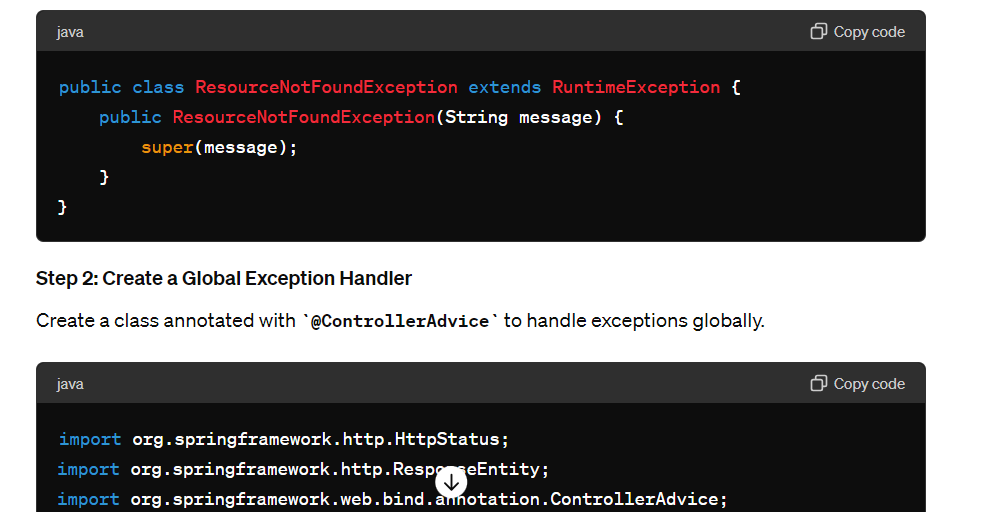
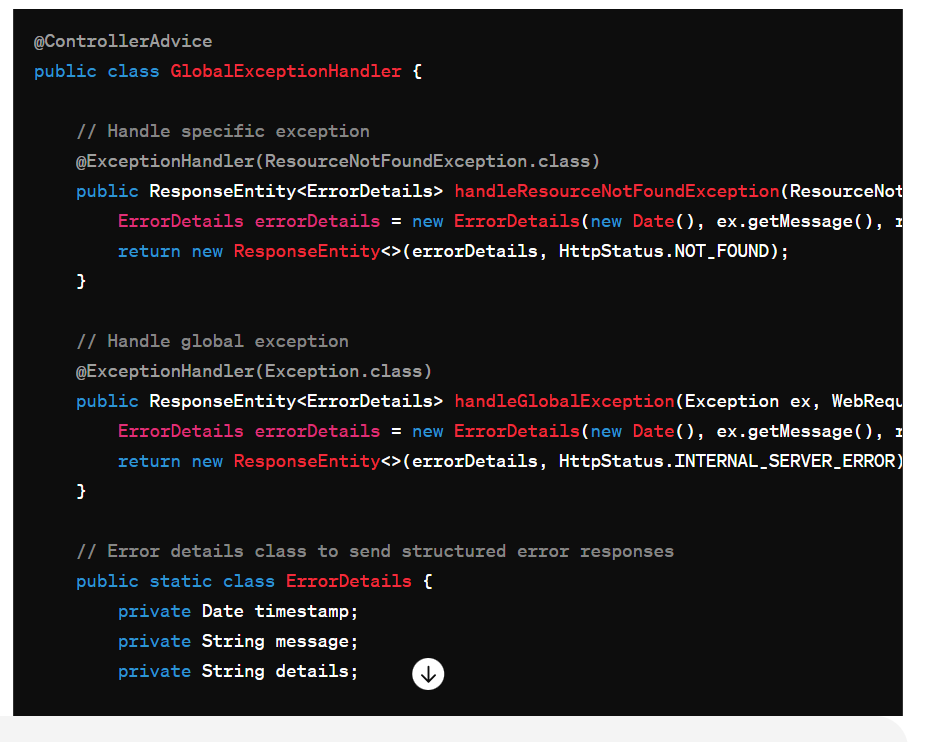
Global exception handling in Spring Boot allows you to manage exceptions in a centralized manner, ensuring consistency and reducing boilerplate code across your application. This can be achieved using **@ControllerAdvice** and **@ExceptionHandler** annotations.

