7장. 예외 처리 및 보안(시큐리티)

Security & Exception

Exception

예외 처리(Exception Handling)

- 예외 처리는 프로그램이 처리되는 동안 특정한 문제가 발생했을때 처리를 중단하고 다른 처리를 하는 것으로 오류 처리라고도 한다.
- 웹 애플리케이션 실행 도중에 발생할 수 있는 오류에 대비한 예외 처리코드를 작성하여 비정상적인 종료를 막을 수 있다.

예외 처리 방법	설 명
page 디렉티브 태그를 이용한 예외 처리	errorPage와 isErrorPage 속성을 이용한다.
try/catch/finally를 이용한 예외 처리	자바 언어의 예외 처리 구문을 이용한다.
web.xml 파일을 이용한 예외 처리	<error-code> 또는 <exception-type>요소를 이용한다.

Exception

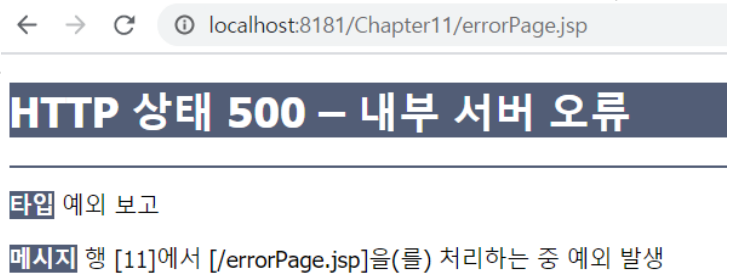
page 디렉티브 태그를 이용한 예외 처리

- **erropPage 속성으로 오류 페이지 호출하기**

```
<%@ page errorPage = "오류 페이지 URL" %>
```

errorPage.jsp

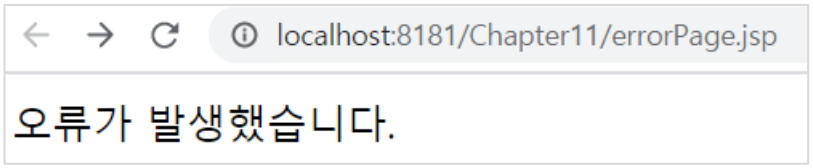
```jsp
<%@ page errorPage="errorPrint.jsp" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Exception</title>
</head>
<body>
    name 파라미터 : <%=request.getParameter("name").toString() %>
</body>
</html>
```



Exception

page 디렉티브 태그를 이용한 예외 처리

- **erropPage 속성으로 오류 페이지 호출하기**




```
<title>ErrorPrint 페이지</title>
</head>
<body>
    오류가 발생하였습니다.
</body>
</html>
```

Exception

page 디렉티브 태그를 이용한 예외 처리

- **erropPage 속성으로 오류 페이지 호출하기**

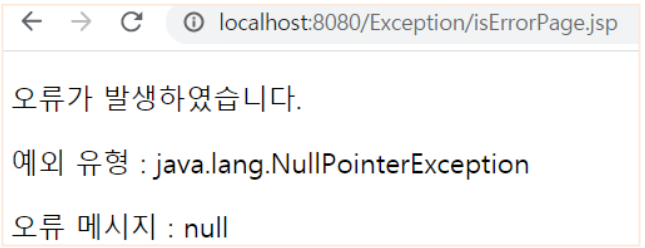
```
<%@ page isErrorPage = "true" %>
```

메소드	형식	설 명
getMessage()	String	오류 이벤트와 함께 들어오는 메시지를 출력
toString()	String	오류 이벤트의 toString()을 호출하여 간단한 오류 메시지를 확인
printStackTrace()	String	오류 메시지의 발생 근원지를 찾아 단계별로 오류를 출력

Exception

page 디렉티브 태그를 이용한 예외 처리

- erropPage 와 isErrorPage 속성으로 오류 페이지 호출하기



errorPage2.jsp

```jsp
<%@ page errorPage="isErrorPrint.jsp" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Exception</title>
</head>
<body>
    name 파라미터 : <%=request.getParameter("name").toUpperCase() %>
    <!-- 에러처리 안할 경우 비정상적인 종료됨.  -->
</body>
</html>
```

Exception

page 디렉티브 태그를 이용한 예외 처리

- erropPage 와 isErrorPage 속성으로 오류 페이지 호출하기

```jsp
<%@ page isErrorPage="true" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>ErrorPrint 페이지</title>
</head>
<body>
    <p> 오류가 발생하였습니다.
    <p> 예외 유형 : <%=exception.getClass().getName() %>
    <p> 오류 메시지 : <%=exception.getMessage() %>
</body>
</html>
```

exception 내장 객체

Exception

web.xml 파일을 이용한 예외 처리

- **오류 코드로 오류 페이지 호출하기**

```
<error-page>
    <error-code>…</ error-code>
    <location>…</ location>
</ error-page >
```

