Spring MVC

28 예외처리(2) - 이론

예외 처리(2) - 이론

1. @ExceptionHandler와 @ControllerAdvice

예외 처리를 위한 메서드를 작성하고 @ExceptionHandler를 붙인다.

```
@Controller
public class ExceptionController {
    @ExceptionHandler({NullPointerException.class, FileNotFoundException.class})
    public String catcher2(Exception ex, Model m) {
        m.addAttribute("ex", ex);
        return "error";
    }

    @ExceptionHandler(Exception.class)
    public String catcher(Exception ex, Model m) {
        System.out.println("catcher() in ExceptionController");
        System.out.println("m="+m);
        m.addAttribute("ex", ex);

        return "error";
    }

    @RequestMapping("/ex")
    public String main(Model m) throws Exception {
        m.addAttribute("msg", "message from ExceptionController.main()");
        throw new Exception("예외가 발생했습니다.");
    }
}
```

예외 처리(2) - 이론

1. @ExceptionHandler와 @ControllerAdvice

@ControllerAdvice로 전역 예외 처리 클래스 작성 가능(패키지 지정 가능)
예외 처리 메서드가 중복된 경우, 컨트롤러 내의 예외 처리 메서드가 우선

```java
//@ControllerAdvice("com.fastcampus.ch2") // 지정된 패키지에만 적용
@ControllerAdvice // 모든 패키지에 적용
public class GlobalCatcher {
    @ExceptionHandler({NullPointerException.class, ClassCastException.class})
    public String catcher2(Exception ex) {
        return "error";
    }

    @ExceptionHandle
    public String ca
        return "erro
    }
}
```

```java
@Controller
public class ExceptionController {
    @ExceptionHandler({NullPointerException.class, FileNotFoundException.cl
    public String catcher2(Exception ex, Model m) {
        m.addAttribute("ex", ex);
        return "error";
    }

    @RequestMapping("/ex2")
    public String main2() throws Exception {
        throw new NullPointerException("예외가 발생했습니다.");
    }
}
```

예외 처리(2) - 이론

2. @ResponseStatus

응답 메시지의 상태 코드를 변경할 때 사용

```
@ResponseStatus(HttpStatus.METHOD_NOT_ALLOWED) //
@ExceptionHandler({NullPointerException.class, Cl
public String catcher2(Exception ex, Model m) {
    m.addAttribute("ex", ex);
    return "error";
}
```

```
@ResponseStatus(HttpStatus.BAD_REQUEST) // 400 Bad Request.
class MyException extends RuntimeException {
    MyException(String msg){
        super(msg);
    }

    MyException() {
        this("");
    }
}
```

HTTP response status code	description
401	Unauthorized
403	Forbidden
404	Not Found
405	Method Not Allowed
…	…
500	Internal Server Error
501	Not Implmented
503	Service Unavailable
…	…

```
<error-page>
    <error-code>400</error-code>
    <location>/error400.jsp</location>
</error-page>
<error-page>
    <error-code>500</error-code>
    <location>/error500.jsp</loca
</error-page>
```

예외 처리(2) - 이론

3. <error-page> - web.xml

상태 코드별 뷰 맵핑

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee https://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/root-context.xml</param-value>
    </context-param>

    ...

