



It's Your Life

with





스프링의 예외 처리



全知全能

전

온전할

지

알

전

온전할

능

능할

어떠한 사물이라도 잘 알고, 모든 일을 다 행할 수 있음. 또는 그런 능력.



번역 <http://ntx.wiki>





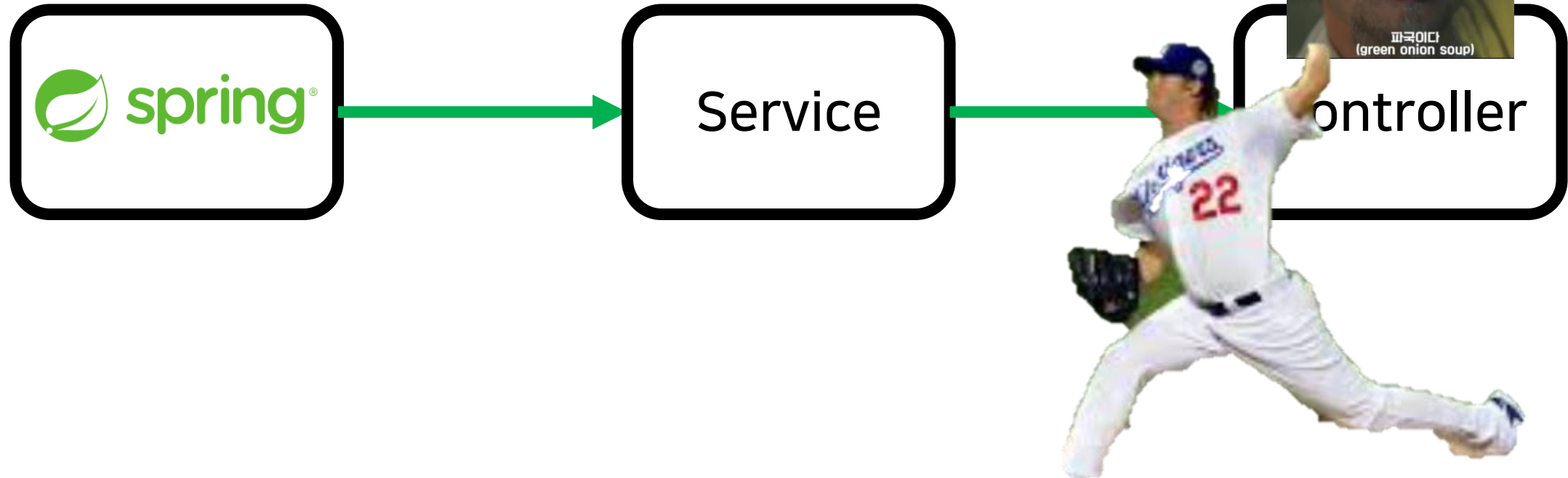
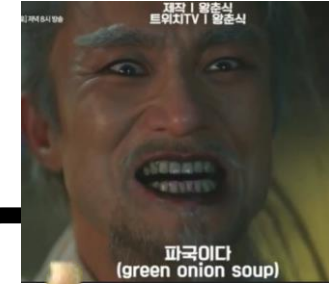
mbc



자바 수업 때 회사에서 쓰는
예외처리 방법 기억 하시는 분!?

아련

예외 발생



Service



좌랑해

선택 A

그래!
내가 처리 하겠어!

인생극장

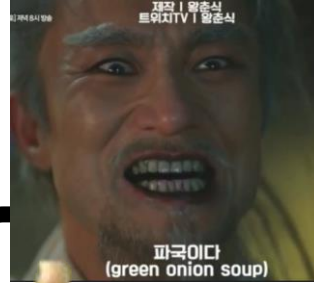
선택 B

아니...
난 처리 못하지!

그래! 결심했어!



예외 발생



Service

Controller





예외 발생



spring®

Service

Controller



예외 발생



Service

Controller







명시적 예외
처리 코드 추가

명시적 예외
처리 코드 추가

명시적 예외
처리 코드 추가

main

Service

Client





대표 예외에 대한 처리!

