Leets 백엔드 4주차 세션

API 명세 & 테스트

2025.10.16

오늘의 목표

- swagger, postman 사용하기
- 예외 처리하기
- 테스트 짜보기 (단위 테스트 / 통합 테스트)
- API 개발한거 피드백

Swagger / PostMan

API 문서화

우리가 만든 애플리케이션의 API를
프론트엔드 쪽에서 연결하자는 연락이 온다면 어떻게 해야할까?

API 문서화

우리가 만든 API의 작동 방식을 다른 개발자에게 정확히 보여주는 도구

클라이언트가 REST API로 백엔드에 요청을 전송하기 위해
알아야 하는 요청 정보 혹은 URL/URI 등을 문서로 정리하는 것!

이 주소(URI)로 이런 데이터를 보내면, 서버는 이런 형식의 응답을 줄거야~

API 문서화

- 만약, API 문서를 워드나 노션 등의 문서를 이용해 수기로 작성한다면 어떨까?

➡️ 이미 작성한 API 문서에 기능이 추가/수정되면, 잊어버리고 수정하지 않는 일이 발생할 수 있음 ㅠㅠ
➡️ 클라이언트에게 제공된 API 정보와 수기로 작성된 API 문서의 정보가 불일치하는 경우도 생길 수 있음

문서를 자동화할 순 없을까..😕

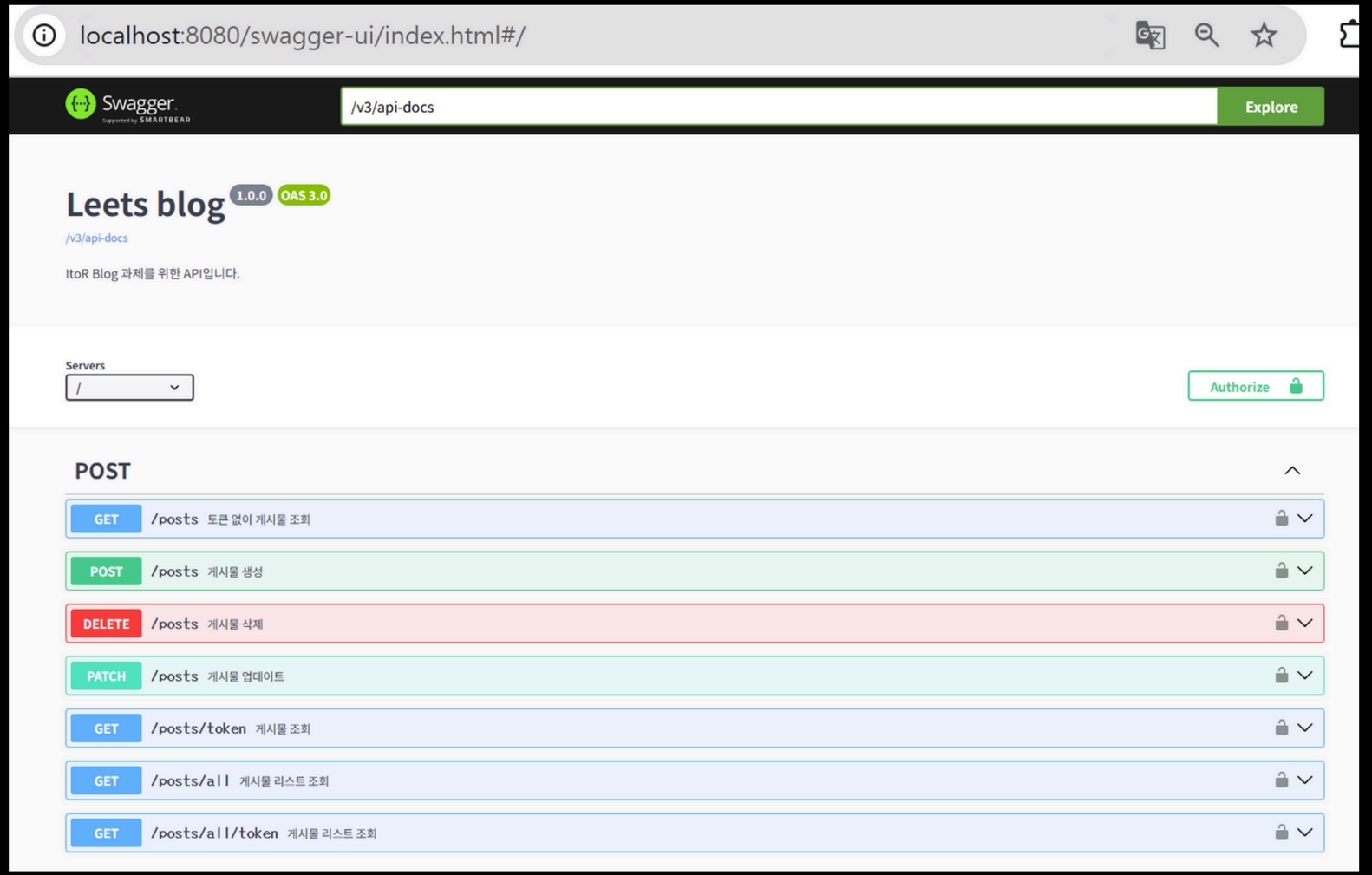
Swagger

- 애너테이션 기반의 API 문서화 방식
- 코드에 문서화를 위한 애너테이션들이 포함됨!
- 그 기반이 되는 명세 규격 = OpenAPI Specification → API의 설계 계약서 역할

장점??

- 프론트 개발자 -> 서버 코드가 완성되지 않아도 swagger 문서를 보고 개발을 미리 시작할 수 있음!!!! (문서화의 장점)
