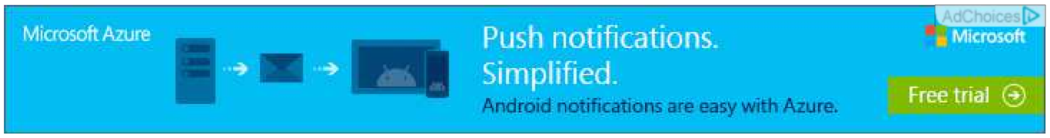
1 help

What's the difference between @Component, @Repository & @Service annotations in Spring?



Can `@Component` , `@Repository` & `@Service` annotations be used interchangeably in Spring or do they provide any particular functionality besides acting as a notation device?

In other words, if I have a Service class and I change the annotation from `@Service` to `@Component` , will it still behave the same way?

Or does the annotation also influence the behavior and functionality of the class?

java spring spring-mvc annotations

asked Jul 26 '11 at 9:10
Colin McCree
1,311 3 6 3

3 This question was never answered, or are you waiting for better explanation? – stviper Jun 27 at 20:20

add a comment

8 Answers

From Spring Documentation:

> In Spring 2.0 and later, the @Repository annotation is a marker for any class that fulfills the role or stereotype (also known as Data Access Object or DAO) of a repository. Among the uses of this marker is the automatic translation of exceptions.
>
> Spring 2.5 introduces further stereotype annotations: @Component, @Service, and @Controller. @Component is a generic stereotype for any Spring-managed component. @Repository, @Service, and @Controller are specializations of @Component for more specific use cases, for example, in the persistence, service, and presentation layers, respectively.
>
> Therefore, you can annotate your component classes with @Component, but by annotating them with @Repository, @Service, or @Controller instead, your classes are more properly suited for processing by tools or associating with aspects. For example, these stereotype annotations make ideal targets for pointcuts.
>
> Thus, if you are choosing between using @Component or @Service for your service layer, @Service is clearly the better choice. Similarly, as stated above, @Repository is already supported as a marker for automatic exception translation in your persistence layer.

```
| Annotation | Meaning                                             |
+------------+-----------------------------------------------------+
| @Component | generic stereotype for any Spring-managed component |
| @Repository| stereotype for persistence layer                    |
| @Service   | stereotype for service layer                        |
| @Controller| stereotype for presentation layer (spring-mvc)      |
```

edited Oct 27 at 16:36
Umpa
47 4

answered Aug 1 '11 at 10:20
stivlo
31.7k 18 75 132

Would it make sense to add @Controller (or @Component) to an @WebServlet? It's not a Spring MVC controller, but that's the conceptually closest match. What about servlet filters? – Rick Mar 11 at 10:08

add a comment

They are almost the same - all of them mean that the class is a Spring bean. `@Service` , `@Repository` and `@Controller` are specialized `@Component` s. You can choose to perform specific actions with them. For example:

- `@Controller` beans are used by spring-mvc
- `@Repository` beans are eligible for persistence exception translation

Another thing is that you designate the components semantically to different layers.

One thing that `@Component` offers is that you can annotate other annotations with it, and then use them the same way as `@Service` .

For example recently I made:

```
@Component
@Scope("prototype")
public @interface ScheduledJob {..}
```

So all classes annotated with `@ScheduledJob` are spring beans and in addition to that are registered as quartz jobs. You just have to provide code that handles the specific annotation.

edited Aug 1 '11 at 12:20

answered Jul 26 '11 at 9:16
Bozho
289k 47 559 756

1 Great answer. Really helped me understand. :-) – cbmeeks Sep 18 '12 at 14:37

1 @Component means a spring bean only, is any other purpose for it ? – kapil das Oct 18 '13 at 13:08

