

제어문, 예외처리, 라이프싸이클

강사 : 강병준

제어문

강사 : 강병준

JSP페이지의 연산자

- ❖ 자바의 기본적인 문법을 그대로 사용
- ❖ 식별자(identifier)규칙
 - JSP는 자바 식별자 규칙을 따르는데, 식별자(identifier)란 클래스명, 메소드명, 멤버변수명, 자동변수명등을 일컫는다.
 - 자바 식별자는 길이에겐 제한이 없고 첫 글자는 반드시 영문자,_,_\$로 시작해야 한다. 자바의 클래스명, 메소드명, 인스턴스변수, 자동변수등에 자바 식별자 규칙이 적용
 - 자바는 대소문자를 구별하므로 주의해야한다.
- ❖ 기본데이터타입(primitive data type)
 - 자바 및 JSP는 byte, short, int, long, float, double, char, boolean등의 기본 데이터 타입(primitive data type)을 제공한다.

타입	크기(byte)	자료범위	기본값
byte	1byte	-128 ~ +127	0
short	2byte	-32,768 ~ +32,767	0
int	4byte	-2,147,243,648 ~ +2,147,243,647	0
long	8byte	-9,223,372,036,854,775,808 ~ +9,223,372,036,854,775,807	0
float	4byte	-3.40292347E+38 ~ +3.40292347E+38	0
double	8byte	-1.79769313486231570E+308 ~ +1.79769313486231570E+308	0
char	2byte	'\u0000' ~ '\uFFFF'	0
boolean	1bit	true or false	false

❖ 연산자

- JSP에서는 자바와 마찬가지로 산술연산자, 관계연산자, 논리연산자, 비트연산자, shift연산자, 증감연산자, 조건연산자, 대입연산자들을 사용할 수 있다.

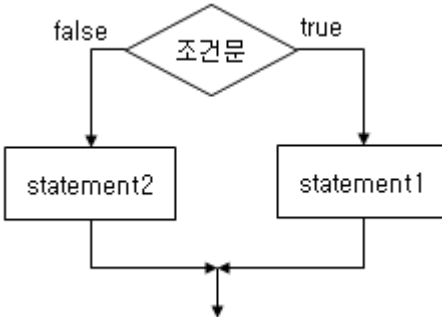
■ 연산자의 종류

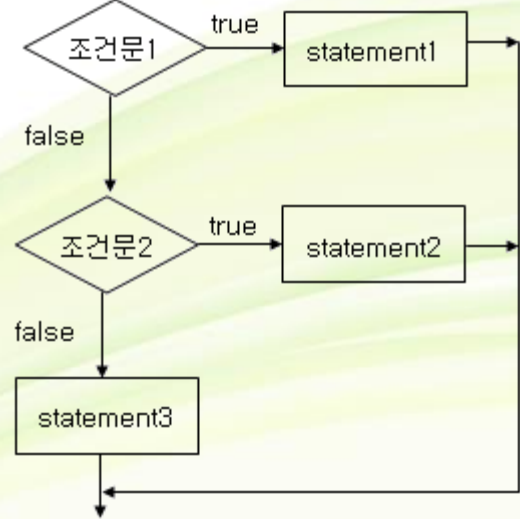
- 산술연산자: * , / , % , + , -
- 관계연산자: < , > , <= , >=
- 논리연산자: && , || , !
- 비트연산자: & , | , ^
- shift연산자: << , >> , >>>
- 증감연산자: ++ , --
- 조건연산자: ? :
- 대입연산자: = , += , -= , *= , /= , %=

JSP페이지의 제어문

❖ if문

- if문은 조건비교 분기문의 하나로 주어진 조건을 비교해서 그 결과에 따라 여러 대안들 중에서 하나를 선택할 때 사용된다.
- if문의 조건에 들어갈 수 있는 타입은 리턴타입 또는 결과 값이 boolean 값을 경우만 가능하다.

