# 6장. 유효성 검사, 예외 처리, 배포

*Validation, Exceptions*

Spring Boot

# 유효성 검사(Validation)

- **유효성 검사**

  Bean Validation을 이용한 유효성 검사는 웹 애플리케이션을 구성하는 특정 도메인 클래스의 프로퍼티(멤버 변수) 즉 필드에 대한 유효성 검사의 제약사항 애너테이션을 선언하여 해당 멤버 변수의 값이 올바른지 검사하는 것이다.

  1. @제약사항 애너테이션을 선언하다.
  2. @Valid를 이용한 유효성 검사를 실행 한다.
  3. 뷰 페이지에 오류 메시지를 출력한다.

  - **Validation 의존성 추가**

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

# 유효성 검사(Validation)

- **유효성 검사**
  - **제약사항 애너테이션 유형**

| 메서드 | 설명 |
|---|---|
| @NotEmpty | 프로퍼티 값이 null, 빈 문자열이 아닌지 모두 검사 |
| @Email | 이메일 형식인지 검사 |
| @Size | 프로퍼티 값이 가질 수 있는 최대, 최소 길이 검사 |
| @Max | 프로퍼티 값이 가질 수 있는 최대 길이 검사 |
| @Min | 프로퍼티 값이 가질 수 있는 최소 길이 검사 |

# 유효성 검사(Validation)

- **유효성 검사**
  - **제약사항 애너테이션 기본 메시지**

| 메서드 | 설명 |
|---|---|
| @NotEmpty | 비어 있을 수 없습니다. |
| @Email | 올바른 형식의 이메일 주소여야 합니다. |
| @Size(min=, max=) | 크기가(min)에서 (max) 사이여야 합니다. |
| @Max(value=) | (value) 이하 여야 합니다. |
| @Min(value=) | (value) 이상 이어야 합니다. |

# 유효성 검사(Validation)

**유효성 검사**

품명: 비어 있을 수 없습니다

가격: 0

확인 취소

**유효성 검사**

품명: 갤럭시

가격: -1000 0 이상이어야 합니다

확인 취소

# 유효성 검사(Validation)

- **Product DTO**

```
@Data
public class Product {

    @NotEmpty
    @Size(min=4, max=10, message="4자~10자 이내로 입력해 주세요")
    private String name;

    @Min(value=0)
    private int price;
}
```

*기본 메시지 대체*

# 유효성 검사(Validation)

- **ValidController**

```
@RequestMapping("/valid01")
@Controller
public class ValidController {

    @GetMapping
    public String showForm(@ModelAttribute Product product) {
        return "validation/viewPage01";
    }

    @PostMapping
    public String submit(@Valid @ModelAttribute Product product,
            BindingResult bindingResult) {
        if(bindingResult.hasErrors())
            return "validation/viewPage01";

        return "validation/viewPage01_result";
    }
}
```

# 유효성 검사(Validation)

- **viewPage01**

```
<h3>유효성 검사</h3>
<form action="/valid01" method="post" th:object="${product}">
    <p>품명: <input type="text" th:field="*{name}">
             <span th:errors="*{name}"></span>
    <p>가격: <input type="text" th:field="*{price}">
             <span th:errors="*{price}"></span>
    <p>
        <input type="submit" value="확인">
        <input type="reset" value="취소">
    </p>
</form>
```

# 유효성 검사(Validation)

- **viewPage01_result**

```
<h3>유효성 검사</h3>
<!--
<p th:text="'품명: ' + ${product.name}">
<p th:text="'가격: ' + ${product.price}">
-->

<p>품명 : [[${product.name}]]
<p>가격 : [[${product.price}]]
```

# 예외 처리(Exception Handling)

- **예외 처리**

  **예외의 개념**

  - 사용자가 시스템을 사용하다 보면 웹 페이지 URL 경로를 잘못 입력하거나 파라미터를 잘못 전달하여 문제가 발생하는 경우가 있다.

  - 또는 서버에서 실행되는 프로그램에서 생각하지 못했던 문제가 발생할 수 도 있다.

  - 이렇게 사용자의 부주의나 코드 자체의 오류로 인해 문제가 발생했을때, 발생된 문제를 적절하게 처리하지 않으면 사용자 브라우저에서 에러 메시지가 출력된다.

