Until now, for creating bean object we were using a tag called <bean> inside <beans> tag. But we have a alternative annotation for do the same. This annotation name is **@Component.**

Let’s look usage,

**Student.java :**

@Component("student")

public class Student {

}

We want the Student class object, so first step is to annotate it with @Component.

@Component takes a argument which represent the **id** for the bean object. This argument is optional. If we don’t provide any argument then by default the bean object id becomes the name of class in camel notation. For ex : if my class name is **MyStudent** then the id for it will be **myStudent.**

**config.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">

<context:component-scan base-package="com.component.annotation" />

</beans>

As you can see, we have dropped the <bean> tag and add a new tag i.e.

<context:component-scan base-package="com.component.annotation" />

When spring watch this tag inside config.xml, then spring will start scanning all the classes present in the **base-package** mentioned in the tag.

If spring found any of class annotate with **@Component** annotation. So spring creates its object and push it into IOC container.

**School.java**

public class School {

public static void main(String[] args) {

String path = "com/component/annotation/config.xml";

ApplicationContext context = new ClassPathXmlApplicationContext(path);

System.out.println("Object created ");

Student stu = context.getBean("student", Student.class);

System.out.println(stu);

}

}