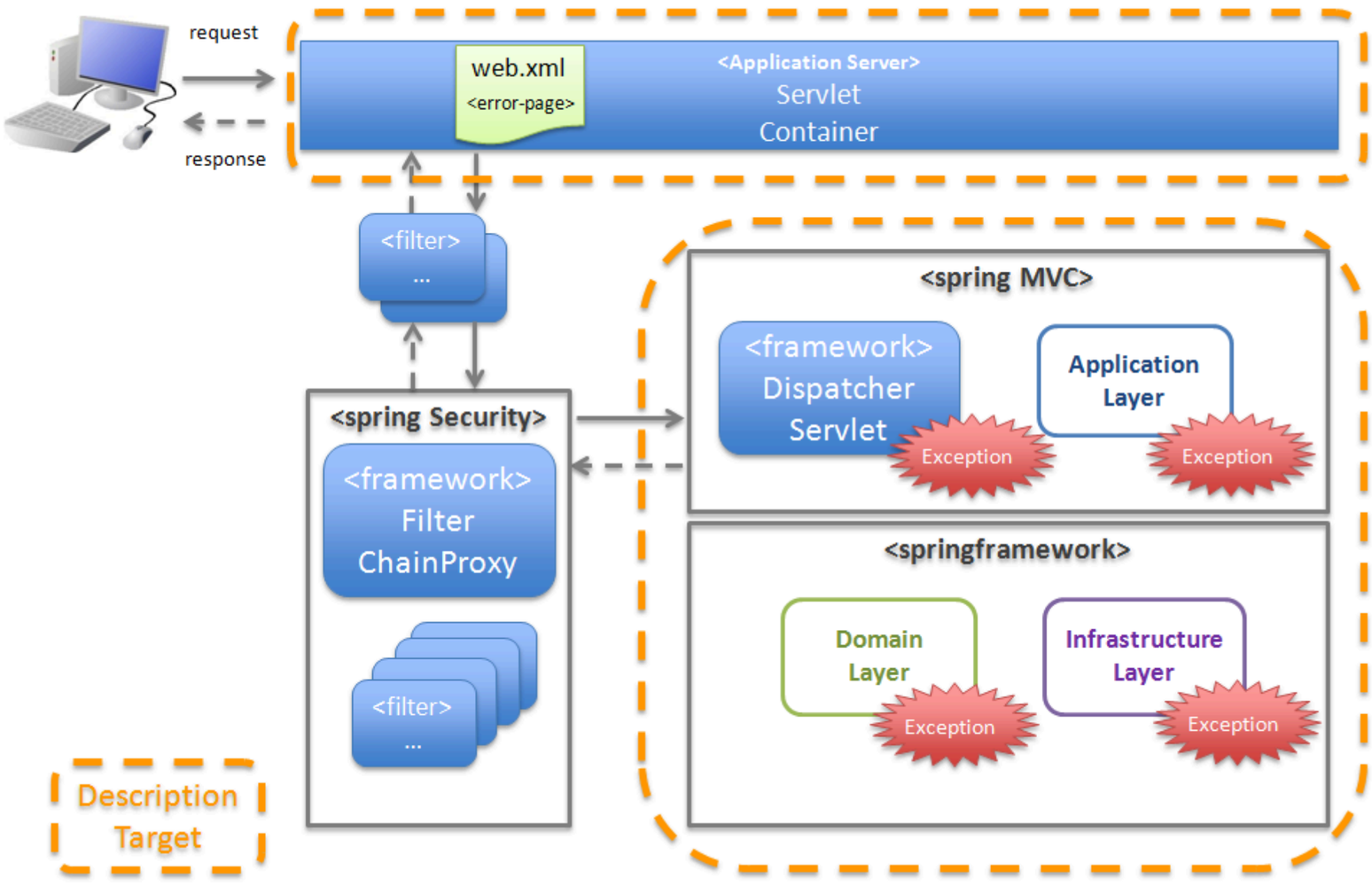