@ControllerAdvice



- 전역적인 처리가 필요할 때 쓰는 어노테이션!
- 보통 전역 처리가 필요한 곳에 사용
 - 예외 처리
 - 전역 데이터 바인딩 - 모든 요청에서 공통으로 사용되는 데이터를 바인딩
 - 모델 객체 변환 - 모든 요청에서 공통으로 사용되는 데이터를 모델에 바인딩 하여 전달
- 해당 컨트롤러를 예외 처리용 클래스에 붙이면 전역에서 관리되는 예외 객체를 자동으로 받아올 수 있습니다!



404 에러 처리





그란데 말입니다

그란데 말입니다

404 에러는 어떤 에러였죠!?





HTTP Status

- 100 번째 : 정보 / 리퀘스트를 받고 처리 중
- 200 번째 : 성공 / 리퀘스트를 정상 처리
- 300 번째 : 리디렉션 / 처리 완료를 위해서는 추가 동작 필요
- 400 번째 : 클라이언트 에러 / 클라이언트에서 요청을 잘못 보냄
- 500 번째 : 서버 에러 / 리퀘스트는 잘 들어 갔지만 서버에서 처리를 못함



HTTP Status Codes

Level 200 (Success)

200 : OK

201 : Created

203 : Non-Authoritative
Information

204 : No Content

Level 400

400 : Bad Request

401 : Unauthorized

403 : Forbidden

404 : Not Found

409 : Conflict

Level 500

500 : Internal Server Error

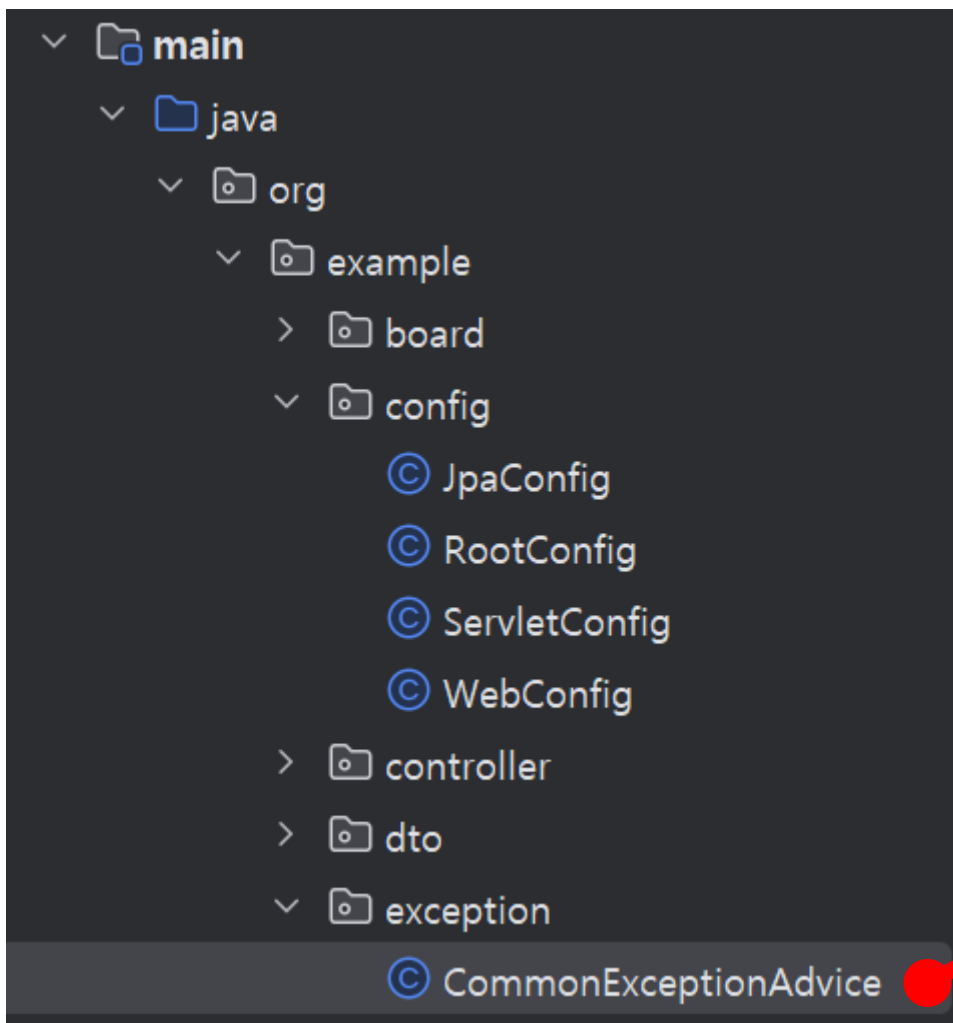
503 : Service Unavailable

501 : Not Implemented

504 : Gateway Timeout

599 : Network timeout

502 : Bad Gateway



예외 처리를 위한
exception 패키지와
CommonExceptionHandler 클래스 생성!

```
@ControllerAdvice
```

```
@Log4j
```

```
public class CommonExceptionHandler {
```

```
    private String context = "/exception"; 2 usages
```

```
    @ExceptionHandler(NoHandlerFoundException.class) new *
```

```
    @ResponseStatus(HttpStatus.NOT_FOUND)
```

```
    public String handle404(NoHandlerFoundException e) {
```

```
        return context + "/404";
```


```
    }
```

예외처리를 위한 핸들러임을
@ContorollerAdvice 어노테이션으로 명시

@ControllerAdvice kdtTetz *

@Log4j

```
