**Exercise 6: Online Bookstore - Exception Handling in REST Controllers**

**Business Scenario:**

In this exercise, the objective is to implement a global exception handling mechanism for the RESTful services of an online bookstore application. The goal is to ensure that any errors or exceptions that occur during the execution of the REST API endpoints are handled in a centralized and consistent manner, providing appropriate HTTP status codes and informative error messages to the client.

**Global Exception Handling:**

To achieve a global exception handling mechanism, we utilize the @ControllerAdvice annotation in Spring Boot. This annotation allows us to create a centralized exception handler that can handle exceptions thrown by any controller in the application. The @ExceptionHandler methods within this global exception handler are used to capture specific exceptions and return appropriate HTTP status codes and error messages.

**Instructions:**

1. **Global Exception Handler:**
   * **Create a GlobalExceptionHandler Class:**
     + The GlobalExceptionHandler class is annotated with @ControllerAdvice, indicating that it should handle exceptions globally across all controllers.
     + Inside this class, various methods are defined to handle different exceptions that might be thrown during the execution of RESTful services.
   * **Define Methods to Handle Various Exceptions:**
     + Each method within the GlobalExceptionHandler class is annotated with @ExceptionHandler, specifying the type of exception it handles.
     + These methods return a ResponseEntity object containing an ErrorResponse object, which includes details about the exception (such as status code and message).
     + Different HTTP status codes are returned depending on the nature of the exception, ensuring that the client receives meaningful feedback about what went wrong.

**Exception Handling Mechanism:**

1. **Custom Exception Handling:**
   * **handleResourceNotFoundException:** Handles ResourceNotFoundException and returns a 404 Not Found status.
   * **handleBookIsbnAlreadyExitsException:** Handles BookIsbnAlreadyExitsException and returns a 409 Conflict status.
2. **Validation Errors:**
   * **handleMethodArgumentNotValidException:** Handles validation errors on DTOs and returns a 400 Bad Request status with detailed error messages.
   * **handleConstraintViolationException:** Handles validation errors on entity fields and returns a 400 Bad Request status.
3. **Type Mismatch and Method Not Allowed:**
   * **handleMethodArgumentTypeMismatchException:** Handles method argument type mismatches and returns a 405 Method Not Allowed status.
   * **handleRequestMethodNotSupportedException:** Handles unsupported HTTP methods and returns a 405 Method Not Allowed status.
4. **Common Exceptions:**
   * **handleIllegalArgumentException:** Handles illegal arguments and returns a 400 Bad Request status.
   * **handleHttpMessageNotReadableException:** Handles unreadable HTTP messages and returns a 400 Bad Request status.
   * **handleNoHandlerFoundException:** Handles cases where no handler is found for a request and returns a 404 Not Found status.
   * **handleUnsupportedMediaTypeException:** Handles unsupported media types and returns a 415 Unsupported Media Type status.
   * **handleCommonException:** Handles any other exceptions and returns a 500 Internal Server Error status.

**ErrorResponse Class:**

* The ErrorResponse class is a simple data structure used to encapsulate the details of an error, including the HTTP status code and a descriptive message.
* It is used as the body of the ResponseEntity returned by the exception handler methods.

**Summary:**

This exercise demonstrates the implementation of a global exception handling mechanism in a Spring Boot application. By using @ControllerAdvice and @ExceptionHandler, we centralize the handling of exceptions and provide consistent and meaningful responses to the client.