Making Java easy to learn

Java Technology and Beyond

Spring Boot Errors And AOP Annotations With Examples

Spring Boot Docker Java Spring *by devs5003* February 28, 2021 1

In this article we will discuss on 'Spring Boot Errors and AOP Annotations with Examples'. Needless to say, these annotations play a crucial role in creating Spring Boot Applications. If you want to learn all annotations which are generally used in a Spring Boot Project, kindly visit our article 'Spring Boot Annotations with Examples'. Let's discuss about 'Spring Boot Errors and AOP Annotations with Examples' here only.

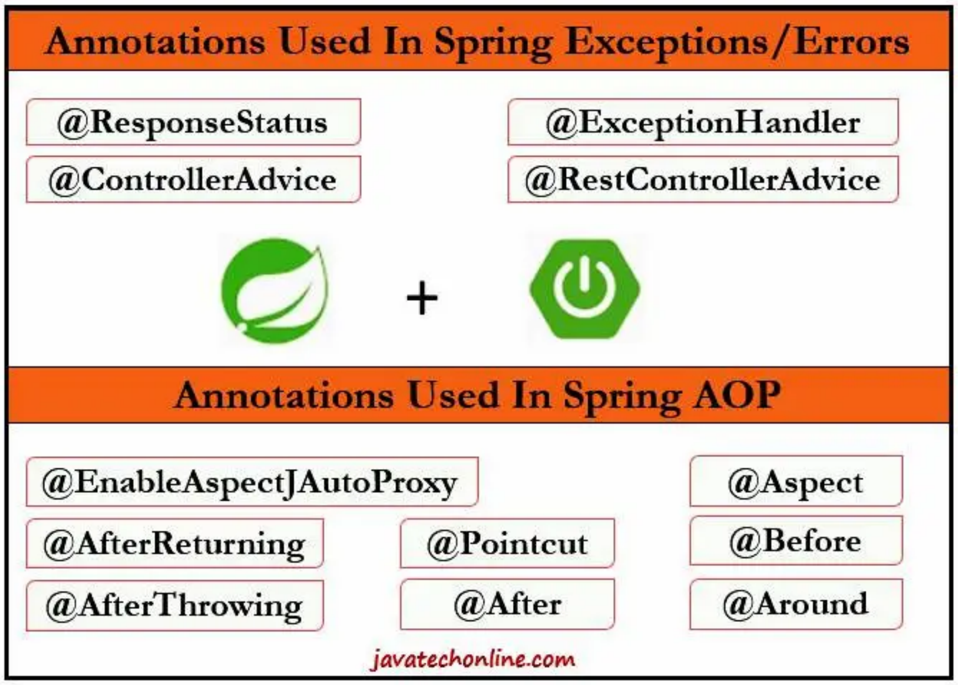

Table of Contents

TOP

Annotations on Spring Errors & Exceptions

Spring Boot proves that Exception handling is a crosscutting concern by offering some annotations such as @Exception Handler, @ContollerAdvice, @ResponseStatus. In order to learn about Spring Boot Exception Handling in detail, kindly visit our article 'How To Handle Exceptions & Errors In Spring Boot?'.

@ResponseStatus

In Spring Boot's Default Exception handling mechanism, When any runtime exception occurs, our web controllers provide a general error response in the response payload. Our error responses always display us the HTTP status 500 (indicates Internal Server Error) instead of a more descriptive status code. Instead of a general error response, we can get a well-formed descriptive error response with the help of *@ResponseStatus*. This annotation allows us to modify the HTTP status of our response. It can be applied at the following places.

1) On the exception class itself
2) Along with the *@ExceptionHandler* annotation on methods
3) Along with the *@ControllerAdvice* annotation on classes

For example, We can annotate our Custom Exception class with *@ResponseStatus* and pass in the desired HTTP response status in its code attribute as shown below. In order to know about other important status codes, kindly visit the list of Status Codes.

```
@ResponseStatus(code = HttpStatus.NOT_FOUND)
public class NoSuchUserFoundException extends RuntimeException {
 ...
}
```

Here if we call our controller with an invalid user, we will receive httpStatus.NOT_FOUND(indica TOP r status code 404) which is a better response.

@ExceptionHandler

If you want to modify the whole structure of the response payload in order to make it more user friendly and descriptive, the *@ExceptionHandler* annotation offers us a lot of flexibility to do so. In fact, we just need to create a method either in the controller class itself or in a *@ControllerAdvice* annotated class and apply *@ExceptionHandler.* Now it is clear that we can apply this annotation on top of a class as well as a method. For example, below code demonstrates the use if *@ExceptionHandler*. We annotated the method with *@ExceptionHandler* and *@ResponseStatus* to define the exception, we want to handle and the status code we want to return. Moreover, the *@ExceptionHandler* can accept a list of exceptions that we want to handle in the defined method.

```
@RestController
@RequestMapping("/user")
public class UserController {
    @Autowired
    private UserService userService;

    @GetMapping("/{id}")
    public Response getUser(@PathVariable String id) {
       return userService.getUser(id);
    }

    @ExceptionHandler(NoSuchUserFoundException.class)
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public ResponseEntity<String> handleNoSuchUserFoundException( NoSuchUserFoundException exception) {
         return ResponseEntity .status(HttpStatus.NOT_FOUND)
        .body(exception.getMessage());
    }
}
```

@ContollerAdvice and @RestControllerAdvice

Exception Handler classes annotated with *@ControllerAdvice* or *@RestControllerAdvice* offer us to apply exception handlers to more than one or all controllers in our application. A controller advice offers us to intercept and alter the return values of controller methods to handle the exceptions. The term 'Advice' comes from Aspect-Oriented Programming (AOP) that offers us to apply cross-cutting concern concepts around existing methods. Needless to say, if we use *@RestControllerAdvice* in place of *@ControllerAdvice*, we don't need to apply @ResponseBody at method level. For example, belc demonstrates the use of *@ControllerAdvice*:

TOP

```
//@RestControllerAdvice
@ControllerAdvice
public class UserExceptionHandler {

    @ExceptionHandler(NoSuchUserFoundException.class)
    @ResponseBody
    public ResponseEntity<Object> handleUserNotFoundException(NoSuchUserFoundException unfe) {
        return new ResponseEntity<>("User not found", HttpStatus.NOT_FOUND);
    }
}
```

Moreover, This handler will handle exceptions thrown by all the controllers in the application and not just let's say UserController.