public class CommonExceptionAdvice {  
    private String context = "/exception";  
  
    @ExceptionHandler(NoHandlerFoundException.class) new *  
    @ResponseStatus(HttpStatus.NOT_FOUND)  
    public String handle404(NoHandlerFoundException e) {  
        return context + "/404";  
    }  
}
```



@ExceptionHandler 어노테이션을 사용하여
특정 예외에 대한 에러 객체를 받아오도록 설정

@ControllerAdvice kdtTetz *

@Log4j

```
public class CommonExceptionHandler {  
    private String context = "/exception";  
  
    @ExceptionHandler({NoHandlerFoundException.class})  
    @ResponseStatus(HttpStatus.NOT_FOUND)  
    public String handle404(NoHandlerFoundException e) {  
        return context + "/404";  
    }  
}
```

404 에러에 대한 예외만 받아야 하므로
받아야하는 클래스를
NoHandlerFoundException 으로 한정



```
@ControllerAdvice  kdtTetz *
@Log4j
public class CommonExceptionHandler {
    private String context = "/exception"; 2 usages

    @ExceptionHandler(NoHandlerFoundException.class)
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public String handle404(NoHandlerFoundException e) {
        return context + "/404";
    }
}
```

404 임을 표현하기 위한
헤더를 자동으로 설정

views 폴더의 404.jsp 페이지로
이동시켜서 사용자를 배려



webapp
WEB-INF
views
exception
JSP 404.jsp

views 폴더에 exception 폴더를 만들고
404.jsp 페이지 만들기




```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>404</title>
</head>
<body>
<%@ include file="../header2.jsp"%>
<h1>404, 존재하지 않는 페이지 입니다</h1>
<a href="/">홈 페이지로 돌아가기</a>
</body>
</html>
```