# 예외 처리(Exception Handling)

- **예외 처리**

  **스프링부트에서 예외를 처리하는 방법**

  1) **@ControllerAdvice** 어노테이션을 이용하여 모든 컨트롤러에서 발생하는 예외를 일괄적으로 처리한다. (전역 예외 처리)

  2) **@ExceptionHandler** 어노테이션을 이용하여 각 컨트롤러마다 발생하는 예외를 개별적으로 처리한다. (로컬 예외 처리)

# 예외 처리(Exception Handling)

- **사용자 정의 예외 만들기**

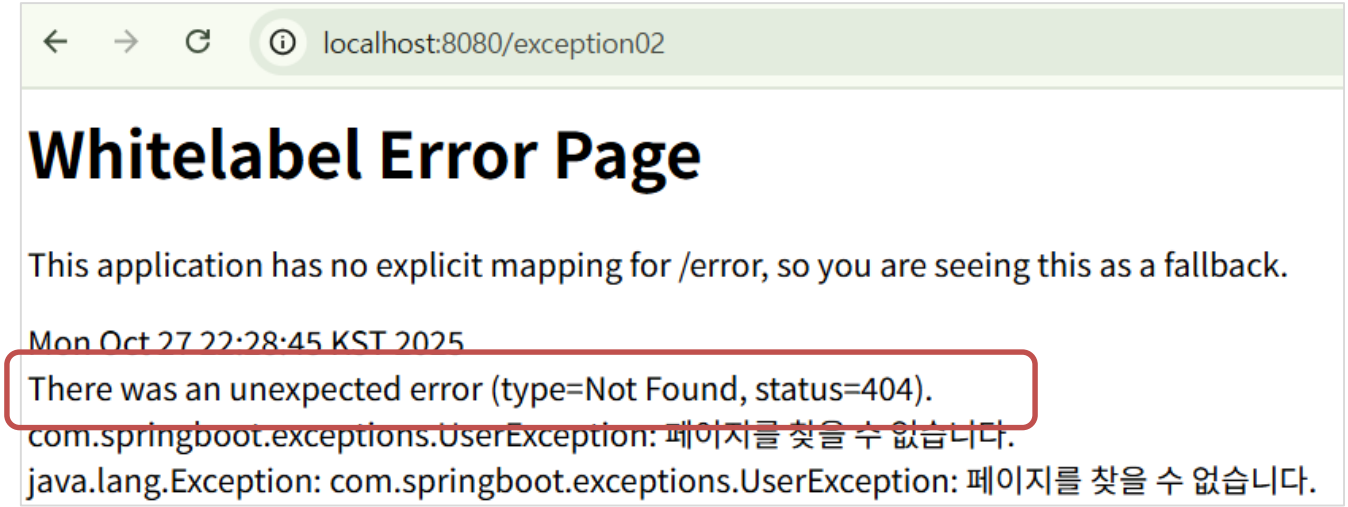
RuntimeException을 상속받아 만든다.

```
//사용자 정의 예외는 RuntimeException을 상속 받아야 함
@ResponseStatus(value=HttpStatus.NOT_FOUND)  //404 오류
public class UserException extends RuntimeException{

    public UserException(String message) {
        super(message);
    }
}
```

# 예외 처리(Exception Handling)

- 404 오류 – 페이지를 찾을 수 없을 때




# 예외 처리(Exception Handling)

- 404 오류 – 페이지를 찾을 수 없을 때

```
@Controller
public class Exception01Controller {

    @GetMapping("/exception01")
    public void method() throws Exception {
        throw new Exception(new UserException("페이지를 찾을 수 없습니다."));
    }

}
```

# 예외 처리(Exception Handling)

- @ExceptionHandler 사용한 예외 처리

```java
@Controller
public class Exception02Controller {

    @GetMapping("/exception02")
    public void method() {
        throw new UserException("페이지를 찾을 수 없습니다.");
    }

    @ExceptionHandler(UserException.class)  //로컬 예외 처리(개별적인 예외)
    public String handleException(UserException ex, Model model) {
        model.addAttribute("data1", ex.getMessage());
        model.addAttribute("data2", ex);
        return "exception/viewPage01";
    }
}
```

# 예외 처리(Exception Handling)

- **@ExceptionHandler 사용**

**예외 처리**

오류 메시지: 페이지를 찾을 수 없습니다.

예외 발생: com.springboot.exceptions.UserException: 페이지를 찾을 수 없습니다.