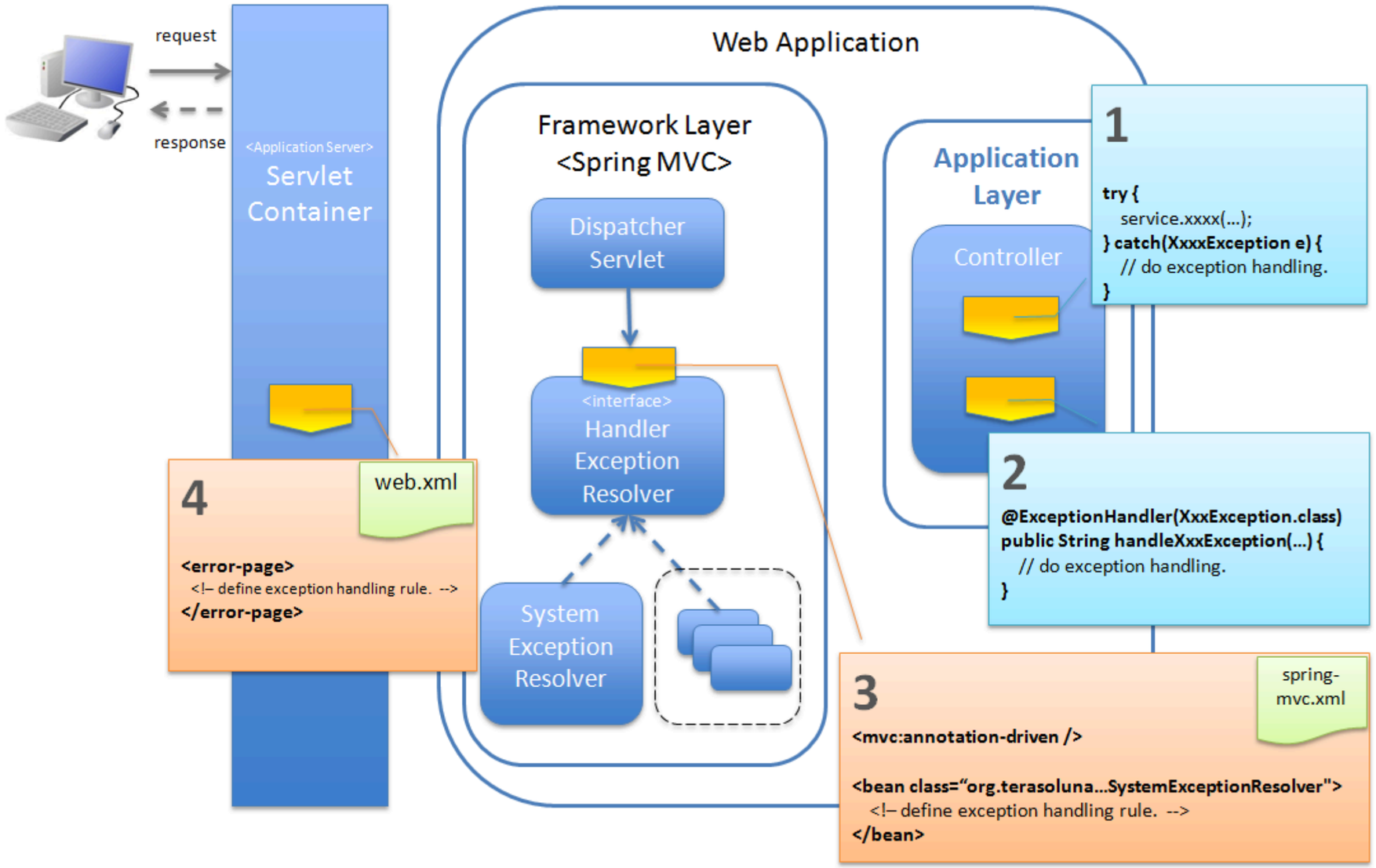
예외처리