- 명세를 읽기 쉽고 상호작용 가능한 웹 인터페이스(Swagger UI)로 변환
- 코드 몇줄로 자동 생성
- 'TRY IT OUT' 기능을 사용하여 서버에 실제 요청을 보낼 수 있음 ><

Swagger 사용법


localhost:8080/swagger-ui/index.html#/
Swagger
Supported by SMARTBEAR
/v3/api-docs
Explore
Leets blog
1.0.0
OAS 3.0
/v3/api-docs
ItoR Blog 과제를 위한 API입니다.
Servers
/
Authorize
POST
GET /posts 토큰 없이 게시물 조회
POST /posts 게시물 생성
DELETE /posts 게시물 삭제
PATCH /posts 게시물 업데이트
GET /posts/token 게시물 조회
GET /posts/all 게시물 리스트 조회
GET /posts/all/token 게시물 리스트 조회

Swagger 사용법

1. Gradle에 springdoc-openapi-ui 의존성 추가
2. Swagger Config 설정
3. 어노테이션 설정
    a. @Tag(name = "장바구니 관리", description = "아이템 추가/조회")
    b. @Operation(summary = "아이템 추가", description = "사용자 장바구니에 새 아이템을 넣습니다.")
    c. @ApiResponse(responseCode = "201", description = "추가 성공")
4. http://localhost:8080/swagger-ui.html 로 접속 → API명세서 자동 생성

