Spring @Component, @Service, @Repository, @Controller Difference

Spring @Component, @Service, @Repository and @Controller annotations are used for automatic bean detection using classpath scan in Spring framework. @Component is a generic annotation. Difference of @Service, @Repository, @Controller with @Component is they are special cases of @Component and used for particular purposes. The difference is just classification only.

For all these annotations (stereotypes), technically the core purpose is same. Spring automatically scans and identifies all these classes that are annotated with " ***@Component, @Service, @Repository, @Controller***" and registers BeanDefinition with ApplicationContext. We read about @Controller in a previous Spring tutorial on annotation based controllers.

Difference Between @Component, @Service, @Repository and @Controller

- Major difference between these stereotypes is they are used for different classification.
- In a multitier application, we will have different layers like presentation, service, business, data access etc. When a class is to be annotated for auto-detection by Spring, then we should use the respective stereotype as below.
    - @Component – generic and can be used across application.
    - @Service – annotate classes at service layer level.
    - @Controller – annotate classes at presentation layers level, mainly used in Spring MVC.
    - @Repository – annotate classes at persistence layer, which will act as database repository.
- If technically they are going to be same then why do we need to use these at different layers level. Why not use the same at all layers. For example, if we use @Service in all layers, all the beans will get instantiated and no issues. There is a minor difference, for example consider @Repository.

  The postprocessor automatically looks for all exception translators (implementations of the PersistenceExceptionTranslator interface) and advises all beans marked with the @Repository annotation so that the discovered translators can intercept and apply the appropriate translation on the thrown exceptions.

-
- Similar to the above, in future Spring may choose to add value for @Service, @Controller and @Repository based on their layering conventions. To that additional feature advantage its better to respect the convention and use them in line with layers.
- Other than the above, with respect to scan-auto-detection, dependency injection for BeanDefinition @Component, @Service, @Repository, @Controller are same.

Spring Configuration for Component-scan

For these beans to be instantiated by Spring, we need to have the following configuration in the spring configuration XML. Assuming com.javapapers.spring is a base package containing these classes. Needless to say, these Java classes should be part of the application classpath.

```
<?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-3.0.xsd
http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-
context-3.0.xsd">
```

```
 <context:component-scan base-package="com.javapapers.spring"/>

</beans>
```

Disable automatic scan of @Component, @Repository, @Service, @Controller:

This automatic scan behavior can be disabled for the default stereotypes by setting the *use-default-filters* property to false,

```
<beans>



<context:component-scan use-default-filters = "false" base-package="com.javapapers.spring" />



</beans>
```

Customize Default Spring Scan Behavior

When component-scan is enable, by default spring scans for @Component, @Service, @Repository and @Controller stereotypes only. All the classes present in the classpath coming under the base-package annotated with these stereotypes will be auto-detected. We can modify this default Spring behavior using ***include-filter*** or ***exclude-filter*** attribute of the component-scan attribute. There are five filter types available 'annotation, assignable, aspectj, regex, custom' and they can be used as below,

```
<beans>



   <context:component-scan base-package="com.javapapers.spring">

      <context:include-filter type="regex" expression=".*Stub.*Repository"/>

      <context:exclude-filter type="annotation"

                              expression="org.springframework.stereotype.Repository"/>

      <context:include-filter type="assignable" expression="com.javapapers.spring.AnimalService"/>

   </context:component-scan>



</beans>
```

Bean Naming

For all @Component, @Service, @Repository and @Controller stereotyped components the bean name is assigned based on *BeanNameGenerator*strategy. We can also supply our name choice during annotation and that will take high precedence.

```
@Service("lionKing")

public class LionService {
```

```
  // ...

}



@Repository

public class AnimalJpa {

  // ...

}
```

For *LionService* the instantiated bean name will be *lionKing* which we have supplied, but in the case of AnimalJpa the bean name will be animalJpa. This behavior is same for all sterotypes @Component, @Service, @Repository and @Controller.