

## 6월 6일

### 중복 폴더명 예외처리 설계 및 구현

요구사항

폴더들 저장시, "중복 폴더명 저장 시" 에러를 발생시킴

AS-IS : 기존 동작

TO-BE : 변경 동작

#### AS-IS

중복을 제외한 모든 폴더명이 저장됨

#### TO-BE

Alert 팝업을 띄워서 사용자가 폴더명을 수정할 수 있도록 유도

- 중복된 폴더명 표시

저장 시도한 폴더 모두 저장되지 않음

#### 해결방법

1. 예외 발생 시, 그동안 DB 에 저장된 폴더들을 삭제 하는 방법

```
List<Folder> savedFolderList = new ArrayList<>();
for (String folderName : folderNames) {
    // 2) 이미 생성한 폴더가 아닌 경우만 폴더 생성
    if (isExistFolderName(folderName, existFolderList)) {
        folderRepository.deleteAll(savedFolderList);

        // Exception 발생!
        throw new IllegalArgumentException("중복된 폴더명을 제거해 주세요!");
    } else {
        Folder folder = new Folder(folderName, user);
        // 폴더명 저장
        folder = folderRepository.save(folder);
        savedFolderList.add(folder);
    }
}
```

2. 트랜잭션 (@Transactional) 을 이용하는 방법

```

@Transactional
public List<Folder> addFolders(List<String> folderNames, User user) {
    // 1) 입력으로 들어온 폴더 이름을 기준으로, 회원이 이미 생성한 폴더들을 조회합니다.
    List<Folder> existFolderList = folderRepository.findAllByUserAndName

    List<Folder> savedFolderList = new ArrayList<>();
    for (String folderName : folderNames) {
        // 2) 이미 생성한 폴더가 아닌 경우만 폴더 생성
        if (isExistFolderName(folderName, existFolderList)) {
            // Exception 발생!
            throw new IllegalArgumentException("중복된 폴더명을 제거해 주세요!");
        } else {
            Folder folder = new Folder(folderName, user);
            // 폴더명 저장
            folder = folderRepository.save(folder);
            savedFolderList.add(folder);
        }
    }
}