문법	순서도(Flowchart)
<pre>if(조건){ statement1; }else{ statement2; }</pre>	

문법	순서도(Flowchart)
<pre>if(조건1){ statement1; }else if(조건2){ statement2; } else{ statement3; }</pre>	

ifTestForm.jsp

```
<html>
<head>
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<title>숫자를 입력하는 폼</title>
</head>
<body>
    <h2>숫자를 입력하세요.</h2>

    <form method="post" action="/ch04_web/ifTestPro.jsp">
        <input type="text" name="number">
        <input type="submit" value="입력완료">
    </form>
</body>
</html>
```

ifTestPro.jsp

```
<html>
<head>
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<title>입력받은 숫자 비교</title>
</head>
<body>
    <h2>입력받은 숫자가 10보다 작거나 같은지 비교</h2>
    <%
        String strNumber = request.getParameter("number");
        int number = Integer.parseInt(strNumber);

        if (number <= 10) {
    %>
        입력받은 숫자는 <%=number %> 입니다.
    <%} %>
</body>
</html>
```

IfDxample.html

```
<h1>If-else Example</h1>
<FORM METHOD=POST ACTION="/ch04_web/IfExample.jsp">
이름 : <INPUT TYPE="text" NAME="name"><p>
좋아하는 색깔 : <SELECT NAME="color">
    <option value="blue" selected>파란색</option>
    <option value="red">붉은색</option>
    <option value="orange">오렌지색</option>
    <option value="etc">기타</option>
</SELECT><p>
<INPUT TYPE="submit" VALUE="보내기">
</FORM>
```

IfExample.jsp

```
<h1>If-else Example</h1>
<%!    String msg; %>
<%    String name = request.getParameter("name");
    String color = request.getParameter("color");
    if (color.equals("blue")) {    msg = "파란색";
    } else if (color.equals("red")) {    msg = "붉은색";
    }else if (color.equals("orange")){    msg = "오렌지색";