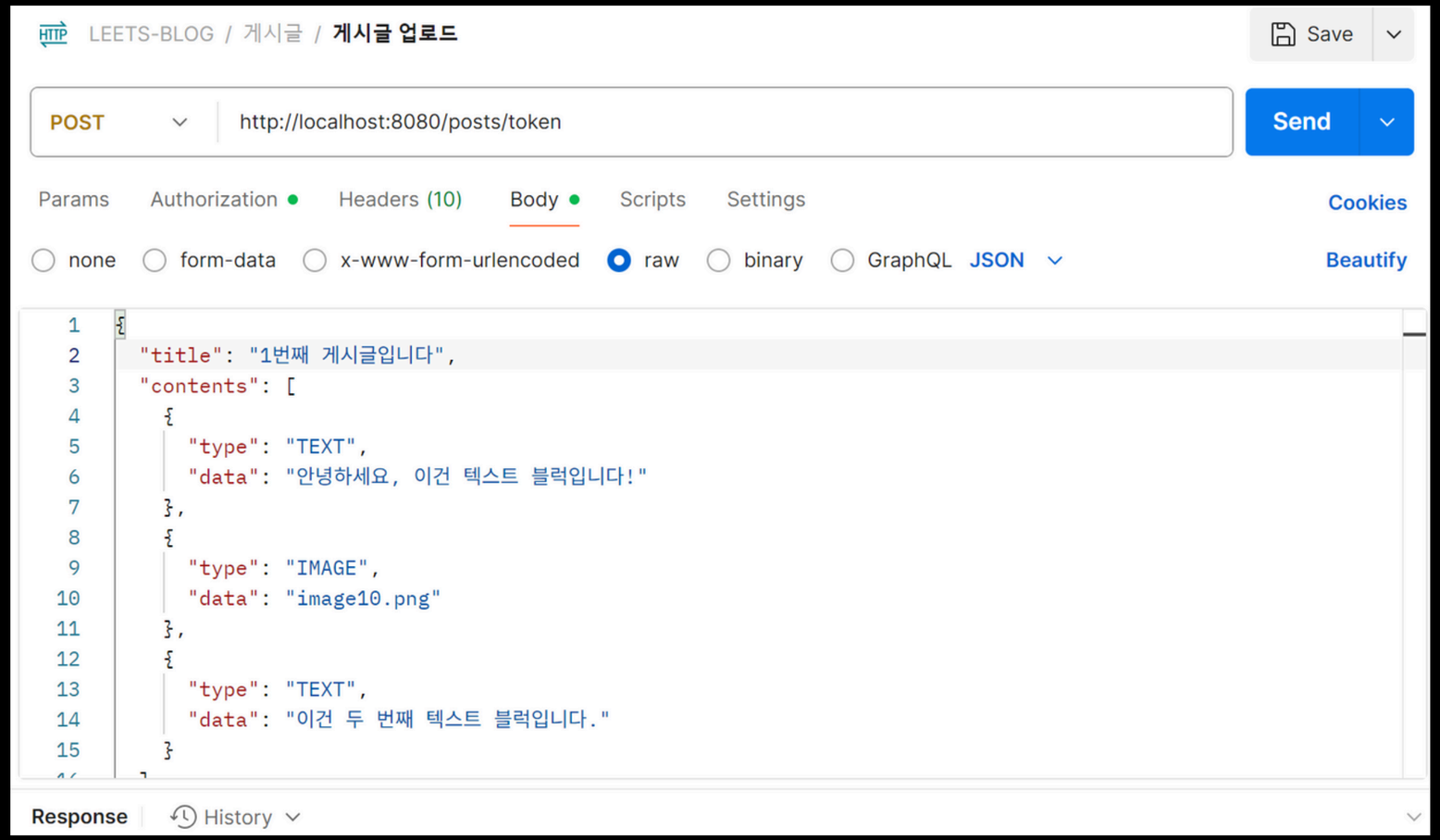
Postman

- 우리가 만든 API가 잘 작동하는지 확인하고 관리하는 도구
- 개인적으로 사용

장점??

- 다양한 시나리오를 쉽게 test할 수 있음 ➡️ 활용성이 더 높고 간편함

Postman 사용법


LEETS-BLOG / 게시글 / 게시글 업로드
Save
POST
http://localhost:8080/posts/token
Send
Params
Authorization
Headers (10)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"title": "1번째 게시글입니다",
"contents": [
{
"type": "TEXT",
"data": "안녕하세요, 이건 텍스트 블럭입니다!"
},
{
"type": "IMAGE",
"data": "image10.png"
},
{
"type": "TEXT",
"data": "이건 두 번째 텍스트 블럭입니다."
}
Response
History

Postman

단점??

- API 코드와 직접적으로 연결되지 않음 → 최대 단점 ㅠㅠ
- 수동으로 postman컬랙션과 문서를 업데이트해야함
- 예제값 채울 수 없음
- 프론트한테 공유하기 어려움 → 자유도가 높고, postman사용에 익숙해야함

→ swagger, postman 둘다 요청이 성공했는지를 확인하는 용도에 가까움 → 정확성을 보장하진 못함 (실제로는 원하는 결과가 나오지 않을수도..🥲)

항목	Swagger	Postman
문서화	자동 문서화, 엔드포인트/파라미터/응답 구조 확인	컬렉션 기반 문서화 가능, 수동 업데이트 필요
실제 테스트	“Try it out”으로 간단한 요청 테스트 가능	다양한 환경/시나리오 테스트, 스크립트/자동화 가능
팀 협업	URL 공유 가능, 문서 중심 협업	컬렉션 공유, 테스트/스크립트 중심 협업
시나리오 테스트	제한적 (주로 단일 요청 확인)	환경변수, 반복 테스트, 인증 등 다양한 시나리오 가능

단위 테스트 / 통합 테스트

테스트 왜 함??