```

## 트랜잭션의 이해

### 트랜잭션이란?

트랜잭션 : 데이터베이스에서 데이터에 대한 하나의 논리적 실행단계

- 트랜잭션의 특징
  - 더 이상 쪼갤 수 없는 최소단위의 작업
  - 모두 저장되거나, 아무것도 저장되지 않음

1개의 저장요청이 있는 트랜잭션

- DB는 1개의 회원 정보가 안전하게 저장됨을 보장함

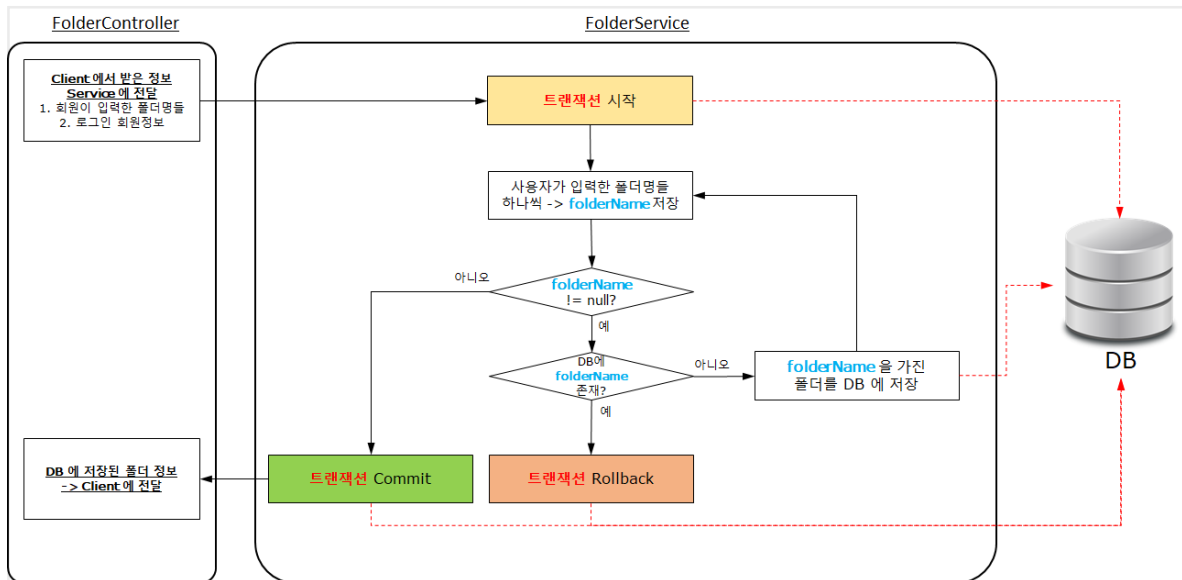
2개 이상 저장요청이 있는 트랜잭션

- 모두 성공 시 -> 트랜잭션 Commit
- 중간에 하나라도 실패 시 -> 트랜잭션 Rollback

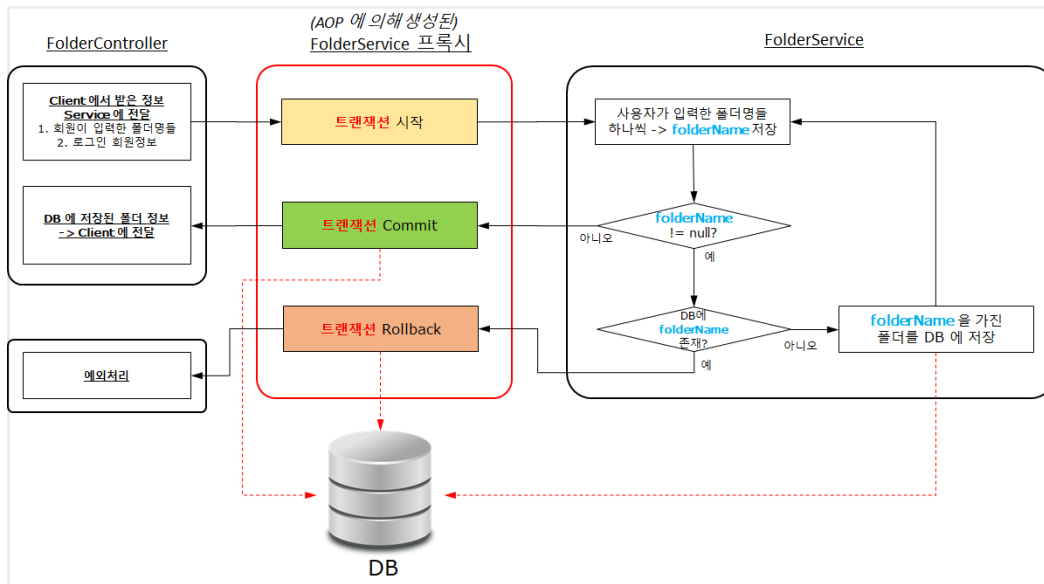
