponentAnnotation;
import org.springframework.stereotype.Component;

// Registering the class as a Spring bean
@Component("collegeBean")
public class College {
    public void test() {
        System.out.println("Test College Method");
    }
}
```

Transitioning from XML to Annotation-Based Configuration

Instead of using an XML file **(beans.xml)** to define beans, Spring allows us to use Java-based configuration. By creating a configuration class annotated with @Configuration and using @ComponentScan, we can replace the XML configuration with a more modern and concise approach. This eliminates the need for beans.xml and makes the application easier to manage.

Step 3: Create a Configuration Class

We create a configuration class (CollegeConfig) and use the @ComponentScan annotation to specify the package to scan for components.

```
// Java Program to Illustrate Configuration of
// College Class

package ComponentAnnotation;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
// Marks this class as a configuration class
@Configuration
// Specifies the package to scan
@ComponentScan(basePackages = "ComponentAnnotation")
```

```
public class CollegeConfig {
}
```

Step 4: Create the Main Application

Now to check our application, we can create a **Main class** to load the Spring application context and retrieve the **College** bean.

```
//Creating a Main class
package ComponentAnnotation;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {
    public static void main(String[] args) {
        // Load the Spring application context using the configuration class
        ApplicationContext context = new
AnnotationConfigApplicationContext(CollegeConfig.class);

        // Retrieve the College bean
        College college = context.getBean(College.class);

        // Call the test method
        college.test();
    }
}
```

Step 5: Run the Application

When we run the Main class, the output will be:

> *Test College Method*

Comment More info Advertise with us