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

Ask Question

Can `@Component`, `@Repository` and `@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

edited Jun 15 '17 at 6:31

Raman Sahasi

14.4k 3 29 47

asked Jul 26 '11 at 9:10

Colin McCree

7,857 3 10 3

3 Being a developer with Microsoft background, I recall the semantic definition of services in the old MS SmartClientSoftwareFactory framework (now a long deprecated complex framework for distributed desktop apps). That definition (nicely documented by Rich Newman) defined the services as stateless reusable

application. Not to mention Spring comes with huge stack of its own. – TriCore Jul 5 '17 at 4:45

6 @TriCore Sprting is a framework, define "rules" for you is its job :) – Walfrat Sep 28 '17 at 15:00

30 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

`@Repository`	`stereotype for persistence layer`
`@Service`	`stereotype for service layer`
`@Controller`	`stereotype for presentation layer (spring-mvc)`

edited Jun 21 at 23:54

Mahozad
1,469 2 7 29

answered Aug 1 '11 at 10:20

stivlo
57.6k 29 123 184

5 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 '14 at 10:08

1 what does "@Repository is already supported as a marker for automatic exception translation in your persistence layer." mean? – Jack Mar 20 '15 at 11:15

8 It's referring to the fact that these annotations are good targets for AOP, and while the other annotations do not define a pointcut yet, they might do that in the future. On the other hand @Repository is already a target for a pointcut at present. That pointcut is used for exception translations, i.e. translating technology-specific exceptions to more generic Spring-based ones, to avoid tight coupling. – stivlo Mar 20 '15 at 11:50

2 @stivlo : I have really tried to understand the term 'stereotype', still not understand. Could you please help me to understand this terminology? It helps a lot and thank you very much – Premraj Jul 2 '15 at 13:24

3 stackoverflow.com/questions/14756486/... – stivlo Jul 3 '15 at 12:52

> First point worth highlighting again is that **with respect to scan-auto-detection and dependency injection for BeanDefinition** all these annotations (viz., @Component, @Service, @Repository, @Controller) are the same. **We can use one in place of another and can still get our way around.**

Differences between @Component, @Repository, @Controller and @Service

> **@Component**

This is a general-purpose stereotype annotation indicating that the class is a spring component.

What's special about @Component

`<context:component-scan>` only scans `@Component` and does not look for `@Controller`, `@Service` and `@Repository` in general. They are scanned because they themselves are annotated with `@Component`.

Just take a look at `@Controller`, `@Service` and `@Repository` annotation definitions:

```
@Component
public @interface Service {
    ….
}
```

```
@Component
public @interface Repository {
    ….
}
```

```
    …
}
```

Thus, it's not wrong to say that `@Controller` , `@Service` and `@Repository` are special types of `@Component` annotation. `<context:component-scan>` picks them up and registers their following classes as beans, just as if they were annotated with `@Component` .

They are scanned because they themselves are annotated with `@Component` annotation. If we define our own custom annotation and annotate it with `@Component` , then it will also get scanned with `<context:component-scan>`

> **@Repository**

This is to indicate that the class defines a data repository.

What's special about @Repository?

In addition to pointing out that this is an *Annotation based Configuration*, `@Repository` 's job is to catch Platform specific exceptions and re-throw them as one of Spring's unified unchecked exception. And for this, we're provided with `PersistenceExceptionTranslationPostProcessor` , that we are required to add in our Spring's application context like this:

```
<bean
class="org.springframework.dao.annotation.PersistenceExceptionTransl
```

This bean post processor adds an advisor to any bean that's annotated with `@Repository` so that any platform-specific exceptions are caught and then rethrown as one of Spring's unchecked data access exceptions.

The `@Controller` annotation indicates that a particular class serves the role of a controller. The `@Controller` annotation acts as a stereotype for the annotated class, indicating its role.

What's special about @Controller?

We cannot switch this annotation with any other like `@Service` or `@Repository`, even though they look same. The dispatcher scans the classes annotated with `@Controller` and detects `@RequestMapping` annotations within them. We can only use `@RequestMapping` on `@Controller` annotated classes.