## @Transactional 의 정체

트랜잭션을 사용한 폴더 생성 Flowchart

비즈니스 로직에 트랜잭션 코드가 포함됨



## @Transactional 사용 시 폴더 생성 Flowchart



## 현업에서 DB 운영 방식 (Primary, Replica)

### 현업에서 DB 운영방식

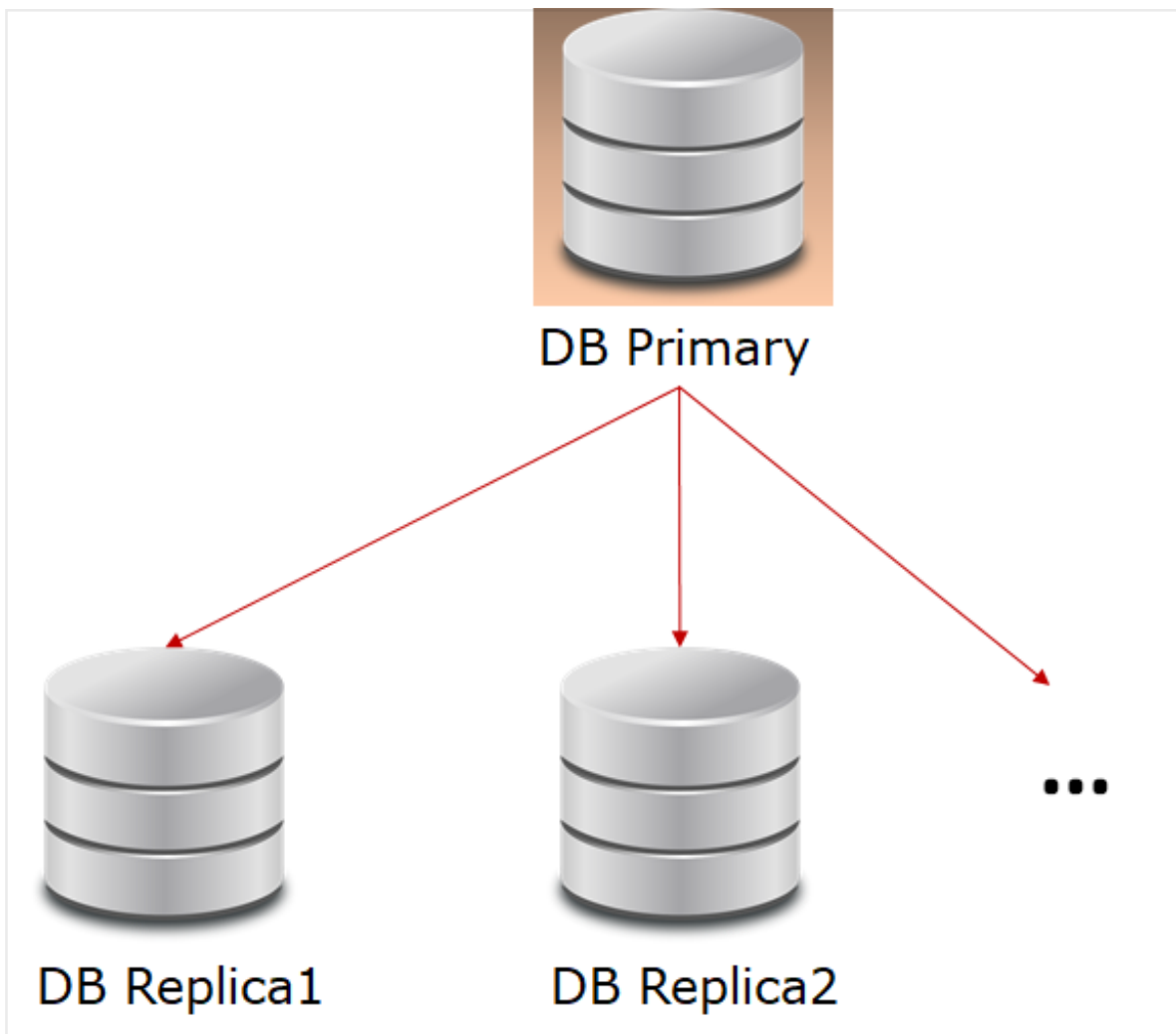
- 웹 서비스에서 DB에 담겨있는 Data는 소중한 자산 (Ex. 회원정보, 서비스 이용 정보)
- DB 훼손 가능성
  - DB도 결국 물리적 HDD가 존재함
  - DB가 저장된 HDD나 DB가 저장된 컴퓨터 자체가 고장 가능성 있음
- 현업에서는 2대 이상에 DB를 운영함