예외(exception)은 크게 서블릿 컨테이너에서 발생하는 예외와 스프링 컨테이너에서 발생하는 예외로 나눌 수 있습니다.



스프링MCV에서 예외의 대부분은 스프링 컨테이너 내부에서 발생합니다. 스프링 컨테이너에서 발생하는 예외는 HandlerExceptionResolver 를 통해 처리되며 다양한 방식으로 접근할 수 있습니다.



웹 어플리케이션 레벨 예외 처리

클라이언트의 요청을 DispatcherServlet에서 처리하기 전에 예외가 발생하면 HandlerExceptionResolver가 예외처리를 하지 못합니다. 이 경우 Web Application 레벨에서 예외를 처리해야합니다.

필터를 통한 예외 처리

우선 필터를 통해서 예외처리를 할 수 있습니다. HandlerExceptionResolver를 주입받아 필터에서 발생하는 예외를 HandlerExceptionResolver로 보내서 처리하면 됩니다. 스프링 시큐리티에서 예외처리를 하는 방법입니다. 필터에 대하여 알아야하기 때문에 생략하겠습니다.

web.xml 에서 에러페이지 설정

다음으로 web.xml 에 에러페이지를 설정하여 예외처리를 할 수 있습니다.

우선 에러페이지 템플릿을 작성하도록 합시다.

/src/main/webapp/WEB-INF/views/error/error.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"  %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"  %>
<html>
<head>
<meta charset="UTF-8">
<title>오류발생</title>
</head>
<body>
    <h1>오류 발생</h1>
</body>
</html>
```

이제 web.xml 파일에 에러페이지를 설정해주면 됩니다. 예외타입 또는 HTTP Status 코드별로 에러페이지를 설정할 수 있습니다.

먼저 HTTP Status 코드에 따른 에러페이지를 설정해보겠습니다. 아래 코드와 같이 `error-page` 태그를 추가합니다.

src/main/webapp/WEB-INF/web.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>

<!-- Creates the Spring Container shared by all Servlets and Filters -->
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

<!-- Processes application requests -->
<servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
    </init-param>
```

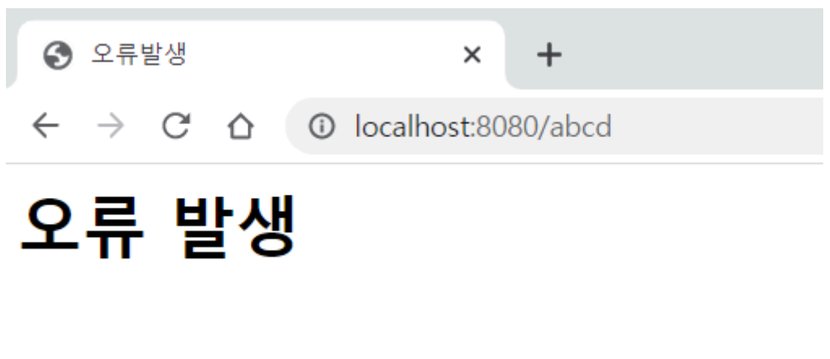
```
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>appServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <filter>
        <filter-name>encodingFilter</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
    </filter>
    <filter>
        <filter-name>httpMethodFilter</filter-name>
        <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>encodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <filter-mapping>
        <filter-name>httpMethodFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

```
    <error-page>
        <error-code>404</error-code>
        <location>/WEB-INF/views/error/error.jsp</location>
    </error-page>
</web-app>
```

매핑되지 않는 주소(localhost:8080/abcd)로 접속하면 404 에러가 발생하므로, 위에서 설정한 에러 페이지를 출력하게 됩니다.



이번에는 예외타입을 통해 에러 페이지를 설정해보겠습니다. 위에 error-page 하위 태그인 error-code를 exception-type 으로 변경하세요.

src/main/webapp/WEB-INF/web.xml