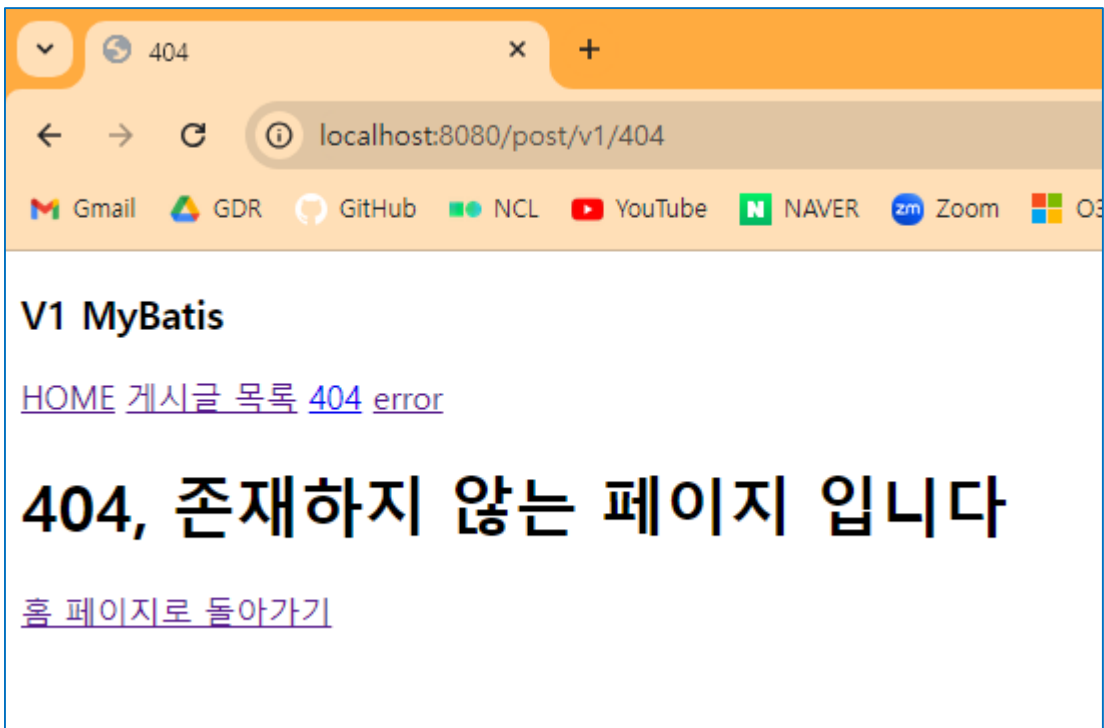
적절한 내용을 출력!

<https://github.com/xenosign/spring-code-repo/blob/main/404.jsp>



```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<header>
  <h3>V1 MyBatis</h3>
  <a href="/">HOME</a>
  <a href="/post/v1/show">게시글 목록</a>
  <a href="/post/v1/404">404</a>
   <a href="/post/v1/error">error</a>
</header>
```

편안한 확인을 위해서
header2.jsp 에 주소 추가





500 에러 처리



500번 에러는 서버에서 발생한 예상 못한 모든 예외를 처리해야 하므로 모든 예외의 부모 클래스인 Exception 클래스를 받습니다!

```
@ExceptionHandler(Exception.class)  kdtTetz *
public String exception(Exception e, Model model) {
    log.error(e.getMessage());
    e.printStackTrace();

    model.addAttribute(attributeName: "errorMessage", e.getMessage());
    model.addAttribute(attributeName: "stackTrace", Arrays.asList(e.getStackTrace()));

    return context + "/error-page";
}
```



```
@ExceptionHandler(Exception.class)  👤 kdtTetz *  
public String exception(Exception e, Model model) {  
    log.error(e.getMessage());  
    e.printStackTrace();  
  
    model.addAttribute(attributeName: "errorMessage", e.getMessage());  
    model.addAttribute(attributeName: "stackTrace", Arrays.asList(e.getStackTrace()));  
  
    return context + "/error-page";  
}
```

log.error 메서드로
어떤 예외가 발생했는지
대표 메시지를 출력



```
@ExceptionHandler(Exception.class)  👤 kdtTetz *  
public String exception(Exception e, Model model) {  
    log.error(e.getMessage());  
    e.printStackTrace();  
  
    model.addAttribute(attributeName: "errorMessage", e.getMessage());  
    model.addAttribute(attributeName: "stackTrace", Arrays.asList(e.getStackTrace()));  
  
    return context + "/error-page";  
}
```

에러가 발생한 순서를 기록한
stack 을 순서대로 출력!



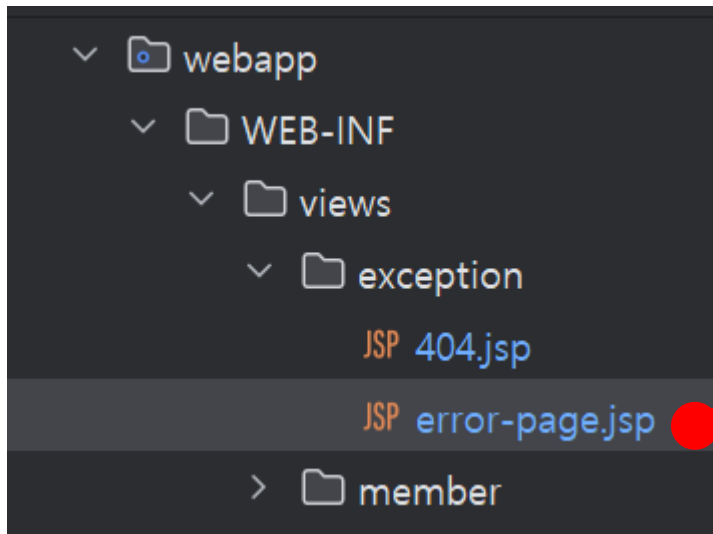
```
@ExceptionHandler(Exception.class)  kdtTetz *
public String exception(Exception e, Model model) {
    log.error(e.getMessage());
    e.printStackTrace();

    model.addAttribute(attributeName: "errorMessage", e.getMessage());
    model.addAttribute(attributeName: "stackTrace", Arrays.asList(e.getStackTrace()));

    return context + "/error-page";
}
```

