Ministry of SMEs and Startups × KOSME Korea SMEs and Startups Agency × LIKELION

Exception handling, HttpStatus, Custom Status

Nguyễn Anh Tuấn

KTECH COLLEGE

Nội dung bài giảng

1 Exception handling

2 Custom Exception

3 HttpStatus

4 Custom Status

Exception handling

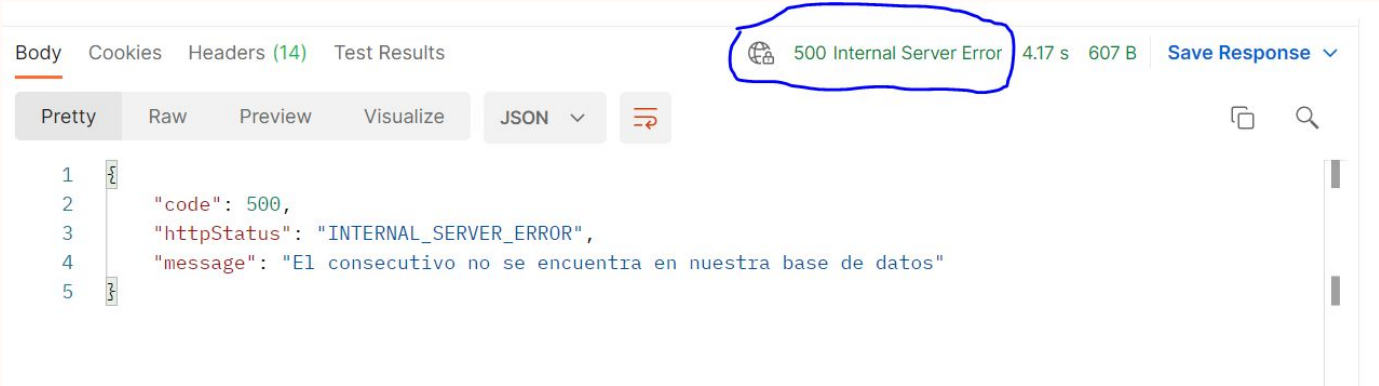
Exception handling

- Trong ứng dụng **Java cơ bản**, khi gặp phải 1 exception mà không được xử lý (không có khối try-catch hoặc throws), ứng dụng sẽ kết thúc ngay lập tức. Điều này xảy ra vì ứng dụng chỉ chạy trên 1 **main thread** duy nhất và không có cách nào để tiếp tục, dẫn đến việc JVM chấm dứt chương trình.

- Trong **Spring Boot**, mỗi yêu cầu HTTP được xử lý bởi một luồng (**thread**) riêng biệt. Khi 1 exception xảy ra, nó chỉ ảnh hưởng đến luồng đó, trong khi các luồng khác vẫn tiếp tục xử lý các yêu cầu khác một cách bình thường.
- **Spring Boot** chạy trong một web server nhúng như **Tomcat**. **Tomcat** sử dụng một **thread pool** để xử lý các yêu cầu HTTP đến. Khi ứng dụng Spring Boot khởi động cùng với **Tomcat**, **Tomcat** sẽ tạo ra 1 **thread pool** và quản lý các luồng. Mỗi khi có một yêu cầu HTTP đến, **Tomcat** sẽ lấy 1 luồng từ pool để xử lý yêu cầu đó.

Exception handling

- Try-catch trong spring boot dùng để xử lý exception cụ thể tại chỗ và phản hồi lại người dùng 1 cách chi tiết.
- Mặc dù try-catch hữu ích, nhưng khi cần xử lý exception trên toàn bộ ứng dụng, Spring Boot khuyến khích sử dụng handler exception như **@ExceptionHandler** và **@RestControllerAdvice**, giúp tránh lặp lại code và làm cho việc quản lý exception nhất quán hơn.