add a comment

Spring 2.5 introduces further stereotype annotations: @Component, @Service and @Controller. @Component serves as a generic stereotype for any Spring-managed component; whereas, @Repository, @Service, and @Controller serve as specializations of @Component for more specific use cases (e.g., in the persistence, service, and presentation layers, respectively). What this means is that you can annotate your component classes with @Component, but by annotating them with @Repository, @Service, or @Controller instead, your classes are more properly suited for processing by tools or associating with aspects. For example, these stereotype annotations make ideal targets for pointcuts. Of course, it is also possible that @Repository, @Service, and @Controller may carry additional semantics in future releases of the Spring Framework. Thus, if you are making a decision between using @Component or @Service for your service layer, @Service is clearly the better choice. Similarly, as stated above, @Repository is already supported as a marker for automatic exception translation in your persistence layer.

```
@Component - Indicates a auto scan component.
@Repository - Indicates DAO component in the persistence layer.
@Service - Indicates a Service component in the business layer.
@Controller - Indicates a controller component in the presentation layer.
```

reference :- http://static.springsource.org/spring/docs/3.0.0.M3/reference/html/ch04s12.html

edited Sep 26 '13 at 17:36
Luiggi Mendoza
47.6k 6 32 81

answered May 15 '13 at 12:48
Ajit Singh
416 4 8

add a comment

@Component is equivalent to

```
<bean>
```

@Service, @Controller , @Repository = {@Component + some more special functionality}

That mean Service,Controller and Repository are functionally the same.

The three annotations are used to separate **"Layers"** in your application,

- Controllers just do stuff like dispatching, forwarding, calling service methods etc.
- Service Hold business Logic, Calculations etc.
- Repository are the DAOs(Data Access Objects), they access the database directly.

Now you may ask why separate them:(I assume you know AOP-Aspect Oriented Programming)

Lets say you want to Monitors the Activity of the DAO Layer only. You will write an Aspect(A class) class that does some logging before and after every method of your DAO is invoked, you are able to do that using AOP as you have three distinct Layers and are not mixed.

So you can do logging of DAO "around", "before" or "after" the DAO methods. You could do that because you had a DAO in the first place. What you just achieved is **Separation of concerns or tasks.**

Imagine if there were only one annotation @Controller, then this component will have dispatching, business logic and accessing database all mixed, so dirty code!

Above mentioned is one very common scenario, there are many more use cases of why to use three annotations.

answered May 23 at 5:15
Oliver
561 4 15

1 short and simple – coding_idiot Jun 3 at 0:44

3 An excellent answer for beginners of spring. +1 for being straight to the point in a very simple manner. – Black Panther Aug 19 at 12:14

add a comment

Use of `@Service` and `@Repository` annotations are important from database connection perspective.

1. Use `@Service` for all your web service type of DB connections
2. Use `@Repository` for all your stored proc DB connections

If you do not use the proper annotations, you may face commit exceptions overridden by rollback transactions. You will see exceptions during stress load test that is related to roll back JDBC transactions.

edited Nov 2 '12 at 16:27   answered Nov 2 '12 at 16:05

Wahyu Kristianto
1,671 2 18 42

Das
221 2 2

add a comment

In Spring `@Component` , `@Service` , and `@Controller` . `@Component` is a Stereotype annotations which is user for:

`@controller:` where your **request mapping from presentation page** done i.e. Presentation page won't go to any other file it goes directly to `@controller` class and check for requested path in `@RequestMapping` annotation which written before method calls if necessary.

`@service` : is annotation of business layer in which our user not directly call persistence method so it will call this methods using this annotation. **It will request @repository as per user request**

`@repository` : used to get data from database. i.e. **all the Database related operations are done by repository.**

`@Component` - Annotate your other components (for example REST resource classes) with component stereotype.

edited Nov 15 at 9:15

answered Mar 25 at 8:00
Harshal Patil
1,798 1 6 26

add a comment

@Repository @Service and **@Controller** are serves as specialization of @Component for more specific use on that basis you can replace @Service to @Component but in this case you loose the specialization.

```
1. **@Repository**   - Automatic exception translation in your persistence layer.
2. **@Service**      - It indicates that the annotated class is providing a business
service to other layers within the application.
```

answered Jul 18 at 11:23
atish shimpi
210 4 14

add a comment

Even if we interchange @Component or @Repository or @service

It will behave the same , but one aspect is that they wont be able to catch some specific exception related to DAO instead of Repository if we use component or @ service

answered Feb 10 at 18:21
Manjush
41 4

add a comment

Not the answer you're looking for? Browse other questions tagged java spring spring-mvc annotations or ask your own question.