에러 페이지에 에러 내용을 전달하기 위해서 model 에 데이터를 바인딩해서 전달!

/views/exception 폴더의 error-page.jsp 를 호출!



views/exception 폴더에
error-page.jsp 파일 생성!

<https://github.com/xenosign/spring-code-repo/blob/main/error-page.jsp>



```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" import="java.util.*"%>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>500</title>
</head>
<body>
<%@ include file="../header2.jsp"%>
<h1>죄송합니다. 예상치 못한 에러가 발생했습니다.</h1>
<a href="/">홈 페이지로 돌아가기</a>
<h4><c:out value="${errorMessage}"></c:out></h4>
<ul>
    <c:forEach items="${stackTrace}" var="stack">
        <li><c:out value="${stack}"></c:out></li>
    </c:forEach>
</ul>
</body>
</html>
```

model 을 통해 전달 받은
에러 메시지 출력

model 을 통해 전달 받은
에러 스택 출력!

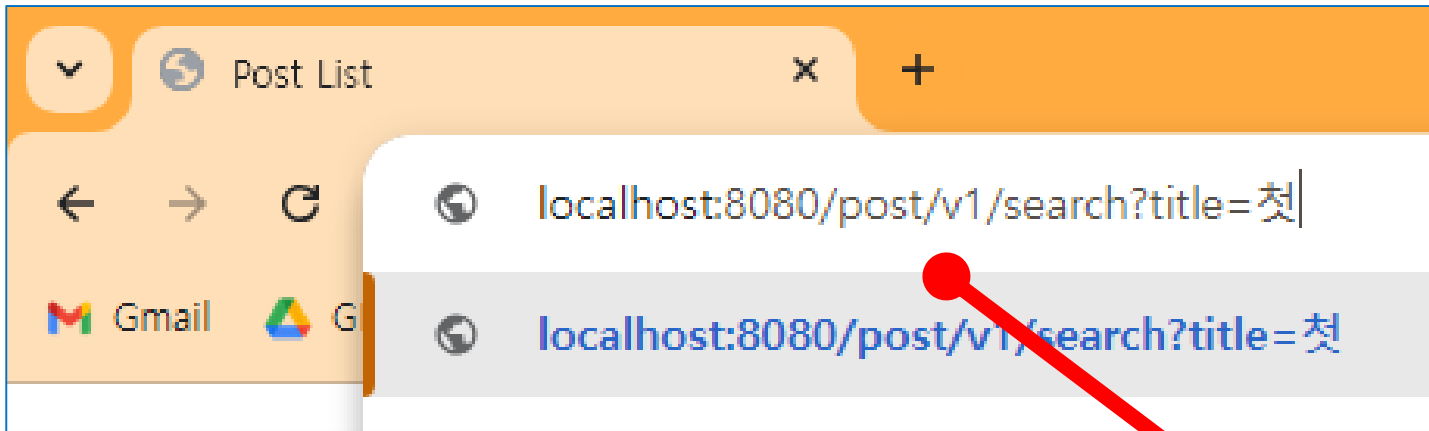


그란데 말입니다

그란데 말입니다

예상치 못한 예외를 저희는
어떻게 만들 수 있죠!?





어제 POSTMAN 테스트에서
검색 조건 중에 하나를 누락하면
예외가 발생했던 것 기억 하시나요!?



V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

죄송합니다. 예상치 못한 에러가 발생했습니다.