Exception handling

- **@ControllerAdvice:**
  - Được sử dụng để xử lý các exception và thực hiện logic chung cho các controller trong 1 ứng dụng Spring MVC.
  - Áp dụng cho toàn bộ ứng dụng hoặc giới hạn cho 1 số controller cụ thể.
  - Thường được sử dụng với các controller trả về giao diện người dùng, thường là dưới dạng HTML.
- **@RestControllerAdvice:**
  - Tương tự như **@ControllerAdvice** nhưng dành cho các REST controller (các controller trả về dữ liệu dưới dạng JSON hoặc XML).
  - Áp dụng cho toàn bộ ứng dụng hoặc giới hạn cho 1 số controller cụ thể.
  - Các phương thức bên trong **@RestControllerAdvice** trả về dữ liệu trực tiếp (thường là JSON) thông qua **@ResponseBody**.
- **@ExceptionHandler:**
  - Được sử dụng để chỉ định 1 phương thức xử lý exception cụ thể.
  - ExceptionHandler của 1 exception cụ thể được gọi khi 1 ngoại lệ tương ứng xảy ra.

Exception handling

```
@RestController
@RequestMapping("/api/products")
public class ProductController {

    @GetMapping("/{id}")
    public ResponseEntity<Product> getProductById(@PathVariable Long id) {
        Product product = productService.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Product not
            found with id " + id));
        return ResponseEntity.ok(product);
    }
}
```

Exception handling

```
@RestControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(ResourceNotFoundException.class)
    public ResponseEntity<String> handleException(ResourceNotFoundException ex) {
        e.printStackTrace();
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(ex.getMessage());
    }

    // Xử lý ngoại lệ chung chung, nên có hàm này nếu đã sử dụng ExceptionHandler
    @ExceptionHandler(Exception.class)
    public ResponseEntity<String> handleGlobalException(Exception ex) {
        e.printStackTrace();
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Lỗi!");
    }
}
```

Exception handling

- **Lợi ích khi sử dụng exception handling:**
  - **Global exception handler:** Cho phép định nghĩa cách xử lý lỗi ở một nơi tập trung, giúp hạn chế lặp lại code (vi phạm quy tắc **DON'T REPEAT YOURSELF**). Thay vì phải xử lý lỗi trong từng phương thức thì có thể xử lý tất cả exception ở một lớp hoặc phương thức duy nhất.
  - **Ẩn đi lỗi thực sự:** Thực tế là không bao giờ được trả về chi tiết lỗi (error details) cho client.
  - **Tùy chỉnh message:** Có thể tùy chỉnh nội dung phản hồi lỗi trả về cho người dùng, bao gồm mã trạng thái HTTP, thông điệp lỗi, và dữ liệu liên quan.
  - **Khả năng scaling:** Có thể mở rộng các phương thức xử lý lỗi để thực hiện các hành động bổ sung như ghi log lỗi, gửi thông báo hoặc thực hiện các hành động khác dựa trên loại lỗi.
  - **Dễ maintain:** Khi lỗi được xử lý tập trung, việc kiểm tra và bảo trì mã dễ dàng hơn, có thể dễ dàng theo dõi và điều chỉnh cách xử lý lỗi mà không cần phải thay đổi từng phương thức xử lý yêu cầu.

Custom Exception

Custom Exception

- Trong trường hợp muốn tuỳ chỉnh thông báo exception theo ý muốn và dễ dàng phân biệt với system exception (các exception của hệ thống):

