



# @Order in Spring

Last modified: February 21, 2019

by [baeldung](#)

Spring ▾

Spring DI

I just announced the new *Learn Spring* course, focused on the fundamentals of Spring 5 and Spring Boot 2:

[>> CHECK OUT THE COURSE](#)

## 1. Overview

In this tutorial, we're going to learn about Spring's `@Order` annotation. **The `@Order` annotation defines the sorting order of an annotated component or bean.**

It has an optional value argument which determines the order of the component; the default value is `Ordered.LOWEST_PRECEDENCE`. This marks that the component has the lowest priority among all other ordered components.

Similarly, the value `Ordered.HIGHEST_PRECEDENCE` can be used for overriding the highest priority among components.

## 2. When to Use `@Order`

Before Spring 4.0, the `@Order` annotation was used only for the AspectJ execution order. It means the highest order advice will run first.

Since Spring 4.0, it supports the ordering of injected components to a collection. As a result, Spring will inject the auto-wired beans of the same type based on their order value.

**Download**  
The E-book

Building a REST API with Spring 5?

Email Address

**Download**

**Download**  
The E-book

Building a REST API with Spring 5?

Email Address

**Download**

### UTM Machine

lyxyantech.com

OEM&ODM Manufacturer

Capacity from 10N to 600KN  
for choose, 0.5 grade  
accuracy, accept customize

Let's explore it with a quick example.

### 3. How to Use *@Order*

First of all, let's set up our project with the relevant interface and classes.

#### 3.1. Interface Creation

Let's create the *Rating* interface that determines the rating of a product:

```
1 public interface Rating {  
2     int getRating();  
3 }
```

#### 3.2. Components Creation

Finally, let's create three components that define the ratings of some products:

```
1 @Component  
2 @Order(1)  
3 public class Excellent implements Rating {  
4  
5     @Override  
6     public int getRating() {  
7         return 1;  
8     }  
9 }  
10  
11 @Component  
12 @Order(2)  
13 public class Good implements Rating {  
14  
15     @Override  
16     public int getRating() {  
17         return 2;  
18     }  
19 }  
20  
21 @Component  
22 @Order(Ordered.LOWEST_PRECEDENCE)  
23 public class Average implements Rating {  
24  
25     @Override  
26     public int getRating() {  
27         return 3;  
28 }
```

OPEN

ezoic

report this ad

#### UTM Machine

lyxyantech.com

OEM&ODM Manufacturer

Capacity from 10N to 600KN  
for choose, 0.5 grade  
accuracy, accept customize

OPEN

ezoic

report this ad

Explore 60+ cloud  
services for free.

Get Started For Free »

Explore 60+ cloud  
services for free.

Get Started For Free »

Note that the *Average* class has the lowest priority because of its overridden value.

## 4. Testing Our Example

Up until now, we've created all the required components and the interface to test the *@Order* annotation. Now, let's test it to confirm that it works as expected:

```
1 public class RatingRetrieverUnitTest {  
2  
3     @Autowired  
4     private List<Rating> ratings;  
5  
6     @Test  
7     public void givenOrder_whenInjected_thenByOrderValue() {  
8         assertThat(ratings.get(0).getRating(), is(equalTo(1)));  
9         assertThat(ratings.get(1).getRating(), is(equalTo(2)));  
10        assertThat(ratings.get(2).getRating(), is(equalTo(3)));  
11    }  
12 }
```

## 5. Conclusion

We've learned about the *@Order* annotation in this quick article. We can find the application of *@Order* in various use cases – where the ordering of the auto-wired components matter. One example is the Spring's request filters.

Due to its influence on injection precedence, it may seem like it might influence the singleton startup order also. But in contrast, the dependency relationships and *@DependsOn* declarations determine the singleton startup order.

All examples mentioned in this tutorial can be found [over on Github](#).

### Universal Test System

OEM&ODM Manufacturer

Capacity from 10N to 600KN for choose, 0.5 grade accuracy, accept customize

lyxyantech.com

OPEN

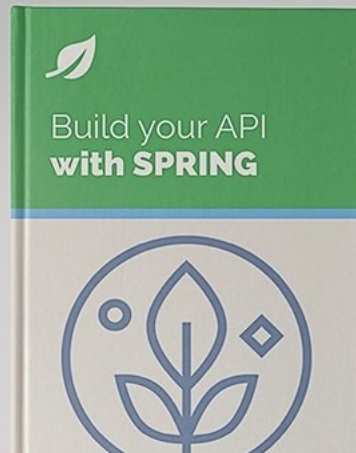
I just announced the new *Learn Spring* course, focused on the fundamentals of Spring 5 and Spring Boot 2:

[>> CHECK OUT THE COURSE](#)



Build Your REST API  
with Spring

[GET THE COURSE](#)



Learning to build your API  
**with Spring?**

Enter your email address

[>> Get the eBook](#)



Build Your REST API  
with Spring

[GET THE COURSE](#)

Comments are closed on this article!

#### Tensile Test Machine

lyxyantech.com

OEM&ODM Manufacturer

Capacity from 10N to 600KN for  
choose, 0.5 grade accuracy,  
accept customize



report this ad



Build Your REST API  
with Spring

GET THE COURSE



## CATEGORIES

[SPRING](#)  
[REST](#)  
[JAVA](#)  
[SECURITY](#)  
[PERSISTENCE](#)  
[JACKSON](#)  
[HTTP CLIENT-SIDE](#)  
[KOTLIN](#)

## SERIES

[JAVA "BACK TO BASICS" TUTORIAL](#)  
[JACKSON JSON TUTORIAL](#)  
[HTTPCLIENT 4 TUTORIAL](#)  
[REST WITH SPRING TUTORIAL](#)  
[SPRING PERSISTENCE TUTORIAL](#)  
[SECURITY WITH SPRING](#)

## ABOUT

[ABOUT BAELDUNG](#)  
[THE COURSES](#)  
[CONSULTING WORK](#)  
[META BAELDUNG](#)  
[THE FULL ARCHIVE](#)  
[WRITE FOR BAELDUNG](#)  
[EDITORS](#)  
[OUR PARTNERS](#)  
[ADVERTISE ON BAELDUNG](#)

[TERMS OF SERVICE](#) [PRIVACY POLICY](#) [COMPANY INFO](#) [CONTACT](#)

We use cookies to improve your experience with the site. To find out more, you can read the full [Privacy and Cookie Policy](#)

Ok