> **@Service**

`@Services` hold business logic and call method in repository layer.

What's special about @Service?

Apart from the fact that it is used to indicate that it's holding the business logic, there's no noticeable speciality that this annotation provides, but who knows, spring may add some additional exceptional in future.

> ***What else?***

Similar to above, in future Spring may choose to add special functionalities for `@Service`, `@Controller` and `@Repository` based on their layering conventions. Hence its always a good idea to respect the convention and use them in line with layers.

edited Oct 14 '17 at 6:10

53 This one is my favourite. There plenty good answers for this question and I envourage you to read them all, but this emphasizes the diferrences between the annotations. – Adrian Cosma Nov 4 '16 at 8:36

5 nice short clean explanation – VedX Jan 27 '17 at 5:50

6 No words to admire this answer. Each and every statement is so clear and easy for understanding. – MAC Feb 3 '17 at 17:40

3 The explanation given here is concise and to the point. This is the only satisfactory answer I have had soo far – Julius Krah Mar 10 '17 at 23:38

6 This is The Answer. – Alessandro May 24 '17 at 12:55

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")
```

edited Aug 1 '11 at 12:20

answered Jul 26 '11 at 9:16

Bozho
471k 106 923 1044

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

16 @Component beans are auto detectable by spring container. You don't need to define bean in configuration file, it will be automatically detected at runtime by Spring. – Akash5288 Dec 16 '14 at 18:07

1 I'm quite fond of the generic @Component... especially in combo with @Scope(proxyMode = ScopedProxyMode.//MODE) – Eddie B Feb 21 '15 at 1:35

@Component is equivalent to

```
<bean>
```

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

That mean Service, The 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.

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

Let's 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.

edited Mar 7 '17 at 13:30

stuart_mad
97 1 14

answered May 23 '14 at 5:15

Oliver
4,515 2 23 50

5 I got a fundamental question - are annotations used by spring mechanism or they are just for the programmer to remember what those pieces of code do? – user107986 Sep 6 '15 at 11:29

20 @user107986 They are mainly for the Programmer to remember layers in the application. However `@Respository` also has automatic exception translation feature. Like when an exception occurs in a `@Repository` there is usually a handler for that exception and there is no need to add try catch blocks in the

can say this advice will apply on all the "@Repository" classes.I was trying to get the sample of this but not able to find.Your help is really appreciated. – Moni Jan 29 '16 at 21:55

Also, while the annotations all currently work the same functionally, it is possible that specific functionality for a given attribute might be added in the future. – Cod3Citrus Apr 10 '17 at 1:05

In Spring `@Component` , `@Service` , `@Controller` , and `@Repository` are Stereotype annotations which are used for:

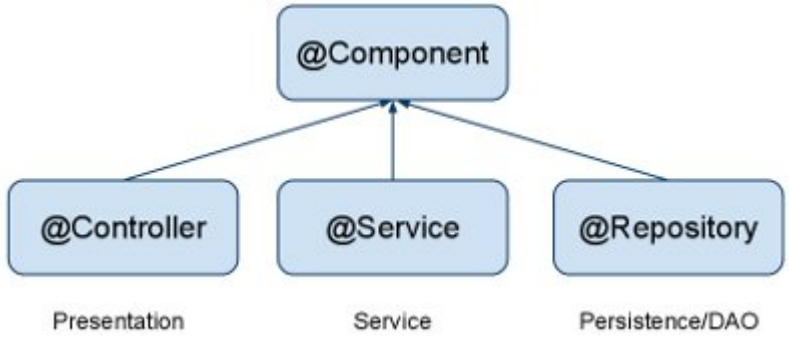
`@Controller:` where your **request mapping from presentation page** done i.e. Presentation layer won't go to any other file it goes directly to `@Controller` class and checks for requested path in `@RequestMapping` annotation which written before method calls if necessary.

`@Service` : All business logic is here i.e. Data related calculations and all.This annotation of business layer in which our user not directly call persistence method so it will call this method using this annotation. **It will request @Repository as per user request**

`@Repository` : This is Persistence layer(Data Access Layer) of application which used to get data from the database. i.e. **all the Database related operations are done by the repository.**

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

> Indicates that an annotated class is a "component". Such classes are considered as candidates for auto-detection when using annotation-based configuration and classpath scanning.



edited Apr 10 at 21:28

Saheb
494 1 5 20

answered Mar 25 '14 at 8:00

Harshal Patil
5,496 8 30 50

17 these answers are all nice and all but im pretty sure what most of us want is some code examples of the features that components like service offers that we can more concretely put in our head rather than just a general description like "business logic" belongs in this object. otherwise, we still assume "oh that's great and everything but i can still apply the same code to component" – dtc Apr 26 '16 at 21:05

Not *all* business logic should go into services! Services, in terms of DDD, should only contain domain logic that affects more than one entity. See answer stackoverflow.com/a/41358034/238134 – deamon Apr 26 '17 at 20:57

@deamon Yes but its depends on developers approach – Harshal Patil Apr 27 '17 at 6:33

2 @HarshalPatil You could of course write an application with all business logic in services, but that would lead to an anemic domain model and it would make it unnecessary hard to enforce constraints and consistence on the entities. –

> 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
> ```

reference :- Spring Documentation - Classpath scanning, managed components and writing configurations using Java

edited Mar 26 at 2:26

Pragati Singh
1,959 14 26

answered May 15 '13 at 12:48

Ajit Singh
1,596 14 22