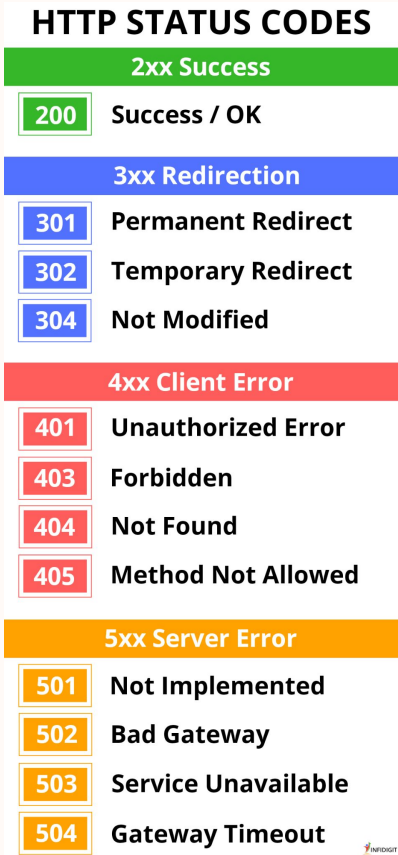
```
@Getter
@AllArgsConstructor
public class AppException extends RuntimeException {
        private int code;
        private String message;
}

public class UserService {
        public User getUser(String username) {
                User user = userRepository.findByUsername(username);
                // Nếu muốn xử lý ngoại lệ thì vẫn phải dùng @ExceptionHandler
                if (user == null) { throw new AppException(404, "User not found"); }
        }
}
```

HttpStatus

HttpStatus

- **HttpStatus** là 1 phần quan trọng của giao thức HTTP (HyperText Transfer Protocol) và được sử dụng để xác định trạng thái của phản hồi từ máy chủ đến khách hàng.
- Mỗi mã trạng thái HTTP bao gồm một mã số và một mô tả văn bản, cho biết kết quả của yêu cầu mà máy khách gửi đến máy chủ.
- **Tham khảo:** https://en.wikipedia.org/wiki/List_of_HTTP_status_codes



HttpStatus

- **1xx - Thông tin (Informational):** Các trạng thái trong nhóm **1xx** thường không được sử dụng phổ biến, nhưng có ích trong việc cung cấp thông tin về tiến trình của yêu cầu.
  - **100 Continue:** Yêu cầu đã được nhận và khách hàng có thể tiếp tục gửi dữ liệu.
  - **101 Switching Protocols:** Server đồng ý chuyển đổi giao thức theo yêu cầu của khách hàng.

- **2xx - Thành công (Successful):** Các trạng thái này cho biết rằng yêu cầu của khách hàng đã được server xử lý thành công.
  - **200 OK:** Yêu cầu thành công và phản hồi chứa thông tin mong đợi.
  - **204 No Content:** Yêu cầu thành công nhưng không có nội dung để trả về.

- **3xx - Chuyển hướng (Redirection):** Các mã trạng thái này cho biết rằng khách hàng cần thực hiện thêm hành động để hoàn tất yêu cầu.
  - **301 Moved Permanently:** Tài nguyên đã được di chuyển đến một địa chỉ URL mới.
  - **302 Found:** Tài nguyên tạm thời di chuyển đến một địa chỉ URL khác.