    }else{    color = "white";    msg = "기타색";
    }
%>
<body bgcolor=<%=color%>>
<%=name%>님이 좋아하는 색깔은 <%=msg%>입니다.
</body>
```


ifTestMultiForm.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<html>
<head>
<title>이름과 전화번호를 입력하는 폼</title>
</head>
<body>
    <h2>이름과 전화번호를 입력하세요.</h2>
    <form method="post" action="/ch04_web/ifMultiTestPro.jsp">
        이름 : <input type="text" name="name"><br>
        전화번호 :
        <select name="local">
            <option value="서울">서울</option>
            <option value="경기">경기</option>
            <option value="강원">강원</option>
        </select>
        - <input type="text" name="tel"><br>
        <input type="submit" value="입력완료">
    </form>
</body>
</html>
```

ifMultiTestPro.jsp

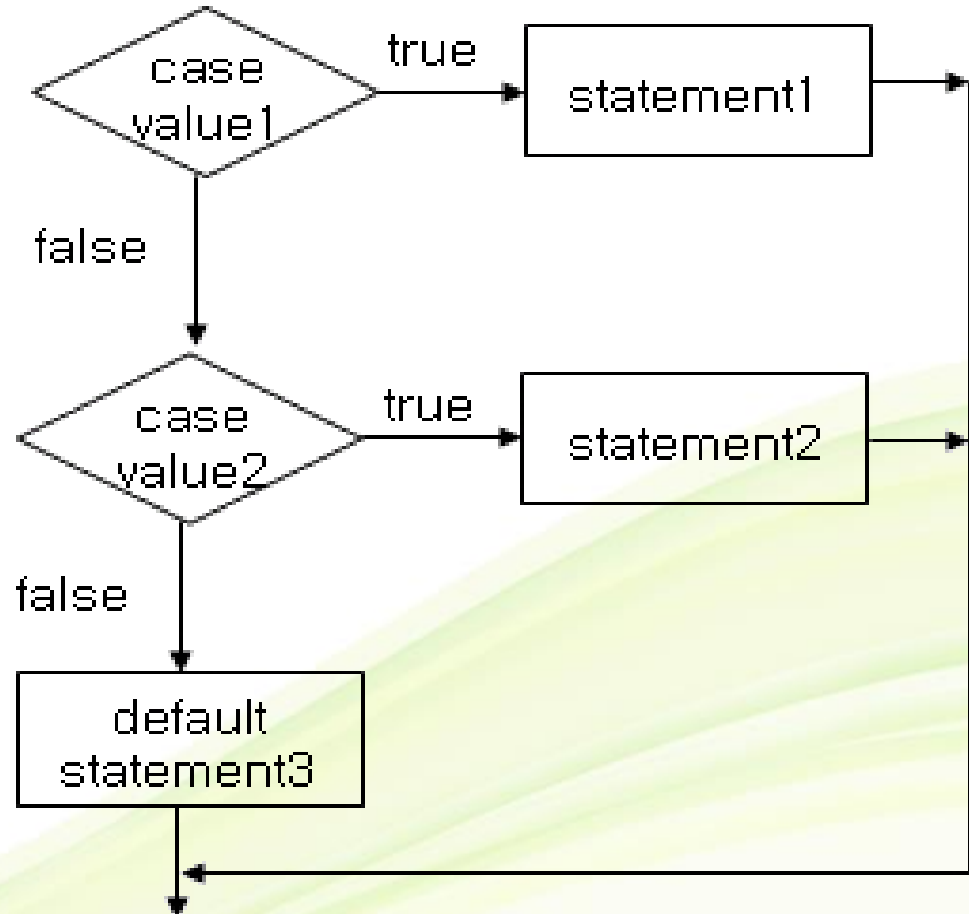
```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<% request.setCharacterEncoding("euc-kr");%>
<%    String name = request.getParameter("name");
    String local = request.getParameter("local");
    String tel = request.getParameter("tel");
    String localNum = "";
    if (local.equals("서울")){ //local변수의 값이 서울일경우
        localNum = "02";
        out.println("<b>" + name + " </b>님의 전화번호는 " + localNum + "-" +
            tel + " 입니다");
    }else if (local.equals("경기")){ //local변수의 값이 경기일경우
        localNum = "031";
        out.println("<b>" + name + " </b>님의 전화번호는 " + localNum + "-" +
            tel + " 입니다");
    }else if (local.equals("강원")){ //local변수의 값이 강원일경우
        localNum = "033";
        out.println("<b>" + name + " </b>님의 전화번호는 " + localNum + "-" +
            tel + " 입니다");
    }
%>
```

❖ Switch문

문법