DB에서 데이터를 CRUD 하는 과정에서 DB간에 싱크가 맞지않아 데이터 불일치가 발생할 수 있음

Primary / Replica 운영방식

쓰기 전용 DB (Primary) 란 읽기 전용 DB (Replica) 를 구분



Primary : 쓰기 전용

@Transactional 의 readOnly 속성

```
@Transactional(readOnly = false)
```

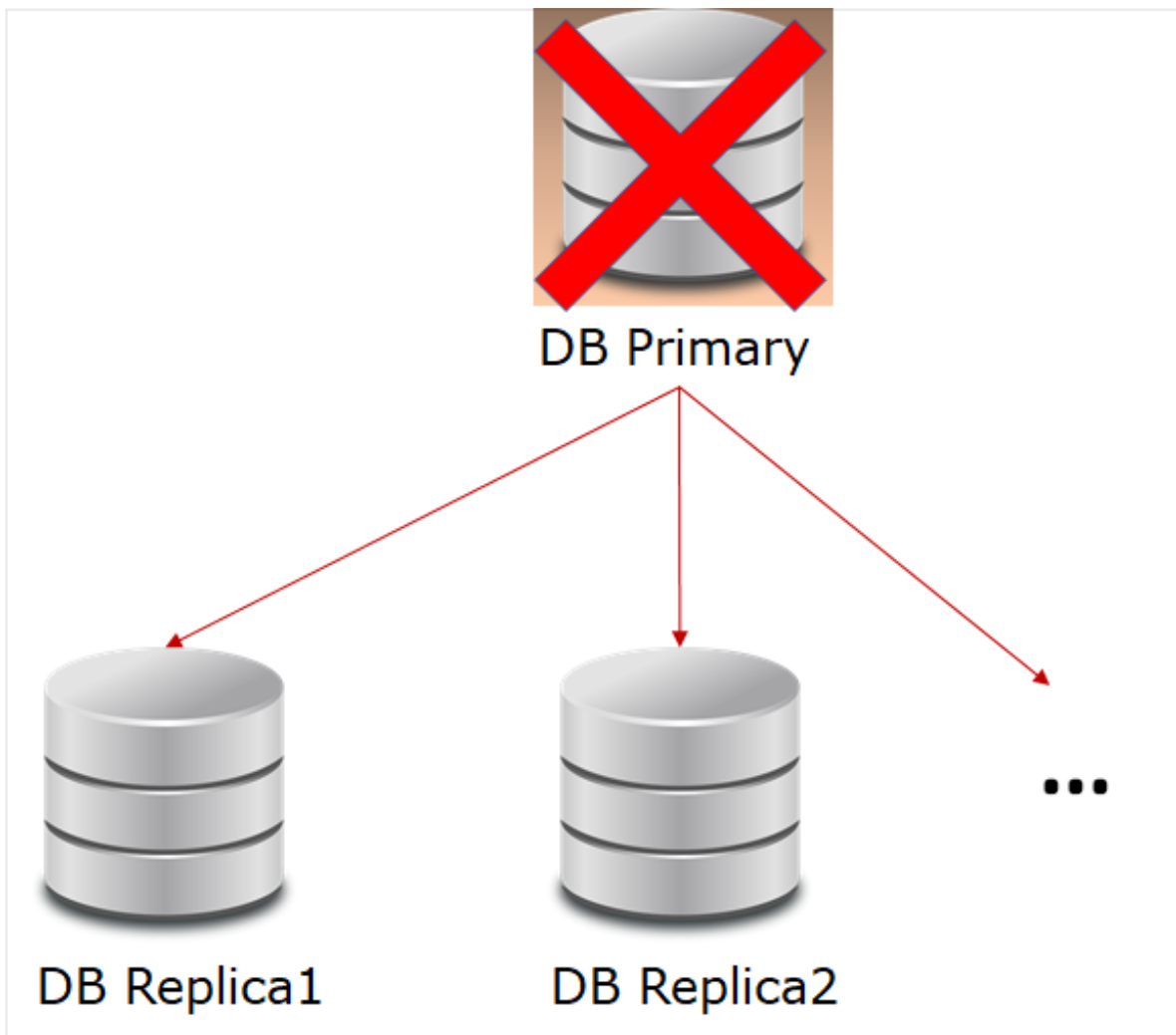
readOnly 를 코드에 적지 않으면, 기본값은 false

Write 된 Data (Create, Update, Delete) 가 Replica 로 Sync 됨(Replication)  
Replica (Secondary) : 읽기 전용