걍 swagger상 잘 작동되면 끝아님???

테스트 왜 함?

테스트는 고객에게 우리는 절대 실수를 하지 않는다는 확신을 주는 안전 보험!!

Swagger는 API가 통신에 성공했음만 보여줄 뿐,
내부 로직의 정확성을 검증하지 못함 ㅠㅠ

우리가 swagger상으로 테스트할 땐 문제 없었는데
사용자가 다양한 시나리오로 요청을 보냈을때 500가 뜬다면??!!

테스트 왜 함?

단위/통합 테스트는 유저정보가 실제로 저장되었는지 등
핵심 기능의 성공 여부를 세밀하게 알아낼 수 있음

여러 요청이 순서대로 연결될 때 발생하는 복잡한 시나리오 오류를 잡아냄

새로운 코드 수정이 기존 기능을 망가뜨리는 것을 자동으로 막아줌!!

단위 테스트 (Unit Test)

- 시스템을 이루는 가장 작은 조각 (함수, 메서드)이 외부 환경(데이터베이스, 인터넷)과 상관없이 정확히 작동하는지 검사하는 과정

- ex) JUnit (Java 테스트 도구)

- 엔티티 계층 (Entity Layer) 테스트
- 서비스 계층 (Service Layer) 테스트
  - createPost() 서비스 계층 메서드에 요청 데이터가 들어왔을 때, 가짜 저장소(Mock)의 savePost() 메서드가 정확히 한 번 호출되었는지 검사!

통합 테스트 (Integration Test)

- 여러 개의 작은 부분이 함께 연결되어 작동할 때 문제가 없나?
- 실제 환경과 최대한 비슷하게 만듦!
- Controller → Service → Repository → 실제 DB(또는 가짜 DB)에 이르는 전체 데이터 흐름
- Swagger로는 controller까지만 확인 (내부 로직 확인 X)

예외 (Exception) 처리

Error ? Exception

- Error는 http 응답 + 심각한 시스템 문제!
- 일단 발생하면 프로그램이 복구하기가 거의 불가능한 상태 ...
- 개발자가 직접 처리하기보다는 **시스템 레벨**에서 발생하며, 보통 애플리케이션을 종료해야함

오류 종류 (Java Class)	상위 클래스	발생 원인 및 심화 설명
InternalError	VirtualMachineError	JVM 자체의 버그나 손상으로 인해 발생. JVM 내부의 문제(예: 캐시 손상, 코드 생성기 오류)가 발생했으며, 개발자가 고칠 수 없는 가장 심각한 오류
StackOverflowError	VirtualMachineError	메서드 호출이 무한 루프에 빠지는 등 스택 공간이 모두 소진되었을 때 발생
OutOfMemoryError	VirtualMachineError	JVM이 힙(Heap) 메모리를 더 이상 할당할 수 없을 때 발생합니다. (Java Heap space 에러가 가장 흔함)
NoClassDefFoundError	java.lang.LinkageError	컴파일 시점에는 존재했던 클래스가 실행 시점(런타임)에 JVM의 클래스 로더가 찾을 수 없을 때 발생합니다. (배포 오류 또는 환경 설정 문제와 관련됨)
UnknownError	VirtualMachineError	분류할 수 없는 예외로, JVM에 예기치 않은 오류가 발생했을 때 사용됩니다. 매우 드물게 발생합니다.
IOError	java.lang.Error	일반적인 입출력(IOException)이 아닌, 복구 불가능한 심각한 I/O 서브시스템 오류가 발생했을 때 사용됩니다.

Error ? Exception

- “Exception”은 프로그램 내부의 문제 상황
- Checked Exception
- **컴파일 시점**에 처리가 강제되는 예외로, 복구 가능성이 높습니다.