```
switch(expression)
{
    case value1:
        statement1;
        break;
    case value2:
        statement2;
        break;
    default:
        statement3;
        break;
}
```

순서도(Flowchart)



switchTestForm.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<html>
<head>
<title>권역을 선택하는 폼</title>
</head>
<body>
    <h2>권역을 선택하세요.</h2>
    <form method="post" action="switchTestPro.jsp">
        <input type="radio" name="localNum" value="0" checked>0권역<br>
        <input type="radio" name="localNum" value="1">1권역<br>
        <input type="radio" name="localNum" value="2">2권역<br>
        <input type="radio" name="localNum" value="3">4권역<br>
        <input type="radio" name="localNum" value="4">5권역<br>
        <input type="radio" name="localNum" value="5">6권역<br>
        <input type="radio" name="localNum" value="6">7권역<br>
        <input type="submit" value="입력완료">
    </form>
</body>
</html>
```

switchTestPro.jsp

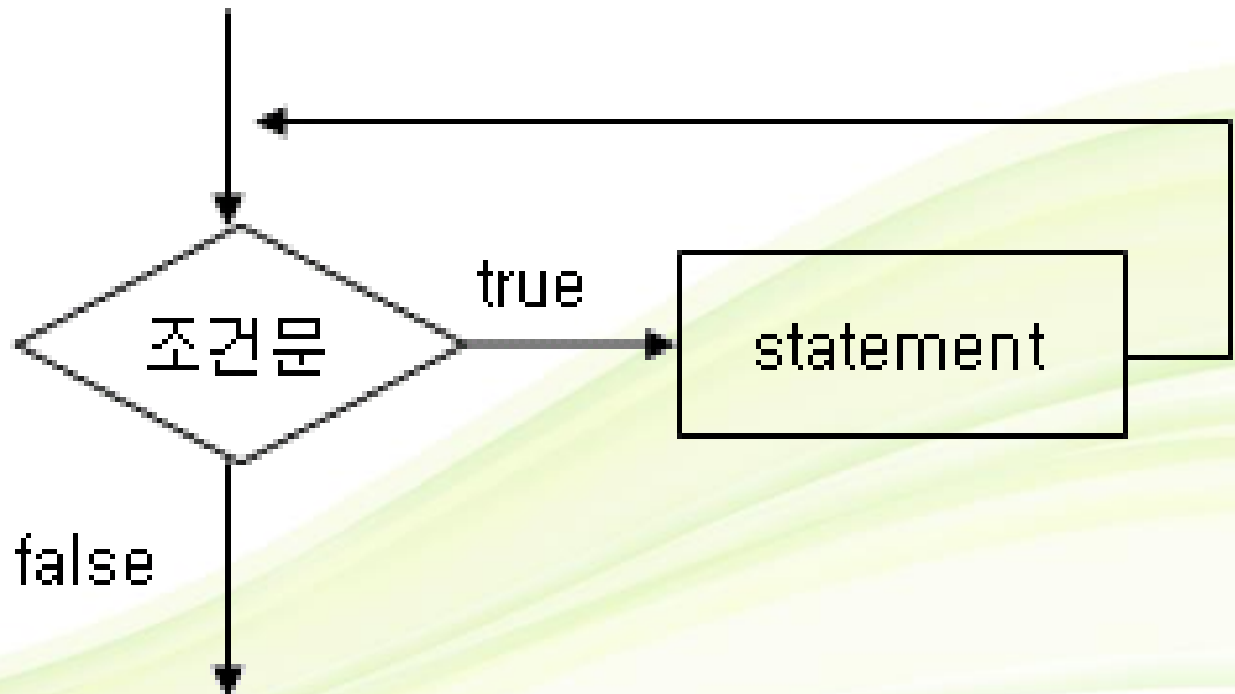
```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<% request.setCharacterEncoding("euc-kr");%>
<%
    int localNum = Integer.parseInt(request.getParameter("localNum"));
    String localName = "";
    switch (localNum){ //localNum변수의 값은 0~7사이의 값
        case 0:    localName = "종로, 중구, 용산";        break;
        case 1:    localName = "도봉, 강북, 노원, 성북";    break;
        case 2:    localName = "동대문, 성동, 중랑, 광진";    break;
        case 3:    localName = "강동, 송파";                break;
        case 4:    localName = "서초, 강남";                break;
        case 5:    localName = "동작, 관악, 금천";          break;
        case 6:    localName = "강서, 양천, 영등포, 구로";    break;
        case 7:    localName = "은평, 마포, 서대문";        break;
        default:   localName = "없는 권역";                break;
    }
    out.println("선택하신 지역은 <b>" + localName + " </b> 입니다");
}%>
```

반복문 – for

❖ 문법

```
for(초기값; 조건문; 증감값){  
    statement;  
}
```

순서도 (Flowchart)



forTest.jsp

```
<%@ page contentType="text/html; charset=euc-kr"
import="java.util.*, java.text.*"
%>
```

```
<html>
```

```
    <body>
```

```
        <font color="blue" size=5>
```

```
            <u> 오늘 날짜 출력 </u></font><br>
```

```
        <%
```

```
            java.util.Date date=new java.util.Date();
```

```
            SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd");
```

```
        %>
```

```
        <%= sdf.format(date) %>
```

```
        <p>
```

```
        <!--
```

```
        <%
```

```
            for(int i=0;i<10;i++)
```

```
                { %>
```

```
                    안녕하세요<br>
```

```
        <% } %> --%>
```

```
</body>
```

whileTest.jsp

```
<%@ page language="java" contentType="text/html;  
    charset=EUC-KR"  
    pageEncoding="EUC-KR"%>
```

```
<%  
    int i = 0;
```