```
@Service – Indicates a Service component in the business layer.
@Controller – Indicates a controller component in the presentation l
```

You will noticed that all `@Repository` , `@Service` or `@Controller` are annotated with `@Component` . So, can we use just `@Component` for all the components for auto scanning? Yes, you can, and Spring will auto scan all your components with @Component annotated.

It's working fine, but not a good practice, for readability, you should always declare `@Repository` , `@Service` or `@Controller` for a specified layer to make your code more easier to read.

edited Jan 29 '16 at 13:45

Luffy
1,055 1 15 38

answered Dec 16 '14 at 18:10

Akash5288
1,383 14 15

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

407 4 2

Difference Between @Component, @Service and @Repository

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.
`@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.

edited Jan 29 '16 at 13:01

Luffy
1,055 1 15 38

answered Oct 10 '15 at 8:11

nijogeorgep
757 10 14

@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 persi
2. **@Service**      - It indicates that the annotated class is prov
business service to other layers within the application.
```

answered Jul 18 '14 at 11:23

atish shimpi
3,496 1 18 38

---

all these annotations are type of stereo type type of annotation,the difference between these three annotations are

> - If we add the @Component then it tells the role of class is a component class it means it is a class consisting some logic,but it does not tell whether a class containing a specifically business or persistence or controller logic so we don't use directly this @Component annotation
>
> If we add @Service annotation then it tells that a role of class

> - Here @Component is a base annotation for @Service,@Repository and @Controller annotations

for example

```
package com.spring.anno;
@Service
public class TestBean
{
    public void m1()
    {
       //business code
    }
}

package com.spring.anno;
@Repository
public class TestBean
{
    public void update()
    {
       //persistence code
    }
}
```

- whenever we adds the `@Service` or `@Repositroy` or `@Controller` annotation by default `@Component` annotation is going to existence on top of the class

edited Apr 3 '17 at 4:06

Harshal Patil
5,496 8 30 50

answered Apr 25 '15 at 16:19

Anil Amane
419 5 9

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.

edited Mar 26 at 1:43

Pragati Singh
1,959 14 26

answered Feb 24 '15 at 12:49

Vikas Gupta
6,046 4 21 30

Spring provides four different types of auto component scan annotations, they are `@Component` , `@Service` , `@Repository` and `@Controller` . Technically, there is no difference between them, but every auto component scan annotation should be used for a special purpose and within the defined layer.

`@Component` : It is a basic auto component scan annotation, it indicates annotated class is an auto scan component.

`@Controller` : Annotated class indicates that it is a controller component, and mainly used at the presentation layer.