    <error-page>
        <error-code>400</error-code>
        <location>/error400.jsp</location>
    </error-page>
    <error-page>
        <error-code>500</error-code>
        <location>/error500.jsp</location>
    </error-page>
</web-app>
```

예외 처리(2) - 이론

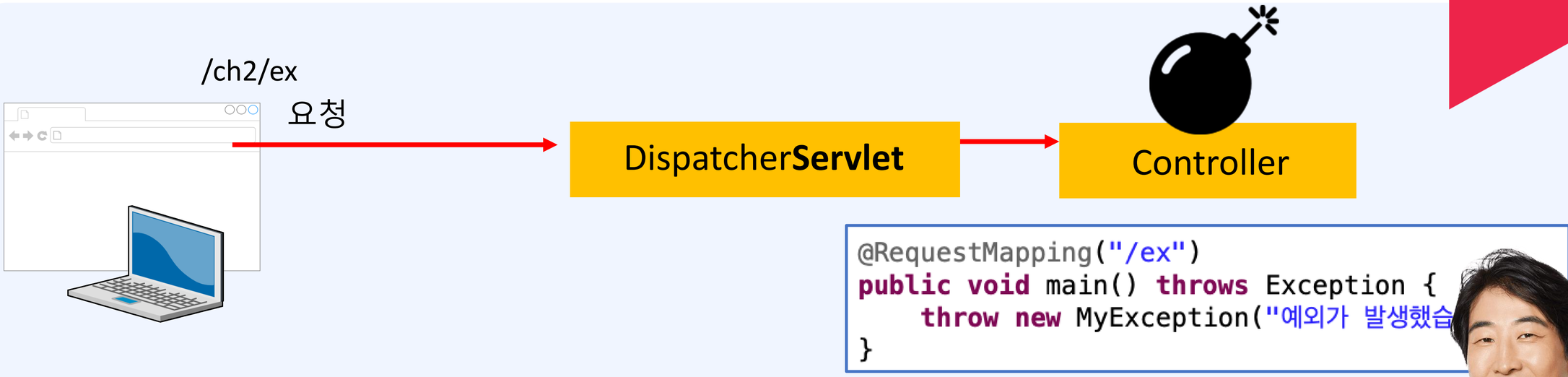
4. SimpleMappingExceptionResolver

예외 종류별 뷰 맵핑에 사용. servlet-context.xml에 등록

```
<beans:bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
        <beans:property name="defaultErrorView" value="error"/>
        <beans:property name="exceptionMappings">
                <beans:props>
                        <beans:prop key="com.fastcampus.ch2.MyException">error400</beans:
                </beans:props>
        </beans:property>
        <beans:property name="statusCodes">
                <beans:props>
                        <beans:prop key="error400">404</beans:prop>
                </beans:props>
        </beans:property>
</beans:bean>
```

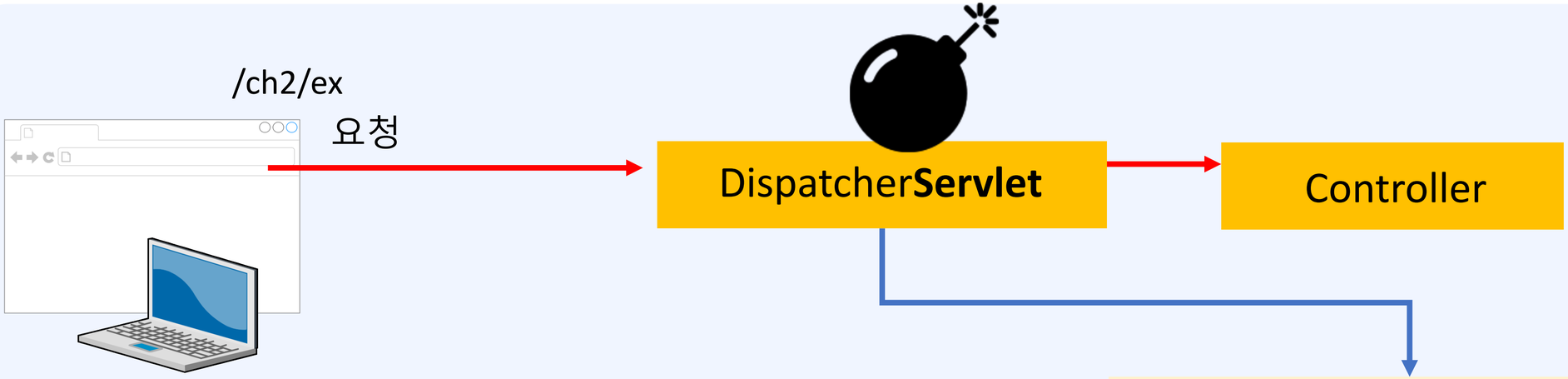
예외 처리(2) - 이론

5. ExceptionResolver



예외 처리(2) - 이론

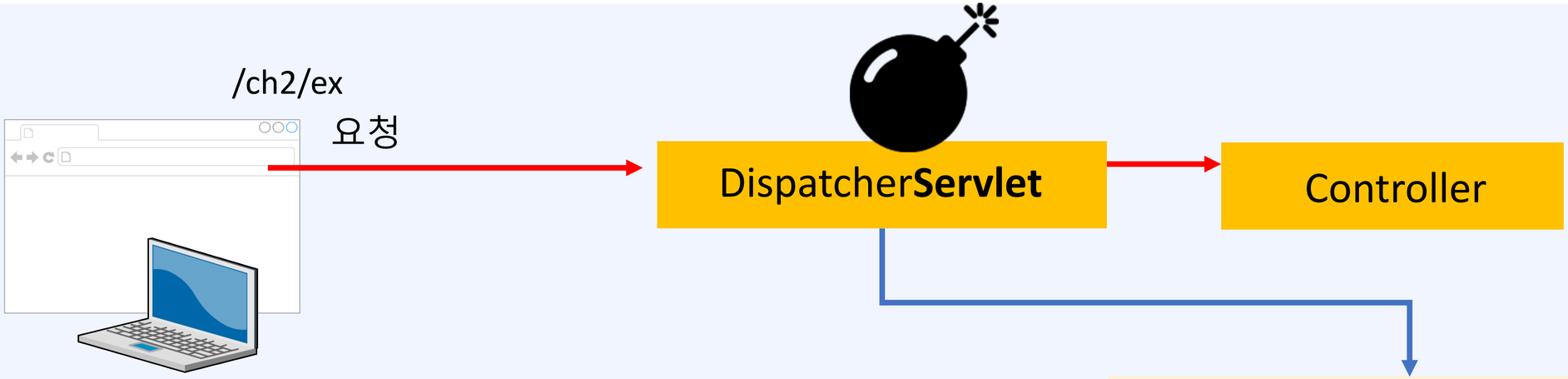
5. ExceptionResolver