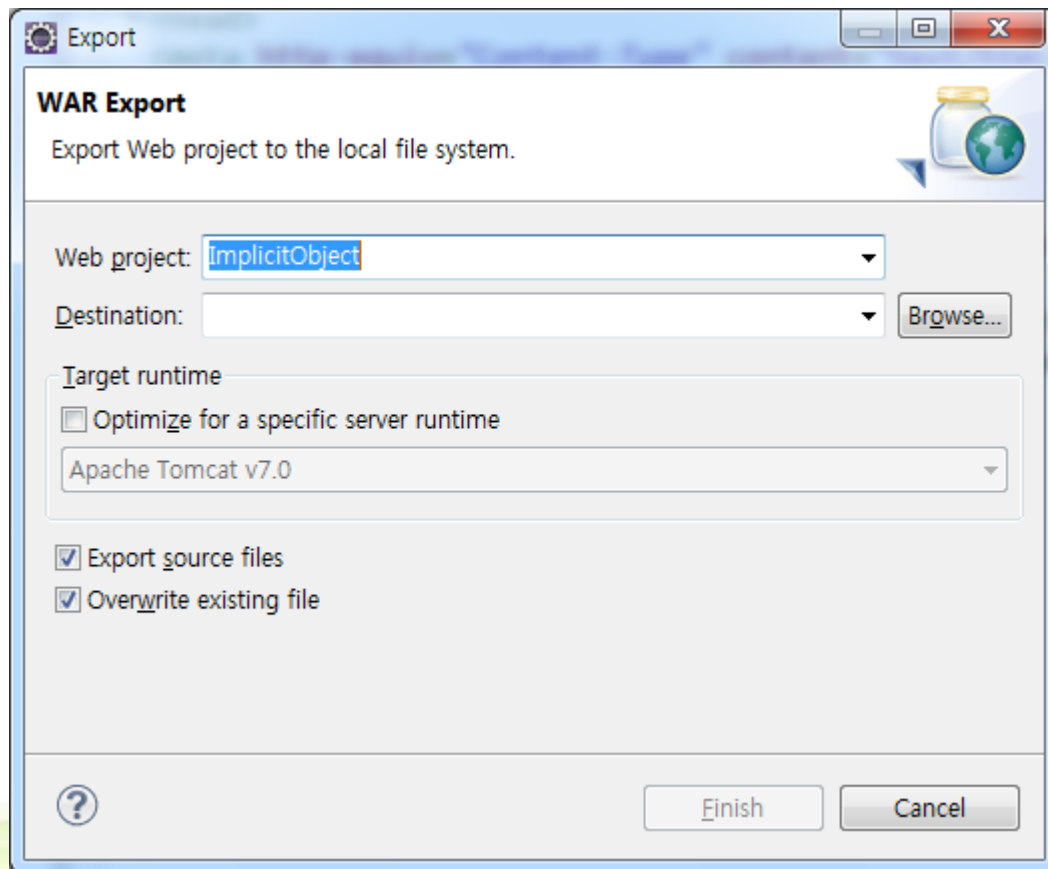
```
    while( i<10){ //0~9까지 값이 출력된다.  
        out.println( "출력되는 값 : " + i + "<br>");  
        i++;  
    }  
%>
```


짝수 입력

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<script type="text/javascript">
do {
    var num = prompt("짝수를 입력하시오", 1);
}while(num%2!=0);
document.write("당신은 짝수 " + num + "을 입력했습니다");
</script>
</body>
</html>>
```

웹 애플리케이션 배포

- ❖ 모든 파일을 직접 배포 가능
- ❖ war(Web Application Archive) 파일로 압축해서 배포
 - ❖ Eclipse에서는 [File] – [Export] – [WAR file]을 선택해서 생성이 가능



익셉션 처리

강사 : 강병준

1. 에러 페이지 지정

- ❖ jsp페이지가 요청을 처리하는 도중에 예외가 발생하면 톰캣 서버가 에러화면을 출력합니다.

HTTP Status 500 - java.lang.NullPointerException

type Exception report

message java.lang.NullPointerException

description The server encountered an internal error that prevented it from fulfilling this request.

exception

```
org.apache.jasper.JasperException: java.lang.NullPointerException
    org.apache.jasper.servlet.JspServletWrapper.handleJspException(JspServletWrapper.java:549)
    org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:470)
    org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:390)
    org.apache.jasper.servlet.JspServlet.service(JspServlet.java:334)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:728)
```

root cause

