Implementing Generic Exception Handling for all Resources

As we discussed earlier that we should define a standard exception structure which is followed across all the RESTful services. In this section, we will discuss the implementation **of generic exception handling for all resources**.

Let's see how to customize the exception message.

**Step 1:** Create a new package with name **com.javatpoint.server.main.exception**.

**Step 2:** Create a class with the name **ExceptionResponse** in the above package.

PlayNext

Mute

Current Time 2:04

/

Duration 18:10

Loaded: 17.24%

Fullscreen

Backward Skip 10sPlay VideoForward Skip 10s

**Step 3:** At the basic level, there are three crucial things for the exception structure: **timestamp, message,** and **detail**. Define these three fields.

**Step 4:** Generate Constructors using Fields.

**Step 5:** Generate Getters, Setters are not needed.

Once the structure is defined, we can define Java implementation for it. Whenever an exception occurs, we would return a response in the specific format. The structure is the most important part and must have to be language independent.

One of the important class in Spring Framework is **ResponseEntityExceptionHandler** class. It is an abstract and base class for the exceptions that provide centralized exception handling across all the different exception handler methods. We will extend this class to handle and provide customized exception handling functionality. This exception handling functionality is applied to all controller as HellowWorldController, UserResource (Controller).

**Step 6:** Create a new class with name **CustomizedResponseEntityExceptionHandler** in the package **com.javatpoint.server.main.exception** and extends **ResponseEntityExceptionHandler** class.

Advertisement

**Step 7:** Add @ControllerAdvice and @RestController annotations.

**Step 8:** Expand Maven Dependencies in the package explorer -> Expand spring-webmvc-5.1.9.RELEASE.jar -> Expand org.springframework.web.servlet.mvc.method.annotation -> Open **ResponseEntityExceptionHandler.class**.

Implementing Generic Exception Handling for all Resources

**Step 9:** Copy the **ResponseEntityMethod<Object>** method from ResponseEntityExceptionHandler.class and paste in **CustomizedResponseEntityExceptionHandler.java** file.

**Step 10:** Override the **ResponseEntityMethod** method. Rename the method name as handleAllExceptions().

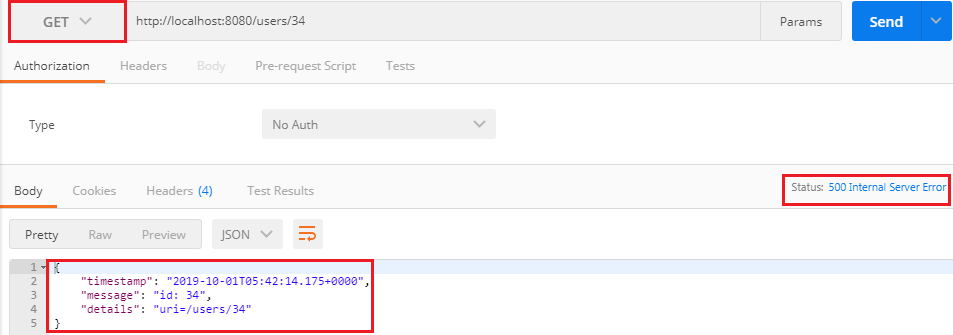
**Step 11:** Create the exception response structure.

**Step 12:** Create a **ResponseEntity** object and pass the exception response and Http status as arguments.

**CustomizedResponseEntityExceptionHandler.java**

1. **package** com.javatpoint.server.main;
2. **import** java.util.Date;
3. **import** org.springframework.http.HttpStatus;
4. **import** org.springframework.http.ResponseEntity;
5. **import** org.springframework.web.bind.annotation.ControllerAdvice;
6. **import** org.springframework.web.bind.annotation.ExceptionHandler;
7. **import** org.springframework.web.bind.annotation.RestController;
8. **import** org.springframework.web.context.request.WebRequest;
9. **import** org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;
10. **import** com.javatpoint.server.main.exception.ExceptionResponse;
11. //defining exception handling for all the exceptions
12. @ControllerAdvice
13. @RestController
14. **public** **class** CustomizedResponseEntityExceptionHandler **extends** ResponseEntityExceptionHandler
15. {
16. @ExceptionHandler(Exception.**class**)
17. //override method of ResponseEntityExceptionHandler class
18. **public** **final** ResponseEntity<Object> handleAllExceptions(Exception ex, WebRequest request)
19. {
20. //creating exception response structure
21. ExceptionResponse exceptionResponse= **new** ExceptionResponse(**new** Date(), ex.getMessage(), request.getDescription(**false**));
22. //returning exception structure and specific status
23. **return** **new** ResponseEntity(exceptionResponse, HttpStatus.INTERNAL\_SERVER\_ERROR);
24. }
25. }

**Step 13:** Open rest client **Postman** and send a **Get** request. We get the **Status: 500 Internal Server Error** and exception structure which we have defined.