HttpStatus

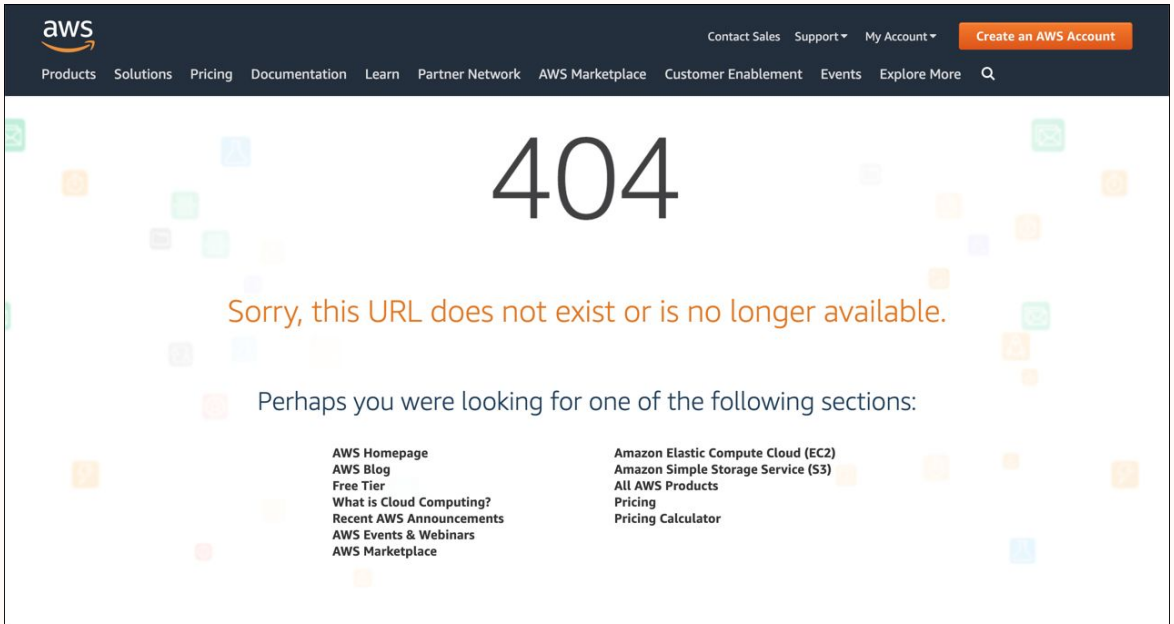
- **4xx - Lỗi của khách hàng (Client Error):** Các trạng thái này cho biết có lỗi trong yêu cầu của client mà server không thể hoặc không muốn xử lý:
  - **400 Bad Request:** Yêu cầu không hợp lệ hoặc không thể xử lý.
  - **401 Unauthorized:** Yêu cầu cần xác thực người dùng.
  - **403 Forbidden:** Server hiểu yêu cầu nhưng từ chối thực hiện nó.
  - **404 Not Found:** Tài nguyên yêu cầu không tìm thấy trên server.
  - **409 Conflict:** Yêu cầu không thể hoàn tất do xung đột.

- **5xx - Lỗi của máy chủ (Server Error):** Các trạng thái này cho biết có lỗi trên server, không thể hoàn tất yêu cầu của khách hàng:
  - **500 Internal Server Error:** Server gặp lỗi không xác định hoặc lỗi nội bộ.
  - **502 Bad Gateway:** Server khi hoạt động như một gateway hoặc proxy, nhận được phản hồi không hợp lệ từ server gốc.
  - **503 Service Unavailable:** Server hiện không khả dụng, thường là do bảo trì hoặc quá tải.
  - **504 Gateway Timeout:** Server khi hoạt động như một gateway hoặc proxy, không nhận được phản hồi kịp thời từ server gốc.

Custom Status

Custom Status

- Mặc dù các custom status không phải là phần của tiêu chuẩn HTTP và có thể không được tất cả các máy khách và máy chủ web nhận diện đúng cách, tuy nhiên vẫn có thể định nghĩa các status code tùy chỉnh và sử dụng chúng trong response của controller.



Custom Status

```
public enum CustomHttpStatus {
        CUSTOM_STATUS_404(404, "Sorry, this URL does not exist or is no longer available."),
        CUSTOM_STATUS_200(200, "Custom Status 200"),
        CUSTOM_STATUS_201(201, "Custom Status 201");
        private final int code;
        private final String message;

        CustomHttpStatus(int code, String message) {
                this.code = code;
                this.message = message;
        }

        public int getCode() { return code; }

        public String getMessage() { return message; }
}
```

Custom Status

```
@ExceptionHandler(ResourceNotFoundException.class)
public ResponseEntity<String> handleException(ResourceNotFoundException ex) {
        // Truy cập status code tùy chỉnh như hằng số
        CustomHttpStatus status404 = CustomHttpStatus.CUSTOM_STATUS_404;
        e.printStackTrace();
        return ResponseEntity.status(status404.getCode()).body(status404.getMessage());
}
```

ROAD TO KOREA

Nếu có bất kỳ thắc mắc nào, hãy đặt câu hỏi qua

mail@mail.com hoặc Zalo 0xxx xxx xxx