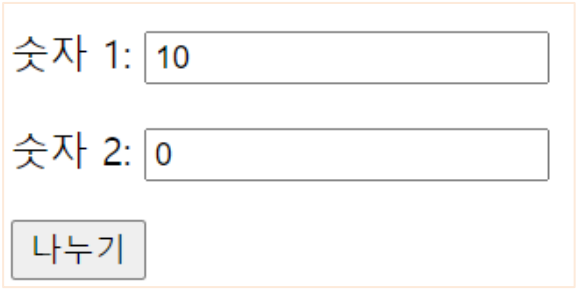
요 소	설 명
<error-code>	오류 코드 설정
<exception-type>	자바 예외 유형의 정규화된 클래스 이름 설정
<location>	오류 페이지의 URL을 설정

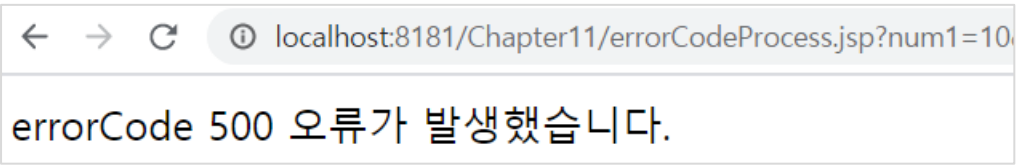
Exception

web.xml 파일을 이용한 예외 처리

- **오류 코드로 오류 페이지 호출하기**

errorCode.jsp






```
<form action="errorCodeProcess.jsp" method="get">
    <p>숫자 1 : <input type="text" name="num1">
    <p>숫자 2 : <input type="text" name="num2">
    <p><input type="submit" value="나누기">
</form>
```

Exception

web.xml 파일을 이용한 예외 처리

- 수를 0으로 나누었을 때 예외 처리하기

```jsp
<%
    int num1 = Integer.parseInt(request.getParameter("num1"));
    int num2 = Integer.parseInt(request.getParameter("num2"));
    int div = num1 / num2;

    out.println(num1 + " / " + num2 + " = " + div);
%>
```

web.xml

```xml
<error-page>
  <error-code>500</error-code>
  <location>/errorCodePrint.jsp</location>
</error-page>
```

errorCodePrint.jsp

```jsp
<body>
    errorCode 500 오류가 발생했습니다.
</body>
```

Exception

try-catch-finally 이용한 예외 처리

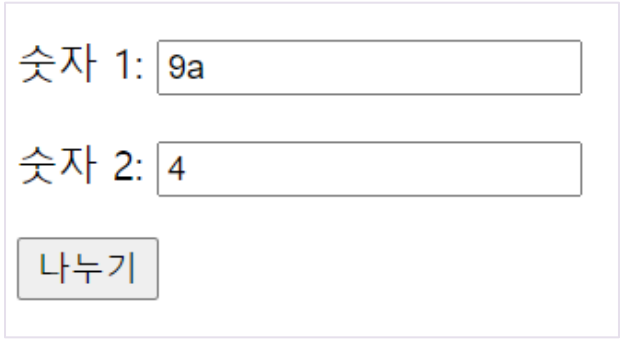
- **try-catch-finally**는 자바의 예외 처리 구문으로 스크립틀릿 태그에 작성한다.

```
try{
        // 예외가 발생할 수 있는 실행문
}catch(처리할 예외 유형){
        // 예외 처리문;
}finally{
        //예외와 상관없이 무조건 실행(생략 가능)
}
```

Exception

try-catch-finally 이용한 예외 처리

- 잘못된 형식의 데이터를 입력한 경우 예외 처리하기



tryCatch.jsp

```
<form action="tryCatch_process.jsp" method="post">
    <p> 숫자 1: <input type="text" name="num1">
    <p> 숫자 2: <input type="text" name="num2">
    <p><input type="submit" value="나누기"> </p>
</form>
```

Exception

try-catch-finally 이용한 예외 처리

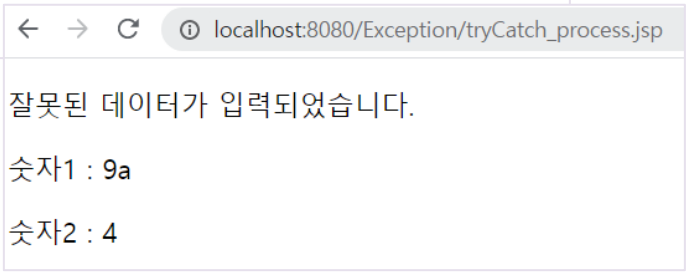
- 잘못된 형식의 데이터를 입력한 경우 예외 처리하기

```jsp
<%
    try{
        String num1 = request.getParameter("num1");
        String num2 = request.getParameter("num2");
        int a = Integer.parseInt(num1);
        int b = Integer.parseInt(num2);
        int c = a / b;
        out.print(num1 + " / " + num2 + " = " + c);
    }catch(NumberFormatException e){
        RequestDispatcher dispatcher = request.getRequestDispatcher("tryCatch_error.jsp");
        //RequestDispatcher 클래스는 포워딩 기능(다른 서블릿이나 JSP로 정보를 전달)을 수행

        dispatcher.forward(request, response);
    }
%>
```

