In this tutorial, we are going to completely drop **config.xml** file using **@Configuration** and **@ComponentScan** Annotation.

Previously, there is only one tag were remaining in the config.xml for scan the base package. If we able to scan through java’s annotation we can completely drop the config.xml.

Let’s understand first what @Configuration annotation will do for us,

As we are going to completely drop config.xml, but in the place of config.xml file we have to create a new Java file which will act as a configuration file for us.

To make that Java file act as a configuration file will be responsibility of **@Configuration.**

Let’s create java configuration file,

**StudentConfig.java**

@Configuration

@ComponentScan(basePackages="com.configurationannotation.usage")

public class StudentConfig {

}

Now this .java file will act as a config.xml file. But for let spring know that our configuration file is not an .xml file rather it is a .java file we will have use the below Class object

new AnnotationConfigApplicationContext(StudentConfig.class);

rather,

new ClassPathXmlApplicationContext("com/component/annotation/config.xml");

Now let’s talk about **@ComponentScan** annotation, so it’s work is as similar as below config.xml tag,

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

Both of them is use for tell to spring that scan the base package provided in it, and if there is any class have @Component annotation found by spring while scanning then spring creates its object and push it into the IOC container.

Let’s look usage,

**Student.java :**

@Component("student")

public class Student {

}

This class is going to be instantiate by spring while scanning.

**School.java**

public class School {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(StudentConfig.class);

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

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

System.out.println(stu);

}

}