**Spring @Component Annotation Overview**

The @Component annotation indicates that an annotated class is a "component". Such classes are considered candidates for auto-detection when using annotation-based configuration and classpath scanning.

In short, @Component is a class-level annotation. During the component scan, Spring Framework automatically detects classes annotated with *@Component* annotation and creates Spring beans for those classes.

**Component Scanning**

Spring can automatically scan a package for beans if component scanning is enabled.  
  
**@ComponentScan** configures which packages to scan for classes with annotation configuration. We can specify the base package names directly with one of the *basePackages* or *value* arguments (value is an alias for basePackages):

@Configuration

@ComponentScan(basePackages = "com.javaguides.annotations")

class UserConfig {}

**Spring @Component Annotation Example**

Let’s create a very simple Spring boot maven application to showcase the use of Spring *@Component* annotation and how Spring autodetects it with annotation-based configuration and classpath scanning.

Add below Spring boot starter web dependency:

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

**Create Spring Component class - ComponentDemo.java**

Let’s create a simple component class and mark it with *@Component*annotation.

@Component

class ComponentDemo{

public String getValue() {

return "Hello World";

}

}

Spring Container will automatically create and manage the spring bean for the above class because it is annotated with *@Component* annotation.

**Running Spring Boot Application**

Note that we have created **ApplicationContext** and retrived **ComponentDemo** bean using **getBean()** Method.

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ConfigurableApplicationContext;

import org.springframework.stereotype.Component;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

ConfigurableApplicationContext applicationContext = SpringApplication.run(DemoApplication.class, args);

ComponentDemo componentDemo = (ComponentDemo) applicationContext.getBean("componentDemo");

System.out.println(componentDemo.getValue());

}

}

@Component

class ComponentDemo {

public String getValue() {

return "Hello World";

}

}

**Output:**

**[](https://3.bp.blogspot.com/-OAl96Jllye8/W-vatIuVC_I/AAAAAAAAEs8/Uu64TgRNwq83wR-6chtjDS6s7eMejA7hQCLcBGAs/s1600/compont-output.PNG)**

By default, the bean instances of this class have the same name as the class name with a lowercase initial. On top of that, we can specify a different name using the optional value argument of this annotation.

@Component("componentD

[Spring Data JPA - Query Creation from Method Names (javaguides.net)](https://www.javaguides.net/2018/11/spring-data-jpa-query-creation-from-method-names.html)

**Query method to find or retrieve a unique product by name:**

/\*\*

\* Return the distinct product entry whose name is given as a method parameter

\* If no product entry is found, this method returns null.

\*/

Product findDistinctByName(String name);

**Query method to find or retrieve products whose price is greater than the given price as a method parameter:**

/\*\*

\* Return the products whose price is greater than given price as method parameter

\* @param price

\* @return

\*/

List<Product> findByPriceGreaterThan(BigDecimal price);

**Query method to find or retrieve products whose price is less than given price as a method parameter:**

/\*\*

\* Return the products whose price is less than given price as method parameter

\* @param price

\* @return

\*/

List<Product> findByPriceLessThan(BigDecimal price);

**Query method to find or retrieve filtered products that match the given text ( contains check):**

/\*\*

\* Return the filtered the product records that match the given text

\* @param name

\* @return

\*/

List<Product> findByNameContaining(String name);

**Query method to find or retrieve products for a specified pattern in a column ( SQL LIKE condition):**

/\*\*

\* Return products based on SQL like condition

\* @param name

\* @return

\*/

List<Product> findByNameLike(String name);

**Query method to find or retrieve products based on the price range ( start price and end price):**

/\*\*

\* Returns a products whose price between start price and end price

\* @param startPrice

\* @param endPrice

\* @return

\*/

List<Product> findByPriceBetween(BigDecimal startPrice, BigDecimal endPrice);