[홈 페이지로 돌아가기](#)

Required request parameter 'content' for method parameter type String is not present

- org.springframework.web.method.annotation.RequestMappingHandlerMethod
- org.springframework.web.method.annotation.RequestMappingHandlerMapping
- org.springframework.web.method.annotation.RequestMappingHandlerMapping
- org.springframework.web.method.support.HandlerMethodInvoker
- org.springframework.web.method.support.HandlerMethodInvoker
- org.springframework.web.method.support.HandlerMethodInvoker

content 파라미터가 안들어 왔다는
예상치 못한 예외 발생

→ 전역에서 발생한 예외를
@ControllerAdvice 어노테이션이 잡아서
CommonExceptionHandler 클래스에 전달!



그림으로 봄시다!





@ControllerAdvice



Service

Controller

예외 발생



?)의 운동신경 이규한





500 에러 발생 시키기



가...갑자기요?



controller
member
post
JpaPostController
PostController
RestPostController

특정 주소 요청에 에러를 발생 시켜야 하므로

기존에 사용 중이던 PostController 에
에러 발생 기능을 추가!

```
// 에러
@GetMapping("/error")
public String error(HttpServletRequest request) {
    log.info("=====> 게시물 목록 페이지 호출, " + request.getRequestURI());

    if (true) {
        throw new RuntimeException("의도적으로 발생시킨 예외");
    }
    return context + "/post-show";
}
```

/post/v1/error 요청이 들어오면
에러를 발생시키는 메서드를 실행

런타임 에러를 강제로 발생 시키기!



500

localhost:8080/post/v1/error

Gmail GDR GitHub NCL YouTube NAVER Zoom O365 NAVER INF

V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

죄송합니다. 예상치 못한 에러가 발생했습니다.

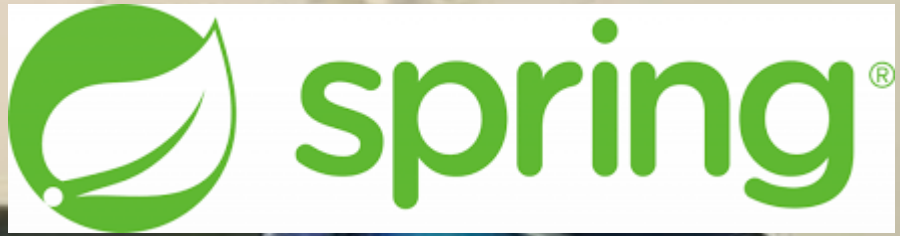
[홈 페이지로 돌아가기](#)

의도적으로 발생시킨 예외

- org.example.controller.post.PostContro
- java.base/jdk.internal.reflect.NativeMet
- java.base/jdk.internal.reflect.NativeMet
- java.base/jdk.internal.reflect.Delegating
- java.base/java.lang.reflect.Method.invo

header 의 error 를 클릭하면 error 컨트롤러가 실행되어 런타임 에러가 발생!

→ 처리 되지 못한 예외는 결국 스프링 서비스로 던져지게 되고 해당 예외는 @ControllerAdvice 에 의해서 CommonExceptionAdvice 클래스에 전달!



@ControllerAdvice

>>>

나는 믿을거야. 가코 믿을거야





```
@PostMapping(🌐"/delete")  👤 kdtTetz
public String postDelete(@RequestParam("id") String id, HttpSe
    log.info("=====> 게시글 삭제 기능 호출, " + request

    long postId = Long.parseLong(id);
    int affectedRows = postRepository.delete(postId);

    if (affectedRows > 0) log.info("삭제 성공");

    return "redire
}
```

이제 DB 통신 등등에서 예상치 못한 예외가 발생해도
해당 컨트롤러 내부에서 직접 예외를 처리해줄 필요가 없습니다!!

→ 즉, 코드가 깔끔해 집니다!!

@ControllerAdvice kdtTetz *

@Log4j

```
public class CommonExceptionAdvice {
```



10:36



그란데 말입니다

그란데 말입니다

모든 예외를 이런 식으로 처리하는게 맞을까요!?





당연히 예측 가능한 예외에 대한 처리는
직접 처리를 해서 사용자가 최대한
error-page 를 안보도록 만드는 것이
좋은 서비스 입니다!!