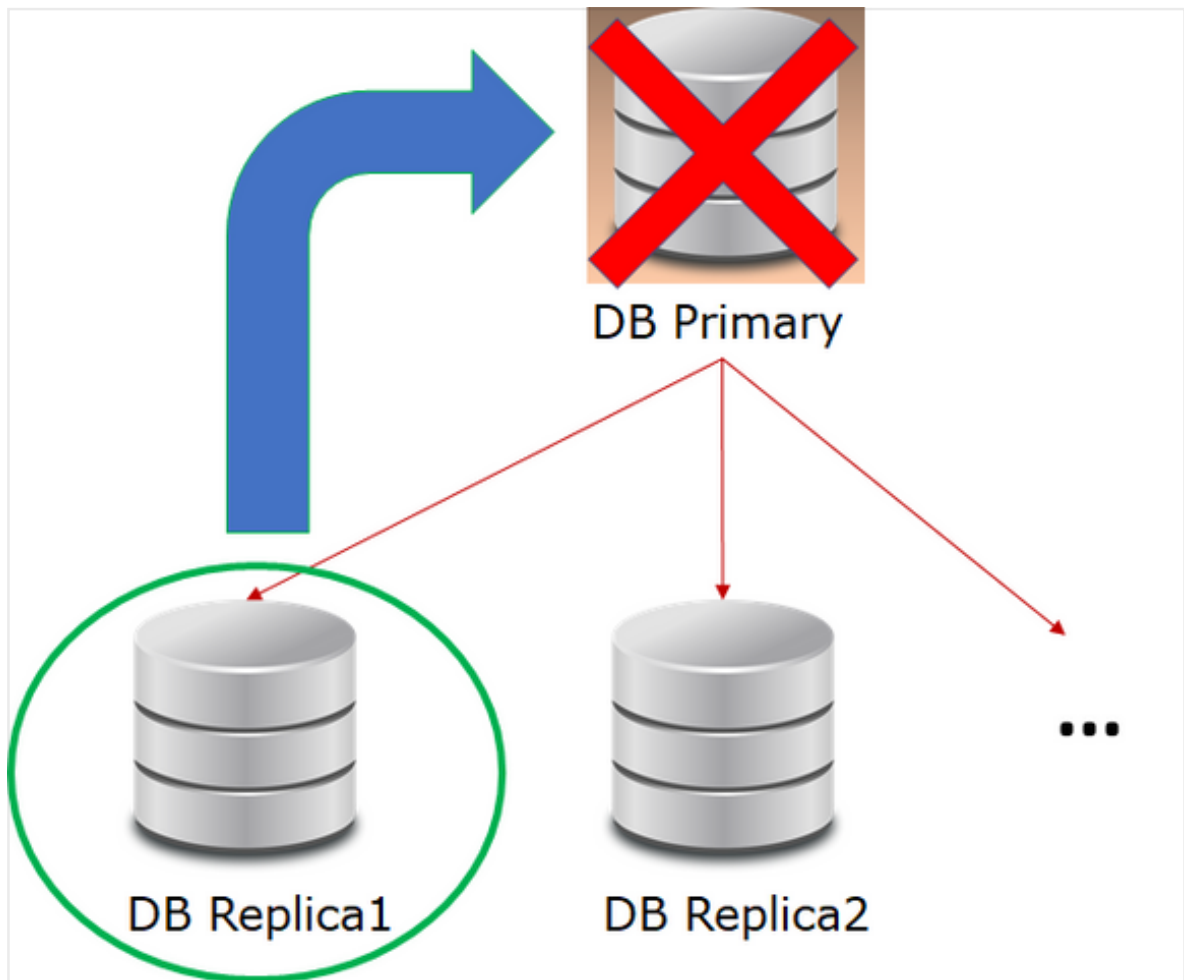
```
@Transactional(readOnly = true)
```

위 개념은 스프링에서 Primary DB endpoint, Replica DB endpoint 를 설정해야지만 가능

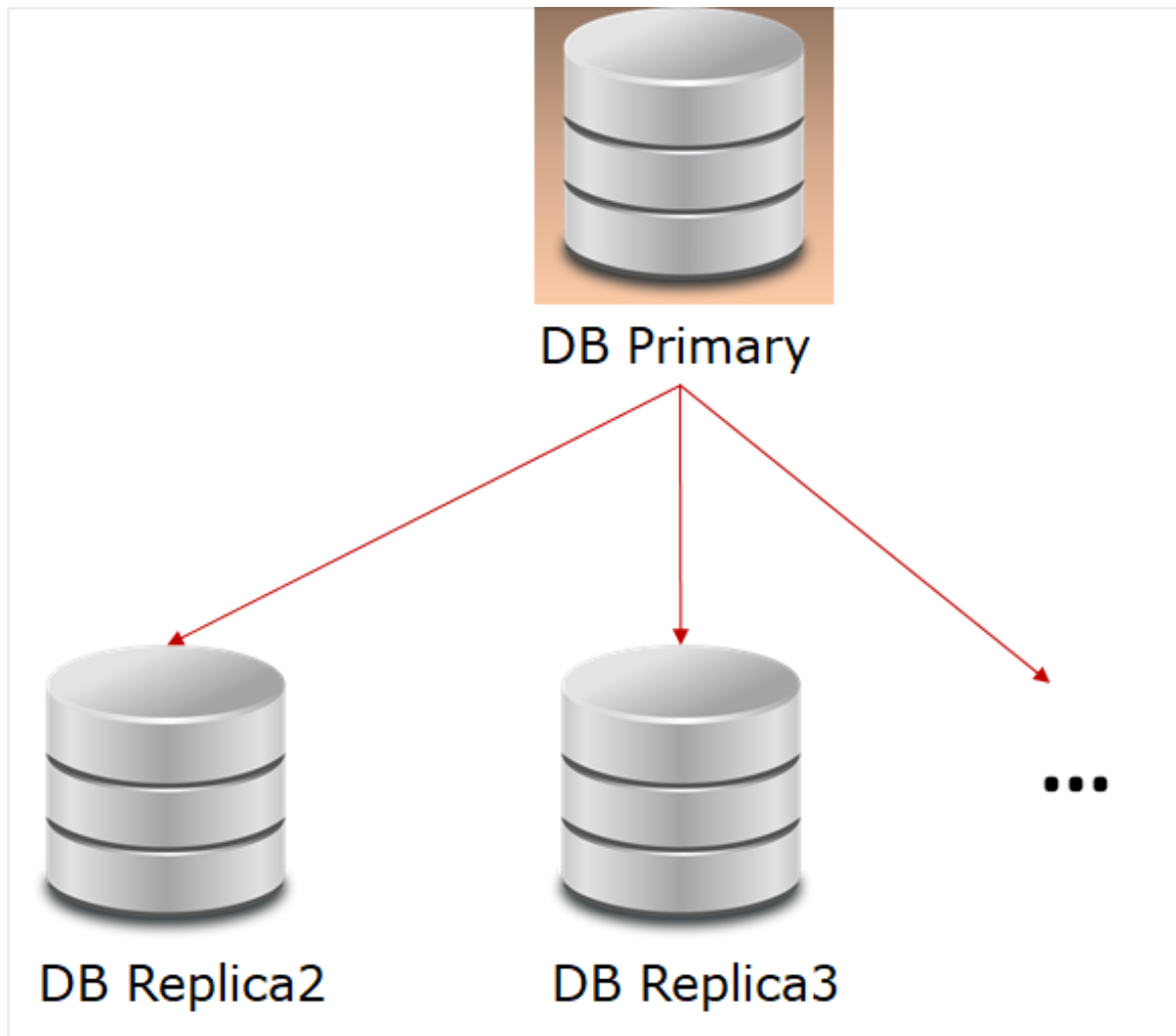
**Primary** 에 문제가 생겼을 때



Replica 중 1개가 Primary 가 됨



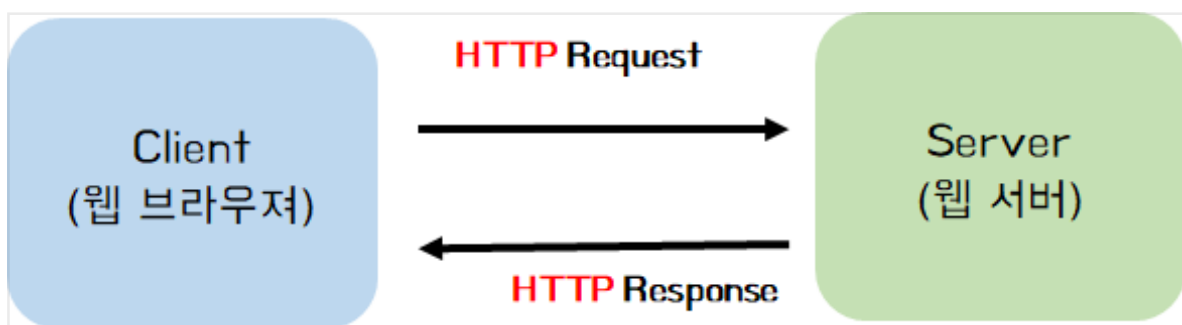
그래서 다시 Primary - Replica 로 정상 운영

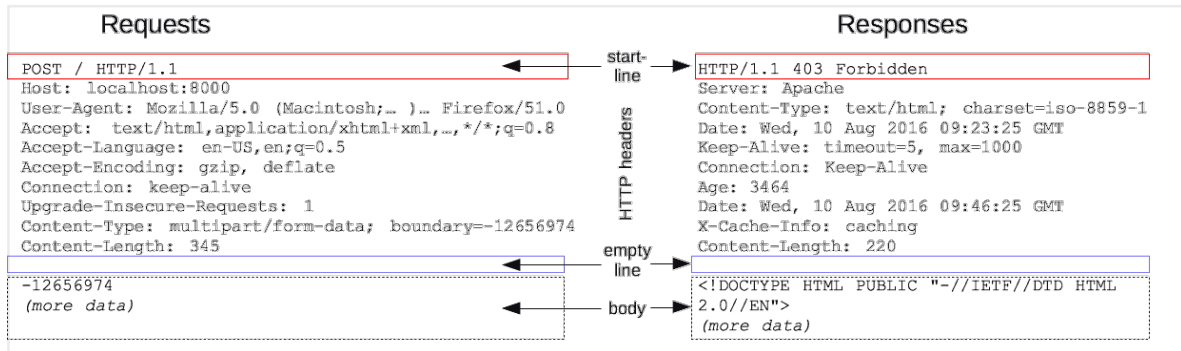


스프링 예외 처리 방법

스프링 기본 에러 처리

HTTP 에러 메시지 전달 방법 이해





## - Response 메시지

### 1. 상태줄 : API 요청 결과 (상태 코드, 상태 텍스트)

## HTTP/1.1 404 Not Found

### HTTP 상태 코드 종류

1. 2xx Success