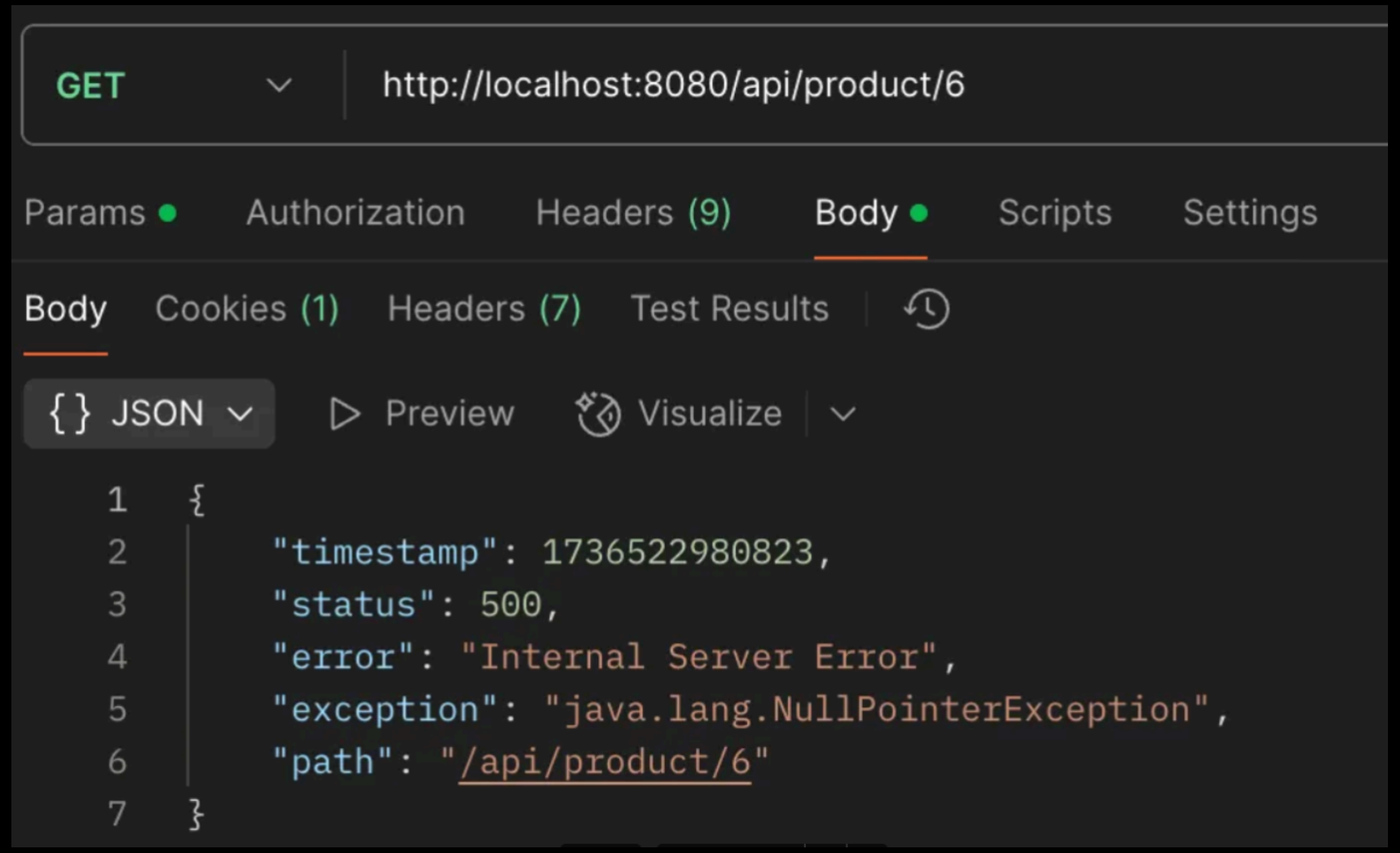
IOException	java.lang.Exception	**출력 작업 중 오류**가 발생했을 때 사용됩니다. (파일 찾기, 읽기/쓰기, 네트워크 연결 문제 등)
FileNotFoundException	IOException	디스크에서 **요청한 파일을 찾을 수 없을 때** 발생합니다.
SQLException	java.lang.Exception	**데이터베이스 접근 및 조작 중 오류**가 발생했을 때 사용됩니다. (쿼리 오류, 연결 끊김 등)
InterruptedException	java.lang.Exception	**스레드(Thread)가 대기, 수면 또는 점유 상태일 때** 다른 스레드로부터 중단 요청을 받았을 때 발생합니다.