tryCatch_error.jsp

```jsp
<p>잘못된 데이터가 입력되었습니다.
<p> <%=" 숫자1 : " + request.getParameter("num1") %>
<p> <%=" 숫자2 : " + request.getParameter("num2") %>
```



Security

보안(security)

- 시큐리티는 허가된 사용자만이 특정 웹 페이지에 접근할 수 있도록 제한하는 보안 기능을 말한다.
- JSP 컨테이너는 요청된 페이지에 보안 제약이 있는지 확인하고 사용자에게 인증(authentication)을 요청한다. 사용자의 이름과 암호를 확인하여 수행하고 승인한다.
- 시큐리트는 사용자가 권한이 없는 데이터에 접근하는 것을 막거나 웹 공격자가 전송 데이터를 중간에 가로채는 것을 방지하는 등 중요한 역을 한다.

시큐리티 처리 방법	설 명
선언적 시큐리티	코드 작성없이 web.xml 파일에 보안 구성을 작성하여 사용자의 인증을 수행하는 방식이다.
프로그래밍적 시큐리티	request 내장 객체의 메소드를 통해 사용자의 권한 부여를 처리하는 프로그래밍 방식이다.

Security

1. 시큐리티 역할 설정하기

<security-role>은 웹 애플리케이션에 사용하는 역할을 나열하는 요소로 web.xml파일에 구성

```
<security-role>
    <role-name>역할 이름</role-name>
</security-role>
```

2. 시큐리티 제약 사항 설정하기

<security-constraint>는 사용자의 요청 URL에 대한 접근 권한을 정의

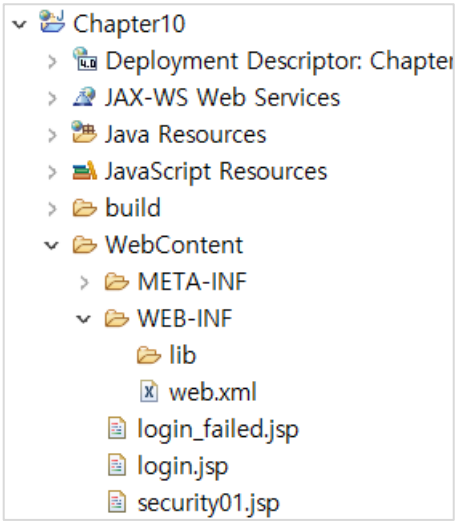
```
<security-constraint>
    <web-resource-collection>컨텍스트 이름</ web-resource-
collection >
    <auth-constraint>인증된 사용자</ auth-constraint>
    <user-data-constraint>데이터보호 설정</ user-data-constraint>
</security-constraint >
```

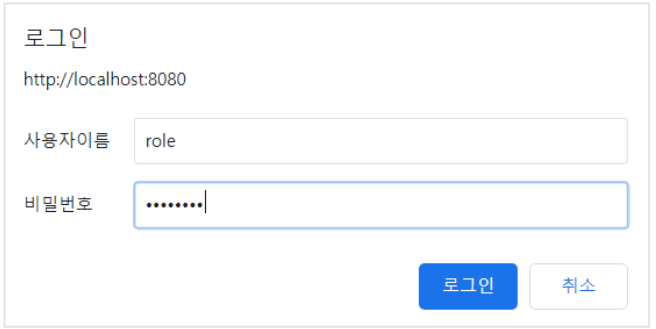
Security

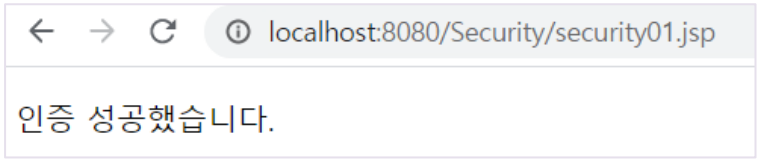
보안(security)

- 기본 인증 처리 방법으로 보안 처리하기.