현재 페이지는 viewPage01.html입니다

viewPage01.html

```
<h3>예외 처리</h3>
<p>오류 메시지: [[${data1}]]
<p>예외 발생: [[${data2}]]
<p>현재 페이지는 viewPage01.html입니다</p>
```

# 예외 처리(Exception Handling)

- **@ControllerAdvice 사용한 예외 처리**

```java
//모든 컨트롤러에서 발생하는 예외를 일괄 처리함
@ControllerAdvice  //전역 예외 처리
public class GlobalException {

    @ExceptionHandler
    private String handleErrorMethod(Exception ex, Model model) {
        model.addAttribute("data1", "GlobalException 메시지입니다.");
        model.addAttribute("data2", ex);
        return "exception/viewPage02";
    }
}
```

# 예외 처리(Exception Handling)

- @ControllerAdvice를 사용한 예외 처리

```java
@Controller
public class Exception03Controller {

    @GetMapping("/exception03")
    public void method() {
        throw new UserException("UserException 메시지입니다.");
    }
}
```

# 예외 처리(Exception Handling)

- **@ControllerAdvice 사용**

**예외 처리**

오류 메시지: GlobalException 메시지입니다.

예외 발생: com.springboot.exceptions.UserException: UserException 메시지입니다.

현재 페이지는 viewPage02.html입니다

viewPage02.html

```
<h3>예외 처리</h3>
<p>오류 메시지: [[${data1}]]
<p>예외 발생: [[${data2}]]
<p>현재 페이지는 viewPage02.html입니다</p>
```

# Friends 프로젝트

- 사용자 정의 예외 만들기 - UserException

```
//사용자 정의 예외 처리
public class UserException extends RuntimeException{

    public UserException(String message) {
        super(message);

    }

}
```

# Friends 프로젝트

- 전역으로 예외 다루기 - @ExceptionHandler

```
@ControllerAdvice
//@RestController
@Controller
public class UserExceptionHandler {

    @ExceptionHandler(value = Exception.class)
    public String GlovalExceptionHandle(Exception e,
            Model model) {
        //return "<h2>" + e.getMessage() + "</h2>"; //RestController 경우
        model.addAttribute("errorMsg", e.getMessage());
        return "error/errorPage";
    }
}
```

# Friends 프로젝트

- 에러 페이지 – errorPage.html

```
<section id="container">
    <h2 th:text="${errorMsg}"></h2>

    <a th:href="@{/members}">회원 목록</a> |
    <a th:href="@{/boards/pages}">게시글 목록</a>
</section>
```

# Friends 프로젝트

- 회원이 존재하지 않을 때 예외 처리 - MemberController

```
@GetMapping("/{id}") //회원 정보(상세)
public String getMember(@PathVariable Long id,
                        Model model) {
    Member member = service.findById(id);
    model.addAttribute("member", member);
    return "member/info";
}
```

# Friends 프로젝트

- 회원이 존재하지 않을 때 예외 처리 - MemberService

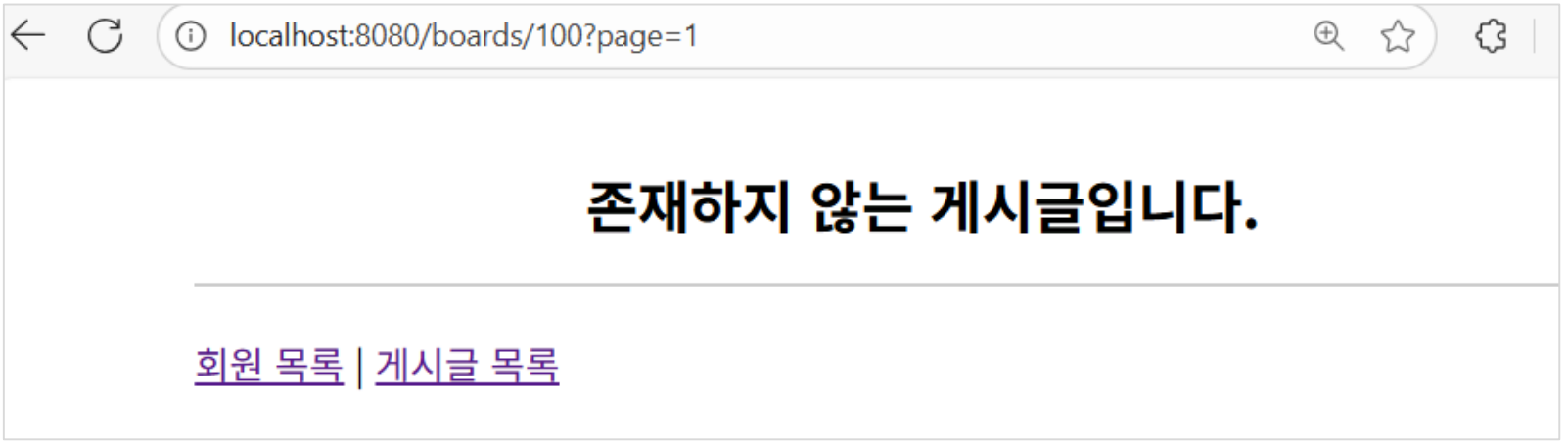
```
//회원 정보
public Member findById(Long id) {
    //예외 처리
    Member findMember =
            repository.findById(id)
                .orElseThrow(() -> {
                    throw new UserException("존재하지 않는 회원입니다.");
                });
    return findMember;
}
```