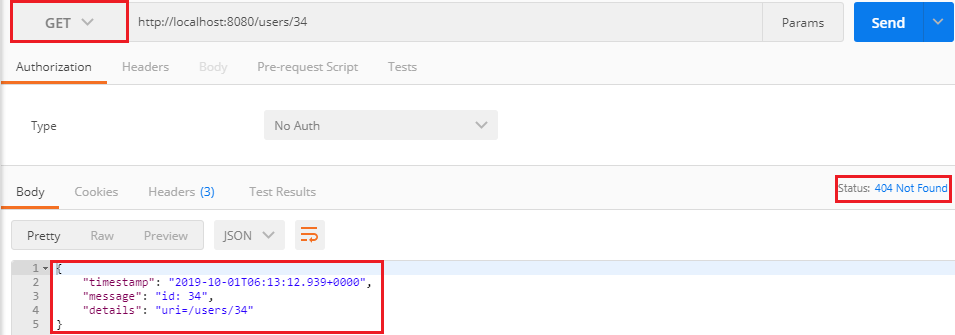


If we want to customize the status **Internal Server Error to Not Found**, we are required to change a few things in **CustomizedResponseEntityExceptionHandler.java** file.

**CustomizedResponseEntityExceptionHandler.java**

1. **package** com.javatpoint.server.main;
2. **import** java.util.Date;
3. **import** org.springframework.http.HttpStatus;
4. **import** org.springframework.http.ResponseEntity;
5. **import** org.springframework.web.bind.annotation.ControllerAdvice;
6. **import** org.springframework.web.bind.annotation.ExceptionHandler;
7. **import** org.springframework.web.bind.annotation.RestController;
8. **import** org.springframework.web.context.request.WebRequest;
9. **import** org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;
10. **import** com.javatpoint.server.main.exception.ExceptionResponse;
11. **import** com.javatpoint.server.main.user.UserNotFoundException;
12. //defining exception handling for all the exceptions
13. @ControllerAdvice
14. @RestController
15. **public** **class** CustomizedResponseEntityExceptionHandler **extends** ResponseEntityExceptionHandler
16. {
17. @ExceptionHandler(Exception.**class**)
18. //override method of ResponseEntityExceptionHandler class
19. **public** **final** ResponseEntity<Object> handleAllExceptions(Exception ex, WebRequest request)
20. {
21. //creating exception response structure
22. ExceptionResponse exceptionResponse= **new** ExceptionResponse(**new** Date(), ex.getMessage(), request.getDescription(**false**));
23. //returning exception structure and Internal Server status
24. **return** **new** ResponseEntity(exceptionResponse, HttpStatus.INTERNAL\_SERVER\_ERROR);
25. }
26. @ExceptionHandler(UserNotFoundException.**class**)
27. //override method of ResponseEntityExceptionHandler class
28. **public** **final** ResponseEntity<Object> handleUserNotFoundExceptions(UserNotFoundException ex, WebRequest request)
29. {
30. //creating exception response structure
31. ExceptionResponse exceptionResponse= **new** ExceptionResponse(**new** Date(), ex.getMessage(), request.getDescription(**false**));
32. //returning exception structure and Not Found status
33. **return** **new** ResponseEntity(exceptionResponse, HttpStatus.NOT\_FOUND);
34. }
35. }

Again move to **Postman** and send a **Get** request. We get the **Status: 404 Not Found** with the defined exception structure.



[Click here to download Implementing Generic Exception Handing For All Users project](https://static.javatpoint.com/tutorial/restful-web-services/download/Implementing%20Generic%20Exception%20Handing%20For%20All%20Users.zip)