```
@ResponseStatus(HttpStatus.METHOD_NOT_ALLOWED) // 405 Method Not Allowed.
@ExceptionHandler({NullPointerException.class, ClassCastException.class})
public String catcher2(Exception ex, Model m) {
    m.addAttribute("ex", ex);
    return "error";
}
```

예외 처리(2) - 이론

5. ExceptionResolver




```java
@ResponseStatus(HttpStatus.BAD_REQUEST) // 400 Bad Request.
class MyException extends RuntimeException {
    MyException(String msg){
        super(msg);
    }

    MyException() {
        this("");
    }
}
```

```xml
<error-page>
    <error-code>400</error-code>
    <location>/error400.jsp</location>
</error-page>
<error-page>
    <error-code>500</error-code>
    <location>/error500.jsp</location>
</error-page>
```

예외 처리(2) - 이론
6. 스프링에서의 예외 처리

- 컨트롤러 메서드 내에서 try-catch로 처리

```
@RequestMapping("/ex")
public String main(Model m) throws Exception {
    m.addAttribute("msg", "message from ExceptionController.main()");
    try {
        throw new Exception("예외가 발생했습니다.");
    } catch(Exception e) {
        return "error";
    }
}
```

예외 처리(2) - 이론
6. 스프링에서의 예외 처리

- 컨트롤러 메서드 내에서 try-catch로 처리

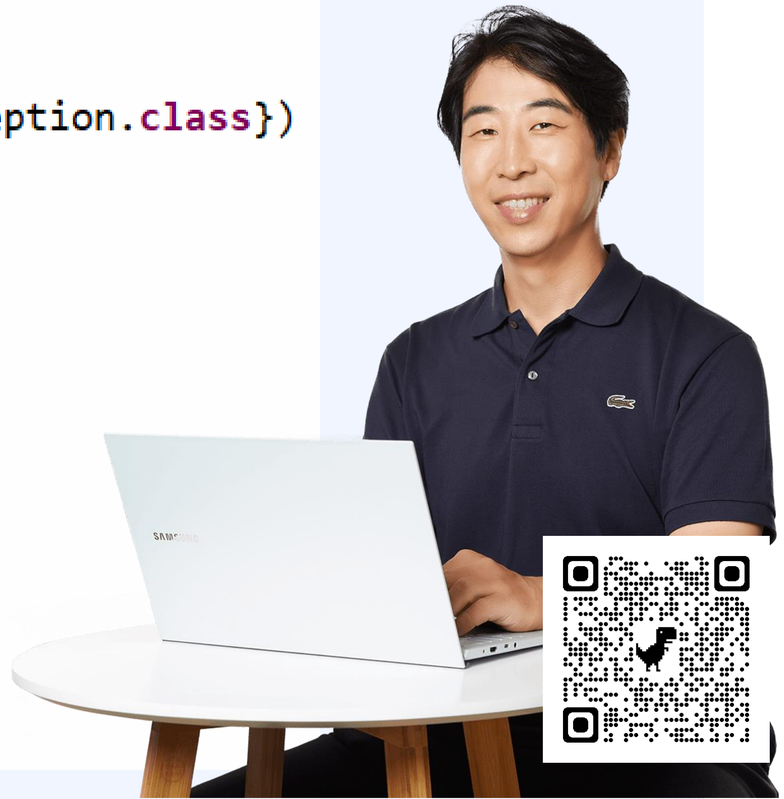
- 컨트롤러에 @ExceptionHandler메서드가 처리