`@Service` : It indicates annotated class is a Service component in the

One should choose a more specialised form of `@Component` while annotating their class as this annotation may contain specific behavior going forward.

edited Jun 16 at 20:23

Saheb

494 1 5 20

answered Jan 3 '16 at 5:10

hardeep thakur

211 2 6

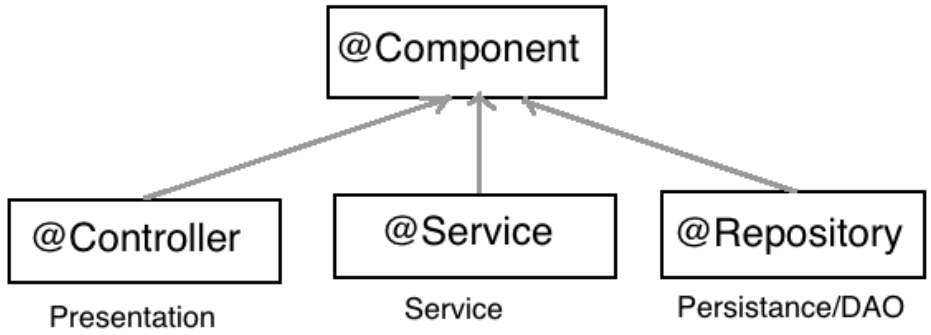
Annotate other components with @Component, for example REST Resource classes.

```
@Component
public class AdressComp{
    .......
    ...//some code here
}
```

@Component is a generic stereotype for any Spring managed component.

@Controller, @Service and @Repository are Specializations of @Component for specific use cases.

@Component in Spring



edited Jan 2 at 7:42

answered Nov 22 '17 at 18:42

Anil Nivargi

326 3 9

Technically @Controller, @Service, @Repository are all same. All of them extends @Components.

From Spring source code:

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

We can directly use @Component for each and every bean, but for better

1) @Controller -> Classes annotated with this, are intended to receive a request from the client side. The first request comes to the Dispatcher Servlet, from where it passes the request to the particular controller using the value of @RequestMapping annotation.

2) @Service -> Classes annotated with this, are intended to manipulate data, that we receive from the client or fetch from the database. All the manipulation with data should be done in this layer.

3) @Repository -> Classes annotated with this, are intended to connect with database. It can also be considered as DAO(Data Access Object) layer. This layer should be restricted to CRUD (create, retrieve, update, delete) operations only. If any manipulation is required, data should be sent be send back to @Service layer.

If we interchange their place(use @Repository in place of @Controller), our application will work fine.

The main purpose of using three different @annotations is to provide better Modularity to the Enterprise application.

edited Jun 21 at 19:47

Saheb

494 1 5 20

answered Jul 27 '17 at 10:20

Yogendra123

316 2 8

1.Major difference between these **stereotypes** is they are used for different classification.

- `@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.

3. 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 it's better to respect the convention and use them in line with layers.
4. Other than the above, with respect to ***scan-auto-detection***, dependency injection for BeanDefinition `@Component`, `@Service`, `@Repository`, `@Controller` are same.
5. As per Spring documentation : The **@Repository** annotation is a marker for any class that fulfills the role or **stereotype** of a **repository** (also known as Data Access Object or ***DAO***). *Among the uses of this marker is the automatic translation of exceptions as described in Section 20.2.2, "Exception translation".* Spring provides further stereotype annotations: **@Component, @Service,** and **@Controller**.

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. It is also possible that @Repository, @Service, and @Controller may carry additional semantics in future releases of the Spring Framework. 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.

edited Nov 28 '16 at 18:59

answered Nov 28 '16 at 18:53

Trilok Singh Devda
513 7 8

We can answer this according to java standard

Referring to `JSR-330`, which is now supported by spring, you can only use `@Named` to define a bean (Somehow `@Named=@Component`). So according to this standard, there seems that there is no use to define stereotypes (like `@Repository`, `@Service`, `@Controller`) to categories beans.

But spring user these different annotations in different for the specific use, for example:

1. Help developers define a better category for the competent. This categorizing may become helpful in some cases. (For example when

layer).