# Friends 프로젝트

- 100번 회원이 존재하지 않을 때 예외 처리


# Friends 프로젝트

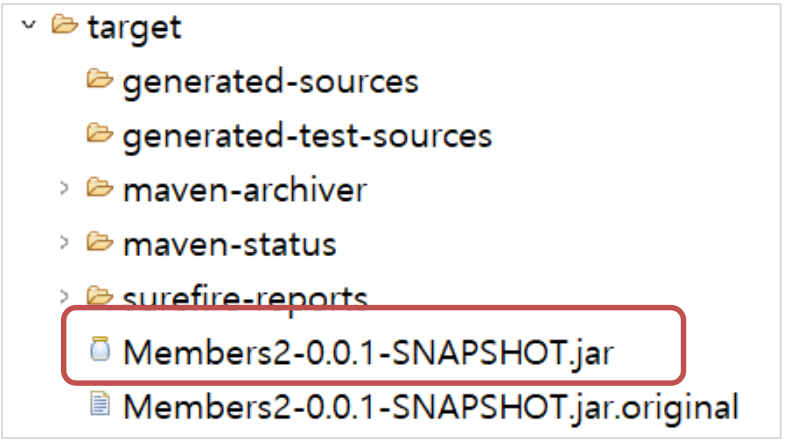
- 100번 게시글이 존재하지 않을 때 예외 처리




# 배포(Deploy)

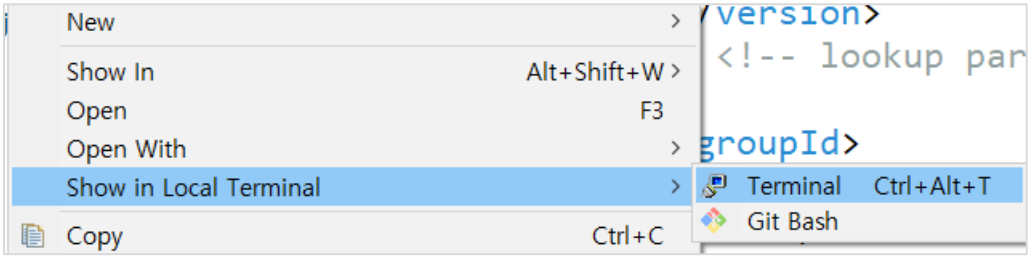
- **배포하기**
  - 프로젝트 > 우클릭 > Run As > **Maven install**

    target > 프로젝트이름.jar 파일 생성




# 배포(Deploy)

- **jar 파일 실행**
  - **프로젝트이름.jar > 우측메뉴 > Show in Local Terminal > Terminal**




  - **java -jar 프로젝트이름.jar** 명령 입력

```
[INFO] ------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------
[INFO] Total time:  31.289 s
[INFO] Finished at: 2025-10-27T18:58:45+09:00
[INFO] ------------------------------------------
```

# 배포(Deploy)

- jar 파일 실행
  - 명령 프롬프트(CMD)에서 실행하기

```
C:\Users\LG>cd c:/java_jar

c:\java_jar>java -jar members.jar

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.4.10)

2025-10-27T19:03:35.894+09:00  INFO 10980 --- [Members] [
ation        : Starting MembersApplication v0.0.1-SNAPSHOT
ar\members.jar started by LG in c:\java_jar)
```

# 배포(Deploy)

- 브라우저에서 실행