Error ? Exception

- ⭐ UnChecked Exception
- **런타임**(실행) 시점에 발생하며, 주로 개발자의 논리적 실수로 인해 발생
- 컴파일러가 처리를 강제하지 않음!!

예외 종류 (Java Class)	상위 클래스	발생 원인 및 심화 설명
NullPointerException (NPE)	RuntimeException	null 값을 가진 객체의 메서드나 필드에 접근하려고 시도했을 때 발생 (가장 흔한 개발 실수)
ArrayIndexOutOfBoundsException	RuntimeException	배열의 유효 범위를 벗어난 인덱스에 접근하려고 시도했을 때 발생
NumberFormatException	IllegalArgumentException	문자열을 숫자로 변환하는 메서드(예: Integer.parseInt())에 숫자가 아닌 문자열이 전달되었을 때 발생
IllegalArgumentException	RuntimeException	메서드에 유효하지 않은 인자가 전달되었을 때 발생 (예: 음수를 허용하지 않는 메서드에 음수 전달)
ClassCastException	RuntimeException	호환되지 않는 타입으로 객체를 강제 형 변환하려고 시도했을 때 발생합니다.

상품 상세 조회 api → 500에러



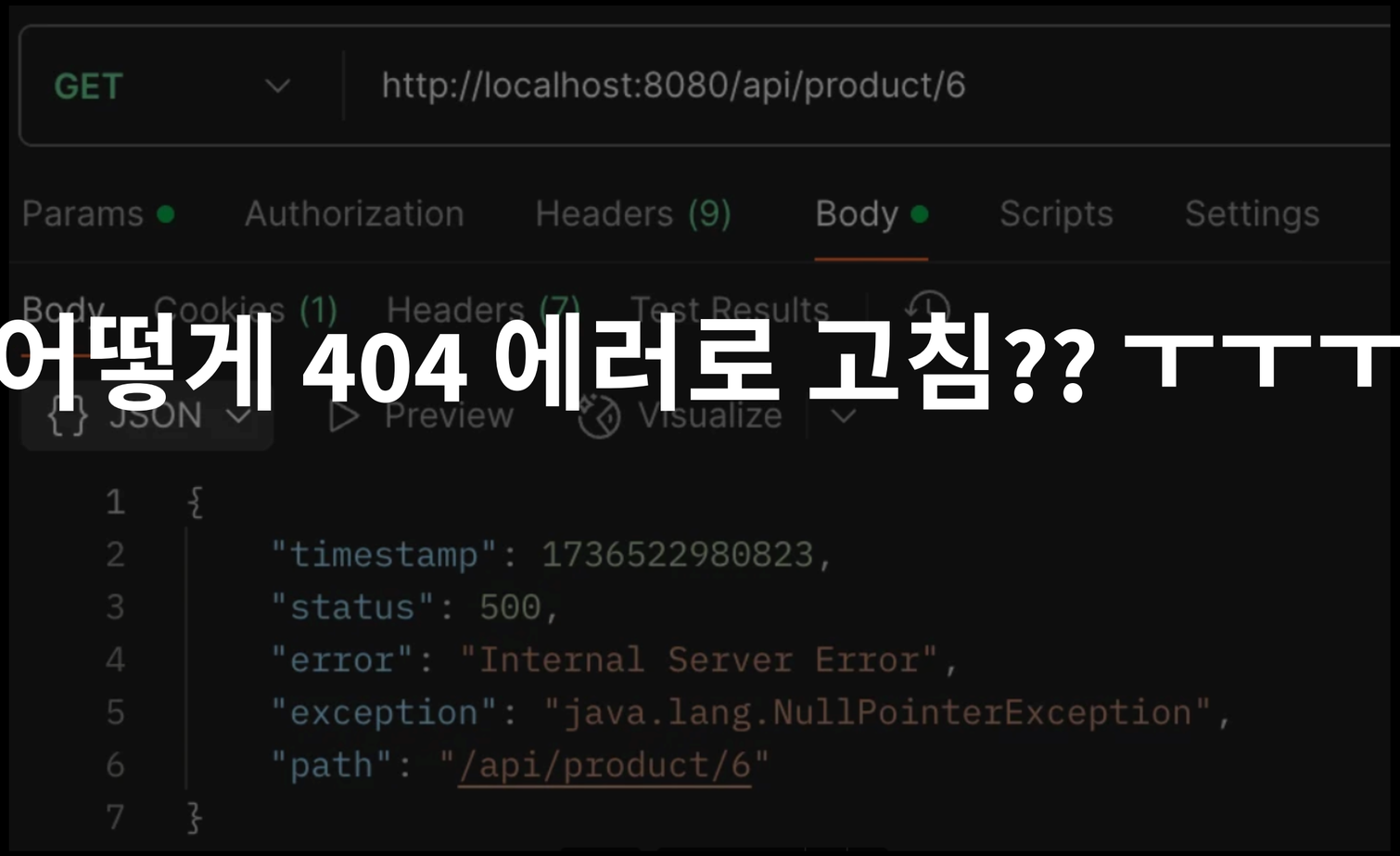
상품 상세 조회 api → 500에러

"내부에서 Exception이 던져졌고(코드 오류 발생) Spring이 그걸 잡지 못했을 때
→ 클라이언트에게 '500 Internal Server Error'를 돌려줌

1. "내부에서 Exception이 던져졌고"
- 예: NullPointerException, IllegalArgumentException 등의 unchecked exception

2. "Spring이 그걸 잡지 못했을 때"
- Spring에는 @ControllerAdvice나 @ExceptionHandler 같은 예외 처리기가 있음
- 이 처리기가 해당 예외를 잡아서(catch) 적절한 응답을 만들어주면, 클라이언트는 친절한 메시지를 받음!!
- 하지만 Spring이 그 예외를 잡지 못하면, 즉 처리하지 않으면…

상품 상세 조회 api → 500에러



어떻게 404 에러로 고침?? ㅜㅜㅜ

@ControllerAdvice, @ExceptionHandler

- Spring Boot에서 **가장 권장되는** 예외 처리 방식