3. If you are using spring MVC, the `@RequestMapping` can only be added to classes which are annotated by `@Controller`.

edited Jun 16 at 20:08

Saheb

494 1 5 20

answered May 4 '16 at 4:23

Alireza Fattahi

19.1k 8 63 87

Regarding your 3rd point. That's not true. I can add @RequestMapping annotation even to methods under service class as well(I mean classes annotated with @Service). – Rahul Gupta Aug 24 at 7:44

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 Bean Definition with `ApplicationContext`.

edited Mar 25 at 21:56

939 8 15

In Spring 4, latest version:

> The @Repository annotation is a marker for any class that fulfills the role or stereotype of a repository (also known as Data Access Object or DAO). Among the uses of this marker is the automatic translation of exceptions as described in Section 20.2.2, “Exception translation”.
>
> Spring provides 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. It is also possible that @Repository, @Service, and @Controller may carry additional semantics in future releases of the Spring Framework. 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.

edited Jun 20 '16 at 12:39

Premraj
25.2k 10 133 110

answered Jun 20 '16 at 9:30

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 '14 at 18:21

Manjush

376 3 9

There is no difference between @Component,@Service,@Controller,@Repository. @Component is the Generic annotation to represent the component of our MVC. But there will be several components as part of our MVC application like service layer components, persistence layer components and presentation layer components. So to differentiate them Spring people have given the other three annotations also.

To represent persistence layer components : @Repository

To represent service layer components : @Service

To represent presentation layer components : @Controller

or else you can use @Component for all of them.

answered Nov 26 '15 at 5:10

tech.yenduri

386 3 7

`@Service` is specialized annotation. it do not bring any new feature as of now but it clarifies the intent of the bean

@Controller is specialized annotation which makes the bean MVC aware and allows the use of further annotation like `@RequestMapping` and all such

Here are more details

edited Jul 7 '17 at 17:26

Mehraj Malik
3,148 1 15 38

answered Dec 21 '16 at 9:12

Amol Dixit
202 2 9

@Component: you annotate a class @Component, it tells hibernate that it is a Bean.

@Repository: you annotate a class @Repository, it tells hibernate it is a DAO class and treat it as DAO class. Means it makes the unchecked exceptions (thrown from DAO methods) eligible for translation into Spring DataAccessException.

@Service: This tells hibernate it is a Service class where you will have @Transactional etc Service layer annotations so hibernate treats it as a Service component.

Plus @Service is advance of @Component. Assume the bean class name is CustomerService, since you did not choose XML bean configuration way so you annotated the bean with @Component to indicate it as a

component with name ‘customerService’. But if you use @Service annotation for the bean class you can provide a specific bean name by

```
@Service("AAA")
public class CustomerService{
```

and you can get the bean object by

```
CustomerService cust = (CustomerService)context.getBean("AAA");
```

answered Nov 9 '17 at 13:32

Arun Raaj
665 1 11 13

A `@Service` to quote spring documentation,

> Indicates that an annotated class is a "Service", **originally defined by Domain-Driven Design (Evans, 2003) as "an operation offered as an interface that stands alone in the model, with no encapsulated state."** May also indicate that a class is a "Business Service Facade" (in the Core J2EE patterns sense), or something similar. This annotation is a general-purpose stereotype and individual teams may narrow their semantics and use as appropriate.

If you look at domain driven design by eric evans,

> A SERVICE is an operation offered as an interface that stands alone in the model, without encapsulating state, as ENTITIES and VALUE OBJECTS do. SERVICES are a common pattern in technical frameworks, but they can also apply in the domain layer. The name

> definition of an object. A SERVICE should still have a defined responsibility, and that responsibility and the interface fulfilling it should be defined as part of the domain model. Operation names should come from the UBIQUITOUS LANGUAGE or be introduced into it. Parameters and results should be domain objects. SERVICES should be used judiciously and not allowed to strip the ENTITIES and VALUE OBJECTS of all their behavior. But when an operation is actually an important domain concept, a SERVICE forms a natural part of a MODEL-DRIVEN DESIGN. Declared in the model as a SERVICE, rather than as a phony object that doesn't actually represent anything, the standalone operation will not mislead anyone.