```
java.lang.NullPointerException
    org.apache.jsp.test_jsp._jspService(test_jsp.java:69)
    org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:70)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:728)
    org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:432)
    org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:390)
    org.apache.jasper.servlet.JspServlet.service(JspServlet.java:334)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:728)
```

Markers Properties Servers Data Source Explorer Snippets Console

Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre7\bin\javaw.exe (2013. 1. 1. 오후 5:43:57)

```
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter
at org.apache.coyote.http11.AbstractHttp11Processor.process(Abstract
at org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.pro
at org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run(JIoEn
at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source
at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Sourc
at java.lang.Thread.run(Unknown Source)
```

에러처리의 개요

- ❖ 웹 페이지에 에러가 발생시 에러발생페이지를 그대로 보여주면 사이트의 신뢰도가 떨어짐
 - 따라서, 에러메시지를 표시하지 말고 다른 페이지를 보여주는것=> 에러페이지
- ❖ page디렉티브의 `errorPage` 속성이 컨테이너의 버전에 따라 제대로 작동을 안할 수 있음.
- ❖ JSP규약에서는 에러코드별 처리를 권장
- ❖ HTTP에러코드에서 자주 발생하는 에러코드
 - 404는 주로 사용자가 잘못된 페이지를 요청할 때
 - 500은 프로그램 코딩 오류일 때 발생

date.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
```

```
    pageEncoding="EUC-KR"%>
```

```
<%@ page import="java.util.*, java.text.*" %>
```

```
<%@ page errorPage="error.jsp"%>
```

```
<%
```

```
    Date date = new Date();
```

```
    SimpleDateFormat simpleDate = new SimpleDateFormat("yyyy-MM-dd");
```

```
    String strdate = simpleDate.format(date);
```

```
%>
```

```
    보통의 JSP 페이지의 형태입니다.<br>
```

```
    오늘 날짜는 <%= strdat%> 입니다. <!--고의로 에러를 발생시킨 라인  
    으로 strdate변수명을 strdat로 틀리게 입력했다. -->
```



AdderInput.html

```
<HTML>
  <HEAD>
    <META http-equiv="Content-Type"
content="text/html; charset=euc-kr">
    <TITLE>덧셈 프로그램 - 입력 화면</TITLE>
  </HEAD>
  <BODY>
    <FORM ACTION=/ch05_web/adder>
      첫 번째 수: <INPUT TYPE=TEXT NAME=NUM1><BR>
      두 번째 수: <INPUT TYPE=TEXT NAME=NUM2><BR>
      <INPUT TYPE=SUBMIT VALUE='더하기'>
    </FORM>
  </BODY>
</HTML>
```



웹 컴포넌트에서 발생하는 익셉션 처리

❖ 스탠드얼론 프로그램과 웹 컴포넌트에서의 익셉션 처리 방법

- 웹 서버와 무관하게 독립적으로 작동하는 스탠드얼론 프로그램에서는 try문을 이용해서 익셉션을 처리한다.

Adder.java

```
public class Adder {  
    public static void main(String args[]) {  
        try {  
            int num1 = Integer.parseInt(args[0]);  
            int num2 = Integer.parseInt(args[1]);  
            int result = num1 + num2;  
            System.out.printf( "%d + %d = %d ", num1, num2, result);  
        }  
        catch (NumberFormatException e) {  
            System.out.println( "잘못된 데이터가 입력되었습니다." );  
        }  
    }  
}
```


❖ 스탠드얼론 프로그램과 웹 컴포넌트에서의 익셉션 처리 방법

AdderServlet.java

```
import javax.servlet.http.*; import javax.servlet.*; import java.io.*;
public class AdderServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println("<html> <body>");
        try {
            int num1 = Integer.parseInt(request.getParameter("num1"));
            int num2 = Integer.parseInt(request.getParameter("num2"));
            int sum = num1 + num2;
            out.printf("%d + %d = %d", num1, num2, sum);
        } catch (NumberFormatException e) {
            out.println("그게 숫자냐");
        }
        out.println("</body> </html>");
        out.close();
    }
}
```