In order to know more about the Errors & Exception Handling in Spring Boot, you may visit our separate article on Spring Boot Errors & Exceptions.

Spring AOP Annotations

In Spring AOP, we use annotations provided by AspectJ library. In order to get the features of AOP, we need to add below dependency in pom.xml file.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>  spring-boot-starter-aop  </artifactId>
</dependency>
```

@EnableAspectJAutoProxy

In Aspect Oriented Programming we use AspectJ annotations to declare Advices. To enable AspectJ annotations you must first apply *@EnableAspectJAutoProxy* annotation to your configuration class. Then it will enable support for components which are marked with @Aspect annotations. For example, below code demonstrates the *@EnableAspectJAutoProxy* annotation.

```
@Configuration
@EnableAspectJAutoProxy
public class MyAopConfiguration { }
```

TOP

@Aspect

We apply *@Aspect* on top of the class to represent that this class is an Aspect. For example, below InvoiceAspect class represents an Aspect.

```
@Aspect
public class InvoiceAspect { }
```

@Pointcut

@Pointcut is applied on top of a method in the Aspect class. It has an expression in the bracket to select the business methods accordingly. Remember that it will only select the business methods which need advice, but never talks about which advice. For example, below p1() method represents a pointcut.

```
@Pointcut("execution(public void com.dev.spring.aop.service.InvoiceBusinessService.saveInvoice())")
public void p1() { }
```

Further below Annotations are the types of Advices. An advice must be annotated with any one type of the advice.
♥ **Note :** Every Advice Type Annotation must describe related pointcut information as a parameter (in the bracket).

@Before

Method annotated with *@Before* represents the Before Advice and it will execute before the execution of business method. For example, below method represents Before Advice with Pointcut as p1().

```
@Before("p1()")
public void beginTransaction() {
        System.out.println("Transaction begins !");
}
```

@After

Method annotated with *@After* represents the After Advice and it will execute after the executic
business method. For example, below method represents After Advice with Pointcut as p1().

TOP

```
@After("p1()")
public void completeTransaction() {
        System.out.println("Transaction completes !");
}
```

@AfterReturning

Method annotated with @*AfterReturning* represents the After Returning Advice and it will execute after the execution of business method but only if method executes successfully. For example, below method represents After Returning Advice with Pointcut as p1().

```
@AfterReturning("p1()")
public void commitTransaction() {
        System.out.println("Transaction committed !");
}
```

@AfterThrowing

Method annotated with @*AfterThrowing* represents the After Throwing Advice and it will execute after the execution of business method, but only if the method fails to execute successfully. For example, below method represents After Throwing Advice with Pointcut as p1().

```
@AfterThrowing("p1()")
public void rollbackTransaction() {
        System.out.println("Transaction rolled back !");
}
```

@Around

Method annotated with @*Around* represents the Around Advice. It executes in two parts. Some part executes before the execution of business method whereas other part executes after the execution of business method. For example, below method represents Around Advice with Pointcut as p4().

```
@Around("p4()")
public void testAroundAdvice(ProceedingJoinPoint pj) throws Throwable {
      System.out.println("Executing Before part of business method");
```

TOP

```
          pj.proceed();          // this code will call business method
          System.out.println("Executing After part of business method");
     }
```

♦ In order to learn more on AOP, kindly visit our article 'How To Implement AOP In Spring Boot Application?'

How to implement AOP in Spring Boot Application?
January 10, 2021
In "Spring Boot"

Spring Transaction Annotations With Examples
August 11, 2022
In "java"

What is Spring Boot ?
June 1, 2021
In "java"

Tagged @After @AfterReturning @AfterThrowing @Around @Aspect @Before @ContollerAdvice @controlleradvice example @controlleradvice in spring @controlleradvice in spring boot @EnableAspectJAutoProxy @ExceptionHandler @ResponseStatus @RestControllerAdvice advice annotation spring aop example aop example in spring boot aop implementation in spring boot example aop in spring boot aop spring aop spring boot aop tutorial spring boot controller advice in spring boot controller advice spring boot controlleradvice spring boot exceptionhandler spring boot Pointcut restcontrolleradvice example spring aop spring aop @annotation spring aop annotations spring aop example spring aop example with spring boot spring aop exception spring aop implementation spring aop in spring boot spring aop tutorial spring boot annotations interview questions spring boot aop spring boot aop annotation example spring boot aop annotations spring boot aop example spring boot aop tutorial spring boot exceprion annotations spring exception annotations springboot aop example what is spring aop

‹ Previous article
How to develop a Reactive CRUD REST API with Spring WebFlux?

Next article ›
Spring Boot MVC REST Annotations With Examples

One thought on "Spring Boot Errors and AOP Annotations With Examples"

Pingback: Spring Boot Annotations With Examples | Making Java easy to learn

Leave a Reply

TOP