and a `Repository` as per Eric Evans,

> A REPOSITORY represents all objects of a certain type as a conceptual set (usually emulated). It acts like a collection, except with more elaborate querying capability. Objects of the appropriate type are added and removed, and the machinery behind the REPOSITORY inserts them or deletes them from the database. This definition gathers a cohesive set of responsibilities for providing access to the roots of AGGREGATES from early life cycle through the end.

answered Dec 28 '16 at 8:18

Bharath

679 1 9 26

`@Component` This annotation is used on classes to indicate a Spring component.It marks the Java class as a bean or component so that the component-scanning mechanism of Spring can add it into the application context.

@Repository, there is a handler for that exception and there is no need to add a try-catch block.

`@Service` It marks a Java class that performs some service, such as executing business logic, performing calculations, and calling external APIs. This annotation is a specialized form of the `@Component` annotation intended to be used in the service layer.

`@Controller` This annotation is used to indicate the class is a Spring controller.

answered May 10 at 17:15

codereal

171 1 9

Explanation of stereotypes :

- `@Service` - Annotate all your service classes with @Service. This layer knows the unit of work. All your business logic will be in Service classes. Generally methods of service layer are covered under transaction. You can make multiple DAO calls from service method, if one transaction fails all transactions should rollback.
- `@Repository` - Annotate all your DAO classes with @Repository. All your database access logic should be in DAO classes.
- `@Component` - Annotate your other components (for example REST resource classes) with component stereotype.
- `@Autowired` - Let Spring auto-wire other beans into your classes using @Autowired annotation.

`@Component` is a generic stereotype for any Spring-managed component. `@Repository` `@Service` and `@Controller` are specializations of

answered Mar 21 at 7:03

Jeevan Patil
5,013 3 25 47

`@Component` – generic and can be used across application. `@Service` – annotate classes at service layer level. `@Repository` – annotate classes at persistence layer, which will act as database repository.

edited Jun 20 at 8:45

Gimhani
587 4 19

answered May 16 at 19:20

asok
101 5

In spring framework provides some special type of annotations,called stereotype annotations. These are following:-

```
@RestController- Declare at controller level.
@Controller – Declare at controller level.
@Component – Declare at Bean/entity level.
@Repository – Declare at DAO level.
@Service – Declare at BO level.
```

above declared annotations are special because when we add `<context:component-scan>` into xxx-servlet.xml file ,spring will automatically create the object of those classes which are annotated with above annotation during context creation/loading phase.

answered Jul 5 at 19:11

@Component, @ Repository, @ Service, @Controller:

@Component is a generic stereotype for the components managed by Spring @Repository, @Service, and @Controller are @Component specializations for more specific uses:

- @Repository for persistence
- @Service for services and transactions
- @Controller for MVC controllers

Why use @Repository, @Service, @Controller over @Component? We can mark our component classes with @Component, but if instead we use the alternative that adapts to the expected functionality. Our classes are better suited to the functionality expected in each particular case.

A class annotated with "@Repository" has a better translation and readable error handling with org.springframework.dao.DataAccessException. Ideal for implementing components that access data (DataAccessObject or DAO).

An annotated class with "@Controller" plays a controller role in a Spring Web MVC application

An annotated class with "@Service" plays a role in business logic services, example Facade pattern for DAO Manager (Facade) and transaction handling

edited Jun 21 at 0:30

answered Jun 21 at 0:10

UHDante

```
@controller
@Repository
@service
@RestController
```

These are all StereoType annotations. If we placed @controller on top of class. It will not become controller class based on the different layers(components) we can annotate with this annotations but compiler will not do anything about this just for developer understanding purpose we can choose based on the components which annotations we have to write

edited Aug 29 at 19:12

TylerH
14.9k 10 49 67

answered Aug 29 at 18:46

siddartha kamble
5 1

protected by ben75 Jan 12 '16 at 20:10

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?