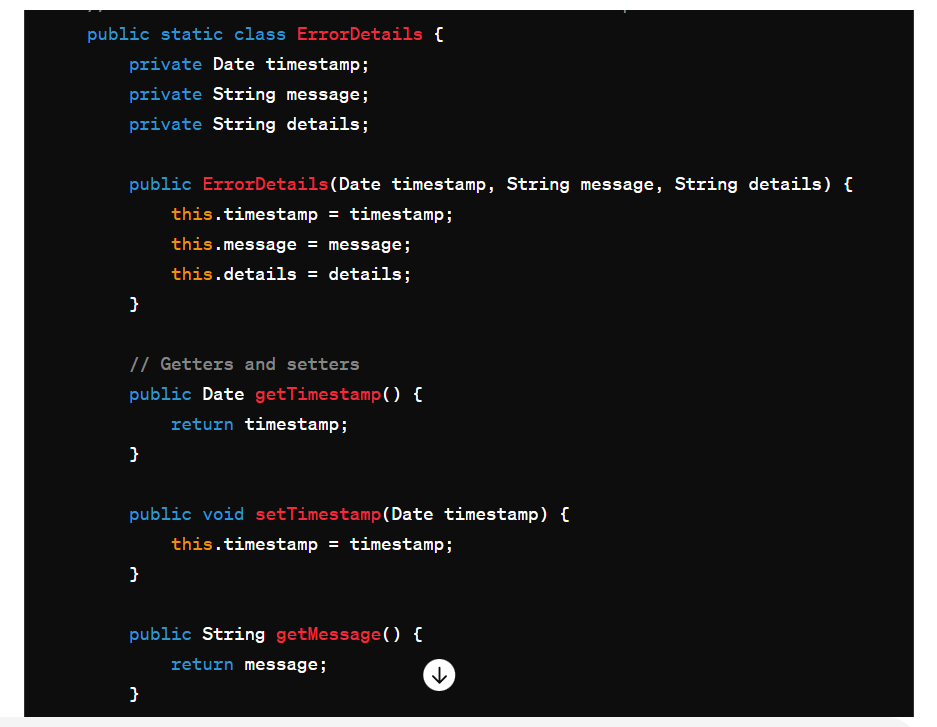
**Steps to Implement Global Exception Handling**

**Step 1: Create Custom Exception Classes**

First, create custom exception classes to represent specific error conditions.

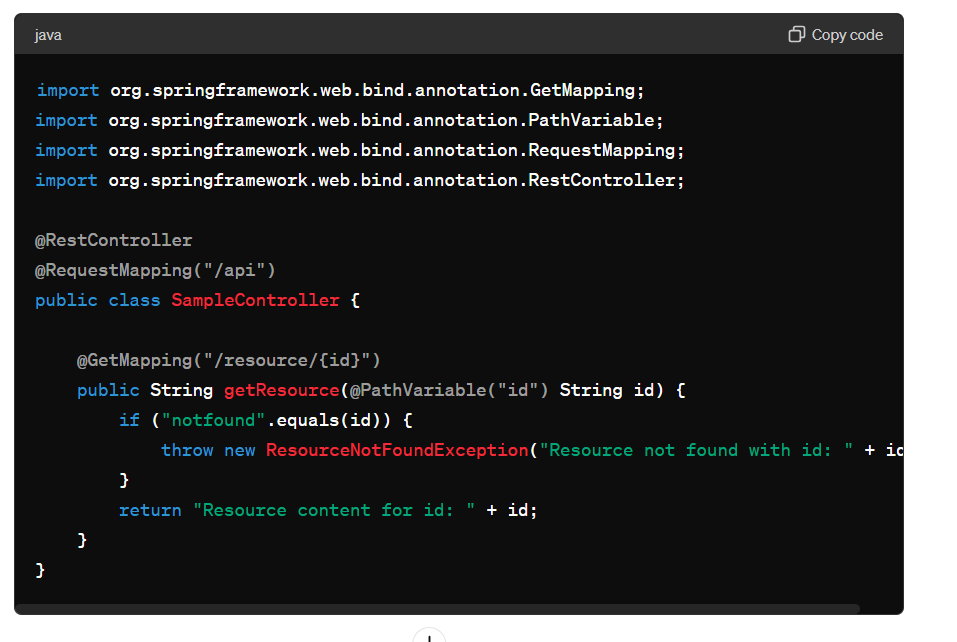






#### Step 3: Use the Custom Exception in a Controller

Create a REST controller and use the custom exception.



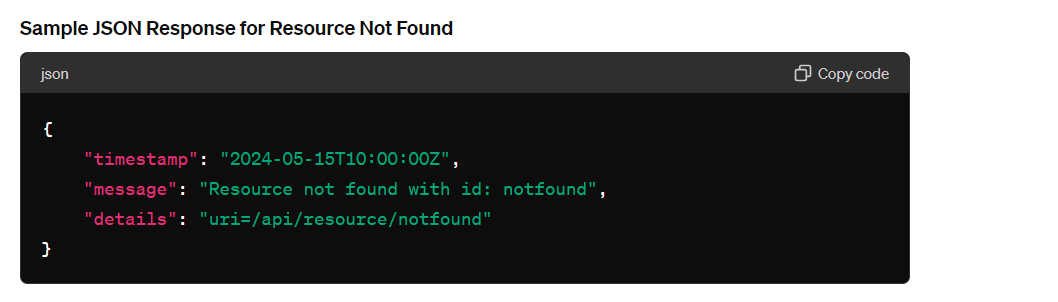
**Summary of Steps**

1. **Create Custom Exception Classes**: Define specific exceptions for your application.
2. **Create a Global Exception Handler**: Use **@ControllerAdvice** and **@ExceptionHandler** to define how different exceptions should be handled globally.
3. **Throw Custom Exceptions**: Use these custom exceptions in your controllers and services as needed.

**Example Usage**

When you hit the endpoint **/api/resource/notfound**, the **ResourceNotFoundException** is thrown. The **GlobalExceptionHandler** catches this exception and returns a structured JSON response with a 404 status code.

**Sample JSON Response for Resource Not Found**



#### Sample JSON Response for Global Exception

If an unexpected exception occurs, the **handleGlobalException** method will catch it and return a 500 status code with the corresponding error details.

By following these steps, you can implement a robust global exception handling mechanism in your Spring Boot application, ensuring that all exceptions are handled consistently and gracefully.