2. 4xx Client Error
3. 5xx Server Error

### 2. 헤더

#### "Content type"

1. 없음
2. Response 본문 내용이 HTML인 경우

## Content type: text/html

### 3. Response 본문 내용이 JSON 인 경우

## Content type: application/json

### 3. 본문

#### a. HTML

```
<!DOCTYPE html>
<html>
  <head><title>By @ResponseBody</title></head>
  <body>Hello, Spring 정적 웹 페이지!!</body>
</html>
```

#### b. JSON




```
{
  "name": "홍길동",
  "age": 20
}
```

현재 에러 발생 시 HTTP 에러 메시지 확인

1. DB 에 중복된 폴더명 존재 시 Exception 발생

```
throw new IllegalArgumentException("중복된 폴더명을 제거해 주세요! 폴더명: " + folder
```

2. 클라이언트에서 에러 메시지 확인  
상태줄

×	Headers	Preview	Response	Initiator	Timing
▼ General					
Request URL: http://localhost:8080/api/folders					
Request Method: POST					
Status Code:  500					
Remote Address: [::1]:8080					
Referrer Policy: strict-origin-when-cross-origin					

헤더

▼ Response Headers	View source
Cache-Control: no-cache, no-store, max-age=0	
Connection: close	
Content-Type: application/json	