- 애플리케이션의 모든 컨트롤러에서 발생하는 예외를 **한 곳에서** 가로채 처리할 수 있게 함

@RestControllerAdvice
= 전역 예외 처리기

@ExceptionHandler
= 특정 예외 처리 메서드

→ 모든 예외를 한 곳에서 처리!

에러(Exception) 처리

1. 서비스 레이어: 잘못된 상황에서 예외 발생
   - (user == null) {throw new UserNotFoundException("사용자 없음");}

2. 예외를 HTTP 상태 코드로 변환
   - 예외 클래스에 @ResponseStatus(HttpStatus.NOT_FOUND) 어노테이션을 붙임
   - @ControllerAdvice 사용

GlobalExceptionHandler 기본 구조

```java
@RestControllerAdvice  // @ControllerAdvice + @ResponseBody
public class GlobalExceptionHandler {

    // 404 에러
    @ExceptionHandler(UserNotFoundException.class)
    public ResponseEntity<ErrorResponse> handleNotFound(
        UserNotFoundException e
    ) {
        ErrorResponse error = new ErrorResponse(
            "USER_NOT_FOUND",
            e.getMessage()
        );
        return ResponseEntity.status(404).body(error);
    }
```

상품 상세 조회 api → 500에러

“개발 중에는 일시적으로 500을 띄워도 된다!”
하지만 운영 배포 시에는 명시적인 에러코드로 보내주자

과제

- **댓글 API 개발하기**
- 게시물 API 리뷰 반영해서 수정하기
- 처음부터 완벽하게 짜지 않아도 됨!! (다음주차에 리팩토링 과제)
- 과제 기한 10/28 23:59 까지
- swagger 세팅해서 api 동작 캡쳐하기

댓글 요구사항 (/comments)

- 사용자는 로그인을 하지 않고도 댓글을 확인할 수 있어야 합니다.
- 사용자는 자신의 댓글만 수정, 삭제할 수 있어야 합니다.
- 댓글에는 댓글을 달 수 없습니다 (단 원하는 경우 구현해도 괜찮습니다)