  security01.jsp로 요청하면 –> 로그인 창이 뜨고 -> 사용자이름, 비밀번호 입력







security01.jsp

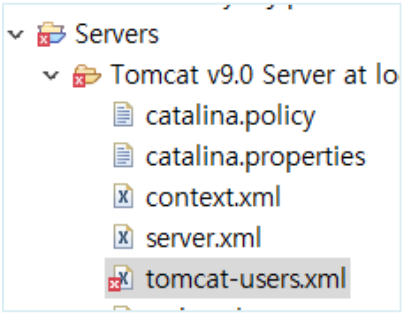
```
<title>Security</title>
</head>
<body>
    <p>인증 성공했습니다.
</body>
```

Security

기본 인증 방법

1. 웹 서버에 사용자와 역할 설정하기

Servers > Tomcat 9 > tomcat-users.xml




```
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="tomcat1234" roles="tomcat"/>
  <user username="both" password="both1234" roles="tomcat,role1"/>
  <user username="role1" password="role1234" roles="role1"/>

</tomcat-users>
```

Security

기본 인증 방법

2. web.xml 에 설정하기

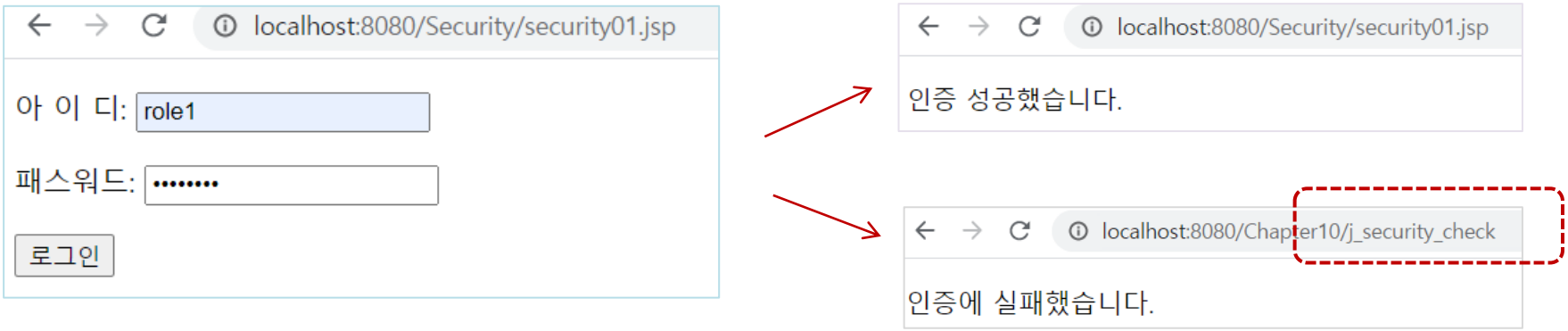
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     <!-- 기본 인증 처리 방법 -->
     <security-role>
        <role-name>role1</role-name>
     </security-role>
     <security-constraint>
        <web-resource-collection>
            <web-resource-name>Security</web-resource-name>
            <url-pattern>/security01.jsp</url-pattern>
            <http-method>GET</http-method>
        </web-resource-collection>
        <auth-constraint>   <!-- 권한이 부여된 사용자 이름 -->
            <description></description>
            <role-name>role1</role-name>
        </auth-constraint>
     </security-constraint>
     <login-config>
        <auth-method>BASIC</auth-method>
     </login-config>
</web-app>
```

Security

보안(security)

- 폼 기반 인증 방법으로 보안 처리하기

security01.jsp로 요청 ->login.jsp로 이동 -> 아이디, 패스워드 입력



주의 사항 -> xml 설정후 서버를 재시작해야함

Security

- 폼 기반 인증 방법으로 보안 처리하기

web.xml 에 등록

```
<security-constraint>
  <web-resource-collection>
      <web-resource-name>Security</web-resource-name>
      <url-pattern>/security01.jsp</url-pattern>
      <http-method>GET</http-method>
  </web-resource-collection>
  <auth-constraint>
      <description>인증된 사용자이름</description>
      <role-name>role1</role-name>
  </auth-constraint>
</security-constraint>

<!--  <login-config>
  <auth-method>BASIC</auth-method>
</login-config> -->

<!-- 폼 기반 인증-->
<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
      <form-login-page>/login.jsp</form-login-page>
      <form-error-page>/login_failed.jsp</form-error-page>
  </form-login-config>
</login-config>
```

기본 인증 방법은 주석처리!!

Security

- 폼 기반 인증 방법으로 보안 처리하기

security01.jsp

```
<title>Security</title>
</head>
<body>
    <p>인증 성공했습니다.
</body>
```

login.jsp

```
<form name="loginForm" action="j_security_check">
    <p>아 이 디: <input type="text" name="j_username">
    <p>패스워드: <input type="password" name="j_password">
    <p><input type="submit" value="로그인"> </p>
</form>
```

login_failed.jsp

```
<body>
    <p>인증에 실패했습니다.
</body>
```