```
@Controller
public class ExceptionController {

    @ExceptionHandler({NullPointerException.class, FileNotFoundException.class})
    public String catcher2(Exception ex, Model m) {
        m.addAttribute("ex", ex);
        return "error";
    }

    @ExceptionHandler(Exception.class)
    public String catcher(Exception ex, Model m) {
        System.out.println("catcher() in ExceptionController");
        System.out.println("m="+m);
        m.addAttribute("ex", ex);

        return "error";
    }
}
```



예외 처리(2) - 이론
6. 스프링에서의 예외 처리

- 컨트롤러 메서드 내에서 try-catch로 처리
- 컨트롤러에 @ExceptionHandler메서드가 처리
- @ControllerAdvice클래스의 @ExceptionHandler메서드가 처리

```
@ControllerAdvice("com.fastcampus.ch3") // 지정된 패키지에서 발생한 예외만 처리
//@ControllerAdvice // 모든 패키지에 적용
public class GlobalCatcher {
    @ExceptionHandler({NullPointerException.class, FileNotFoundException.class})
    public String catcher2(Exception ex, Model m) {
        m.addAttribute("ex", ex);
        return "error";
    }

    @ExceptionHandler(Exception.class)
    public String catcher(Exception ex, Model m) {
        m.addAttribute("ex", ex);

        return "error";
    }
}
```

예외 처리(2) - 이론
6. 스프링에서의 예외 처리

- 컨트롤러 메서드 내에서 try-catch로 처리
- 컨트롤러에 @ExceptionHandler메서드가 처리
- @ControllerAdvice클래스의 @ExceptionHandler메서드가 처리
- 예외 종류별로 뷰 지정 – SimpleMappingExceptionResolver

```
<beans:bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
        <beans:property name="defaultErrorView" value="error"/>
        <beans:property name="exceptionMappings">
                <beans:props>
                        <beans:prop key="com.fastcampus.ch2.MyException">error400</beans:prop>
                </beans:props>
        </beans:property>
        <beans:property name="statusCodes">
                <beans:props>
                        <beans:prop key="error400">404</beans:prop>
                </beans:props>
        </beans:property>
</beans:bean>
```

예외 처리(2) - 이론
6. 스프링에서의 예외 처리

- 컨트롤러 메서드 내에서 try-catch로 처리
- 컨트롤러에 @ExceptionHandler메서드가 처리
- @ControllerAdvice클래스의 @ExceptionHandler메서드가 처리
- 예외 종류별로 뷰 지정 – SimpleMappingExceptionResolver
- 응답 상태 코드별로 뷰 지정 - <error-page>

```
@ResponseStatus(HttpStatus.BAD_REQUEST) // 400 Bad Request.
class MyException extends RuntimeException {
    MyException(String msg){
        super(msg);
    }

    MyException() {
        this("");
    }
}
```

```
<error-page>
    <error-code>400</error-code>
    <location>/error400.jsp</location>
</error-page>
<error-page>
    <error-code>500</error-code>
    <location>/error500.jsp</location>
</error-page>
```