HTTP 500

To Do

1. HTTP 500 -> 400 Client Error
2. 응답 본문(body) 내용 정의
  - 에러 시 전달되는 본문 정리 -> 프론트엔드 개발자와 공유
    1. errorMessage : 에러 내용
    2. httpStatus : 스프링에 선언된 HttpStatus 값

```

public enum HttpStatus {
    // 1xx Informational
    CONTINUE(100, Series.INFORMATIONAL, "Continue"),
    // ...

    // 2xx Success
    OK(200, Series.SUCCESSFUL, "OK"),
    CREATED(201, Series.SUCCESSFUL, "Created"),
    // ...

    // 3xx Redirection
    MULTIPLE_CHOICES(300, Series.REDIRECTION, "Multiple Choices"),
    MOVED_PERMANENTLY(301, Series.REDIRECTION, "Moved Permanently"),
    FOUND(302, Series.REDIRECTION, "Found"),
    // ...

    // --- 4xx Client Error ---
    BAD_REQUEST(400, Series.CLIENT_ERROR, "Bad Request"),
    UNAUTHORIZED(401, Series.CLIENT_ERROR, "Unauthorized"),
    PAYMENT_REQUIRED(402, Series.CLIENT_ERROR, "Payment Required"),
    FORBIDDEN(403, Series.CLIENT_ERROR, "Forbidden"),
    // ...

    // --- 5xx Server Error ---
    INTERNAL_SERVER_ERROR(500, Series.SERVER_ERROR, "Internal Server Error"),
    NOT_IMPLEMENTED(501, Series.SERVER_ERROR, "Not Implemented"),
    BAD_GATEWAY(502, Series.SERVER_ERROR, "Bad Gateway"),
    // ...
}

```

Sample

```

{
  "errorMessage": "중복된 폴더명을 제거해 주세요! 폴더명: 신발",
  "httpStatus": "BAD_REQUEST"
}

```

스프링 예외 처리 방법

백엔드: 예외처리

스프링이 제공하는 **ResponseEntity** 클래스 사용

**ResponseEntity**: HTTP response object 를 위한 Wrapper

선언가능 목록

- HTTP status code
- HTTP headers
- HTTP body

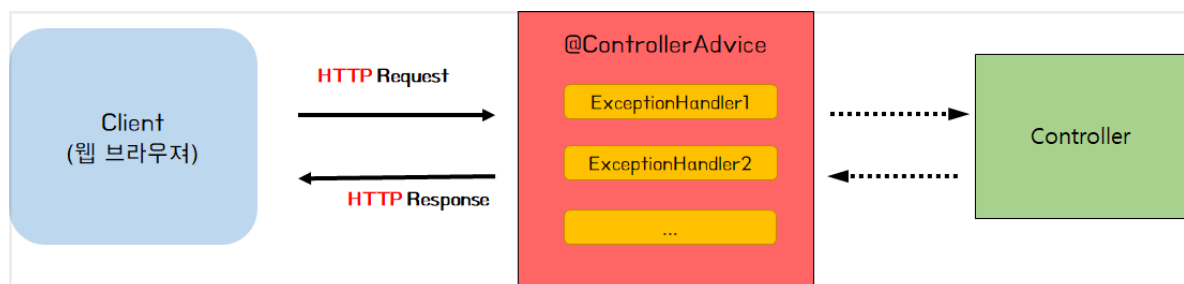
스프링 Global 예외 처리 방법

Global 예외 처리 방법

관심상품, 희망최저가 등록 시

```
public Product updateProduct(Long id, ProductMypriceRequestDto requestDto) {  
    int myprice = requestDto.getMyprice();  
    if (myprice < MIN_MY_PRICE) {  
        throw new IllegalArgumentException("유효하지 않은 관심 가격입니다. 최소 " + MIN_MY_PRICE + " 원 이상으로 설정해 주세요.");  
    }  
}
```

Global 예외 처리



@controllerAdvice 사용

@RestControllerAdvice

@ControllerAdvice + @ResponseBody

ErrorCode 선언

서비스 전체에 사용할 에러코드들 (ErrorCode) 을 선언

예외발생 시 서버 및 클라이언트에서 선언한 ErrorCode 사용

1. httpStatus: HTTP 상태코드

2. errorCode: 에러 코드

a. 에러 종류별 Unique 한 에러코드를 소유

b. 국제화에 사용가능

l. 클라이언트가 사용하는 언어 (한국어, 영어, 일본어 등) 에 따라

에러메세지를 다르게 보여줌

3. errorMessage

**OOP VS AOP**

**OOP** 는 핵심기능을 모듈화

**AOP** 는 부가기능 을 모듈화

부가기능의 예

API 시간 측정, 트랜잭션, 예외처리, 로깅 등

**AOP 는 OOP 를 “보완” 해줌**

**자바 컴파일 과정 공부해보기**