A **Global Exception Handler** in Spring allows you to manage exceptions across the whole application in a centralized way. Using Spring's @ControllerAdvice and @ExceptionHandler, you can handle exceptions globally, instead of writing custom exception handling logic for each controller.

**✅ Steps to Create a Global Exception Handler in Spring Boot:**

**1. Create a Custom Exception Class (optional)**

First, you might want to define custom exceptions to handle specific error scenarios.

java

CopyEdit

public class ResourceNotFoundException extends RuntimeException {

public ResourceNotFoundException(String message) {

super(message);

}

}

**2. Create the Global Exception Handler with @ControllerAdvice**

This class will catch exceptions thrown in any controller and return a proper response.

java

CopyEdit

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.ResponseStatus;

@ControllerAdvice

public class GlobalExceptionHandler {

// Handle ResourceNotFoundException globally

@ExceptionHandler(ResourceNotFoundException.class)

@ResponseStatus(HttpStatus.NOT\_FOUND) // Send 404 response status

public ResponseEntity<String> handleResourceNotFound(ResourceNotFoundException ex) {

return new ResponseEntity<>(ex.getMessage(), HttpStatus.NOT\_FOUND);

}

// Handle general exceptions

@ExceptionHandler(Exception.class)

@ResponseStatus(HttpStatus.INTERNAL\_SERVER\_ERROR) // Send 500 response status

public ResponseEntity<String> handleGeneralException(Exception ex) {

return new ResponseEntity<>("An error occurred: " + ex.getMessage(), HttpStatus.INTERNAL\_SERVER\_ERROR);

}

// Handle ValidationException or any other custom exception

@ExceptionHandler(ValidationException.class)

@ResponseStatus(HttpStatus.BAD\_REQUEST)

public ResponseEntity<String> handleValidationException(ValidationException ex) {

return new ResponseEntity<>(ex.getMessage(), HttpStatus.BAD\_REQUEST);

}

}

**3. What Happens Here:**

* **@ControllerAdvice**: It indicates that this class will handle exceptions globally for all controllers in your application.
* **@ExceptionHandler**: Used to handle specific exceptions. For example, ResourceNotFoundException will be handled with one method, and general exceptions will be caught with another method.
* **@ResponseStatus**: This annotation allows you to set a specific HTTP status (like 404 for not found or 500 for server error) for the exception response.

**✅ Example: Custom Exception and Global Handling**

**Custom Exception Class:**

java

CopyEdit

public class ResourceNotFoundException extends RuntimeException {

public ResourceNotFoundException(String message) {

super(message);

}

}

**Controller Example:**

java

CopyEdit

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class UserController {

@GetMapping("/user/{id}")

public String getUser(@PathVariable("id") String userId) {

if (userId.equals("1")) {

return "User found!";

} else {

throw new ResourceNotFoundException("User not found with ID: " + userId);

}

}

}

**Global Exception Handler:**

java

CopyEdit

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(ResourceNotFoundException.class)

@ResponseStatus(HttpStatus.NOT\_FOUND)

public ResponseEntity<String> handleResourceNotFound(ResourceNotFoundException ex) {

return new ResponseEntity<>(ex.getMessage(), HttpStatus.NOT\_FOUND);

}

@ExceptionHandler(Exception.class)

@ResponseStatus(HttpStatus.INTERNAL\_SERVER\_ERROR)

public ResponseEntity<String> handleGeneralException(Exception ex) {

return new ResponseEntity<>("An error occurred: " + ex.getMessage(), HttpStatus.INTERNAL\_SERVER\_ERROR);

}

}

**✅ What Happens:**

* When you hit the endpoint /user/{id}, if the userId is not "1", the ResourceNotFoundException will be thrown.
* The **global exception handler** will catch the exception and return a **404 Not Found** response with the exception message.

For example, calling:

bash

CopyEdit

GET http://localhost:8080/user/2

Will return:

json

CopyEdit

{

"message": "User not found with ID: 2"

}

**✅ Handling Validation Errors Globally**

If you want to handle **validation exceptions** (e.g., using @Valid or @NotNull), you can use a separate handler for validation errors.

**Example:**

java

CopyEdit

import org.springframework.validation.BindException;

import org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(BindException.class)

public ResponseEntity<String> handleValidationException(BindException ex) {

return new ResponseEntity<>("Validation failed: " + ex.getMessage(), HttpStatus.BAD\_REQUEST);

}

}

**✅ Return a Custom Error Response**

Sometimes you might want to return a structured response instead of just a message. You can create a custom error response DTO.

**Custom Error Response DTO:**

java

CopyEdit

public class ErrorResponse {

private String message;

private String details;

// Constructor, Getters, Setters

}

**Updated Global Exception Handler:**

java

CopyEdit

@ExceptionHandler(ResourceNotFoundException.class)

@ResponseStatus(HttpStatus.NOT\_FOUND)

public ResponseEntity<ErrorResponse> handleResourceNotFound(ResourceNotFoundException ex) {

ErrorResponse errorResponse = new ErrorResponse("Resource Not Found", ex.getMessage());

return new ResponseEntity<>(errorResponse, HttpStatus.NOT\_FOUND);

}

Now, when an exception is thrown, the response will look like:

json

CopyEdit

{

"message": "Resource Not Found",

"details": "User not found with ID: 2"

}

**✅ Conclusion**

* **@ControllerAdvice** allows you to handle exceptions globally.
* **@ExceptionHandler** defines the methods to handle specific exceptions.
* **@ResponseStatus** allows you to set HTTP status codes on the response.