Chapter 7-1 Exception Handling

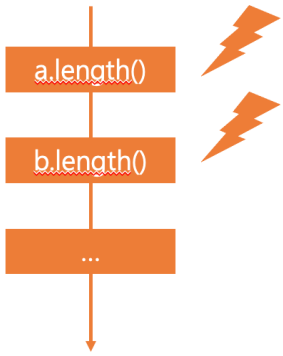
Java의 예외처리
Spring Boot의 예외처리 방법들
@ExceptionHandler

Java의 예외처리

Exception - 예외

특수한 처리를 필요로 하는 비상적 또는 예외적 상황

```
// a 또는 b가 null일 경우 NullPointerException이 발생
public void compareLength(String a, String b) {
    int aLength = a.length();
    int bLength = b.length();
    // 이하 생략
}
```




```
public void compareLength(String a, String b) {
    // try 안쪽의 코드에서 예외가 발생하면
    try {
        int aLength = a.length();
        int bLength = b.length();
        // 이하 생략
    // catch에서 지정한 예외가 발생했을 때의 처리법을 정의합니다.
    } catch (NullPointerException e) {
        System.out.println("null pointer exception, pass");
    } finally {
        // 예외의 발생 여부와 관계없이 항상 실행하는 코드
    }
}
```

예외가 일어날 수 있는 구역

```java
public void compareLength(String a, String b) {
    // try 안쪽의 코드에서 예외가 발생하면
    try {
        int aLength = a.length();
        int bLength = b.length();
        // 이하 생략
    // catch에서 지정한 예외가 발생했을 때의 처리법을 정의합니다.
    } catch (NullPointerException e) {
        System.out.println("null pointer exception, pass");
    } finally {
        // 예외의 발생 여부와 관계없이 항상 실행하는 코드
    }
}
```

예외가 일어날 수 있는 구역

예외가 발생할 시 실행

```java
public void compareLength(String a, String b) {
    // try 안쪽의 코드에서 예외가 발생하면
    try {
        int aLength = a.length();
        int bLength = b.length();
        // 이하 생략
    // catch에서 지정한 예외가 발생했을 때의 처리법을 정의합니다.
    } catch (NullPointerException e) {
        System.out.println("null pointer exception, pass");
    } finally {
        // 예외의 발생 여부와 관계없이 항상 실행하는 코드
    }
}
```

예외가 일어날 수 있는 구역

예외가 발생할 시 실행

예외의 발생과 무관

`finally` : 예외의 발생과 무관

ex) 파일을 다룰 때, 파일을 닫았다는 부분을 검증하는 부분

```java
public void compareLength(String a, String b) throws NullPointerException {
    int aLength = a.length();
    int bLength = b.length();
    // 이하 생략
}
```

예외 처리를 호출하는 대상에게 전가

Java에서는 Method Signature의 일부로, 처리되지 않은 예외는 Compile Error를 발생시킨다.

* RuntimeException 제외

함수 호출 대상이 method signature를 보고 이 예외를 발생시킬 가능성이 존재하는구나 라고 생각하면서 처리해야된다 라고 생각하게됨!

Spring Boot의 예외처리 방법들

1. ResponseStatusException - 단발적 예외
2. @ExceptionHandler - Controller 내부 예외
3. HandlerExceptionResolver - 예외 처리 Handler
4. @ControllerAdvice - ExceptionHandler 모음

`ControllerAdvice` : • @Controller나 @RestController에서 발생한 예외를 한 곳에서 관리하고 처리할 수 있게 도와주는 어노테이션

@ExceptionHandler

@ExceptionHandler같은 경우는 @Controller, @RestController가 적용된 Bean내에서 발생하는 예외를 잡아서 하나의 메서드에서 처리해주는 기능을 한다.

```
@RestController
public class MyRestController {
    ...
    ...
    @ExceptionHandler(NullPointerException.class)
    public Object nullex(Exception e) {
        System.err.println(e.getClass());
        return "myService";
    }
}
```

위와 같이 적용하기만 하면 된다. @ExceptionHandler라는 어노테이션을 쓰고 인자로 캐치하고 싶은 예외클래스를 등록해주면 끝난다.

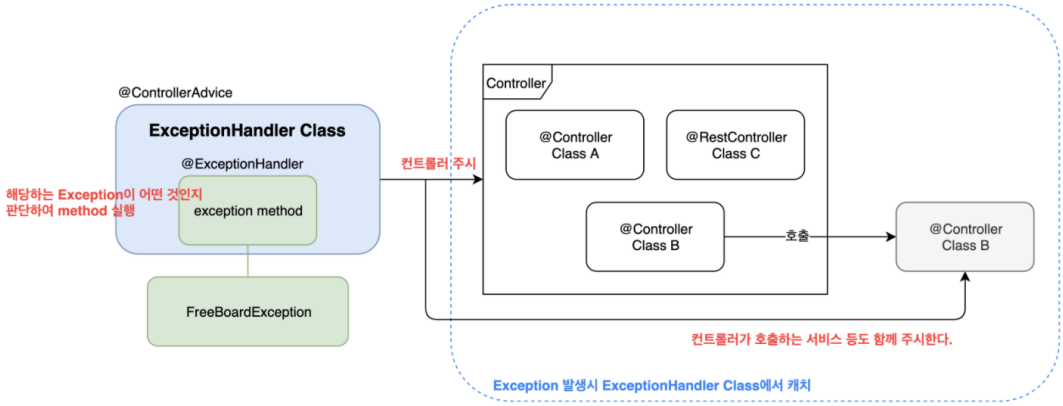
→ @ExceptionHandler({ Exception1.class, Exception2.class}) 이런식으로 두 개 이상 등록도 가능하다.

위의 예제에서 처럼하면 MyRestController에 해당하는 Bean에서 NullPointerException이 발생한다면, @ExceptionHandler(NullPointerException.class)가 적용된 메서드가 호

출될 것이다.

주의사항/알아 둘 것

- Controller, RestController에만 적용가능하다. (@Service같은 빈에서는 안됨.)
- 리턴 타입은 자유롭게 해도 된다. (Controller내부에 있는 메서드들은 여러 타입의 response를 할 것이다. 해당 타입과 전혀다른 리턴 타입이어도 상관없다.)
- @ExceptionHandler를 등록한 Controller에만 적용된다. 다른 Controller에서 NullPointerException이 발생하더라도 예외를 처리할 수 없다.
- 메서드의 파라미터로 Exception을 받아왔는데 이것 또한 자유롭게 받아와도 된다.



@Controlleradvice, @ExceptionHandler

@Controlleradvice, @ExceptionHandler에 대해 알아보자

https://incheol-jung.gitbook.io/docs/q-and-a/spring/controlleradvice-exceptionhandler