```
...
    <error-page>
        <exception-type>kro.rubisco.config.TestException</exception-type>
```

```
        <location>/WEB-INF/views/error/error.jsp</location>
    </error-page>
...
```

TestException이 던져지면 설정한 에러 페이지를 출력하게 됩니다. 위에 작성한 예외 타입은 임시로 작성한 예외타입입니다.

/kro/rubisco/config/TestException.java

```
package kro.rubisco.config;

import lombok.NoArgsConstructor;

@NoArgsConstructor
public class TestException extends Exception {
    public TestException(String msg) {
        super(msg);
    }
}
```

BoardController의 getBoardListView 메소드를 아래와 같이 수정하고 localhost:8080/board 로 접속하면 동일한 에러 페이지가 출력됩니다.

/kro/rubisco/controller/BoardController.java

```
...
    @GetMapping()
    public Model getBoardListView(
        PageDTO<BoardDTO> boardPage,
```

```
        SearchDTO search,
        Model model
    ) throws Exception {
        throw new TestException();
    }
...
```

DispatcherServlet의 예외 처리

처음 설명한 것과 같이 스프링 컨테이너의 경우 DispatcherServlet에서 발생하는 예외는 `HandlerExceptionResolver` 가 처리합니다.

스프링에서 기본적으로 `ExceptionHandlerExceptionResolver` , `ResponseStatusExceptionResolver` , `DefaultHandlerExceptionResolver` 를 제공하고 있으며, 순서대로 우선순위를 가집니다.

ExceptionHandlerExceptionResolver

해당 resolver는 컨트롤러 레벨에서 `@ExceptionHandler` 어노테이션이 붙은 메소드를 호출하여 예외처리를 합니다.

위에서 web.xml에 작성한 error-page 태그를 지우고 BoardController에 다음 메소드를 작성해보세요.

/kro/rubisco/controller/BoardController.java

```
...
    @ExceptionHandler(TestException.class)
    public ModelAndView testExceptionHandler(Exception e) {
        ModelAndView mv = new ModelAndView();
        mv.setStatus(HttpStatus.BAD_REQUEST);
        mv.setViewName("/error/error");
        return mv;
    }
...
```

BoardController에 매핑되는 request에서 `TestException` 타입의 예외가 발생하면 `testExceptionHandler` 메소드가 호출됩니다. 해당 메소드는 상태코드를 400으로 설정하여 위에서 작성한 에러 페이지를 response 합니다.

만약 컨트롤러 레벨이 아니라 전역 레벨에서 예외처리를 하고자 한다면 `@ControllerAdvice` 어노테이션이 붙은 클래스를 작성하면 됩니다.

/kro/rubisco/controller/ExceptionHandlingController.java

```
package kro.rubisco.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.servlet.ModelAndView;
```

```
@ControllerAdvice
public class ExceptionHandlingController {


    @ResponseStatus(HttpStatus.BAD_REQUEST)
    @ExceptionHandler(Exception.class)
    protected ModelAndView handleBadRequest(HttpServletRequest req, Exception e) {
        ModelAndView mv = new ModelAndView();
        mv.addObject("exception", e);
        mv.setViewName("error/error");
        return mv;
    }
}
```

ResponseStatusExceptionResolver

해당 resolver는 예외에 따라 HTTP State Code를 지정해줍니다. 위에서 작성한 ExceptionHandlingController의 handleBadRequest 메소드에 보면 `@ResponseStatus` 어노테이션을 통해 상태코드를 설정해 주었습니다.

DefaultHandlerExceptionResolver

해당 resolver는 스프링 컨테이너 내부에서 발생되는 기본적인 예외들을 처리합니다. 즉, 예외처리를 따로 해주지 않아도 제공되는 해당 resolver에 의하여 예외처리가 됩니다. 예를 들어 파라미터 타입이 일치하지 않은 경우 `TypeMismatchException` 이 발생하는데, DispatcherServlet에서 발생한 오류이므로 상태코드

받게 됩니다.