❖ 스탠드얼론 프로그램과 웹 컴포넌트에서의 익셉션 처리 방법

Adder_old.jsp

```
<html> <head> <meta http-equiv= "Content-Type" content="text/html;
charset=UTF-8">
<title>Insert title here</title> </head>
<body>
<%
    try{
        int num1 = Integer.parseInt(request.getParameter("num1"));
        int num2 = Integer.parseInt(request.getParameter("num2"));
        out.println(num1 + " + " + num2 + " = " +(num1+num2));
    }catch(NumberFormatException e ) {
%>
<script type= "text/javascript">
    alert("그게 숫자냐");
    history.go(-1);
</script>
<% } %>
</body>
</html>
```

에러 페이지 만들어서 호출하기

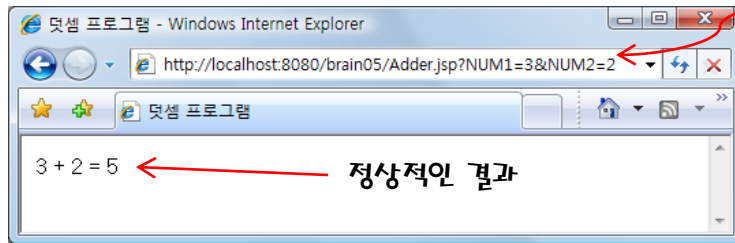
Adder.jsp

```
<html> <head> <meta http-equiv= "Content-Type" content="text/html;
charset=UTF-8">
<title>Insert title here</title> </head>
<body>
<%
    try{
        int num1 = Integer.parseInt(request.getParameter("num1"));
        int num2 = Integer.parseInt(request.getParameter("num2"));
        out.println(num1 + " + " + num2 + " = " +(num1+num2));
    }catch(NumberFormatException e ) {
        request.setAttribute("err", e.getMessage());
        RequestDispatcher rd =
            request.getRequestDispatcher("dataError.jsp");
        rd.forward(request, response);
    }
%>
</body>
</html>
```

❖ JSP 페이지에서 에러 페이지 호출하기

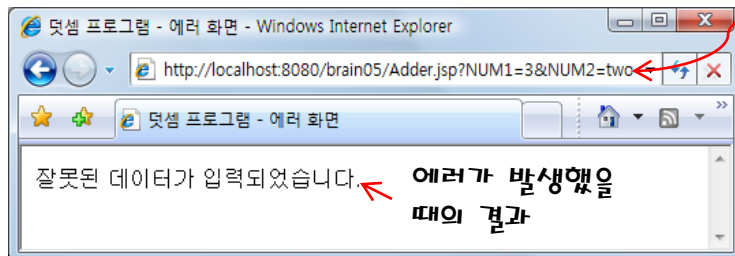
DataError_old.jsp

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>덧셈 프로그램 - 에러 화면</TITLE></HEAD>
  <BODY>
    잘못된 데이터가 입력되었습니다.
  </BODY>
</HTML>
```



(그림 5-4)를 호출하면서 URL 뒤에 데이터를 직접 쓰세요

er.jsp?NUM1=3&NUM2=2



이번에는 입력 데이터 중 하나를 숫자가 아닌 값으로 쓰세요

r.jsp?NUM1=3&NUM2=two

[그림 5-1] 예제 5-4, 5-5의 실행 결과

에러 페이지 지정

- ❖ page 디렉티브를 이용해서 에러 페이지를 지정할 수 있습니다.
- ❖ 에러 페이지 지정 형식: <%@ page errorPage = 페이지 경로 %>
- ❖ 에러 페이지 작성 요령
 - ❖ page 디렉티브에서 isErrorPage 속성을 true로 설정
 - ❖ exception 객체 사용 가능
 - ❖ explorer에서는 전체의 응답 상태 코드가 404나 500 같은 에러코드이고 전체 응답 결과 데이터의 길이가 513바이트 보다 작으면 자체적으로 제공하는 오류 메시지 화면을 출력하므로 유의

test.jsp

```
<%@ page contentType = "text/html; charset=UTF-8" %>
<%@ page errorPage = "/viewErrorMessage.jsp" %>
<html>
<head> <title>파라미터 출력</title> </head>
<body>

name 파라미터 값: <%=
request.getParameter("name").toUpperCase() %>

</body>
</html>
```

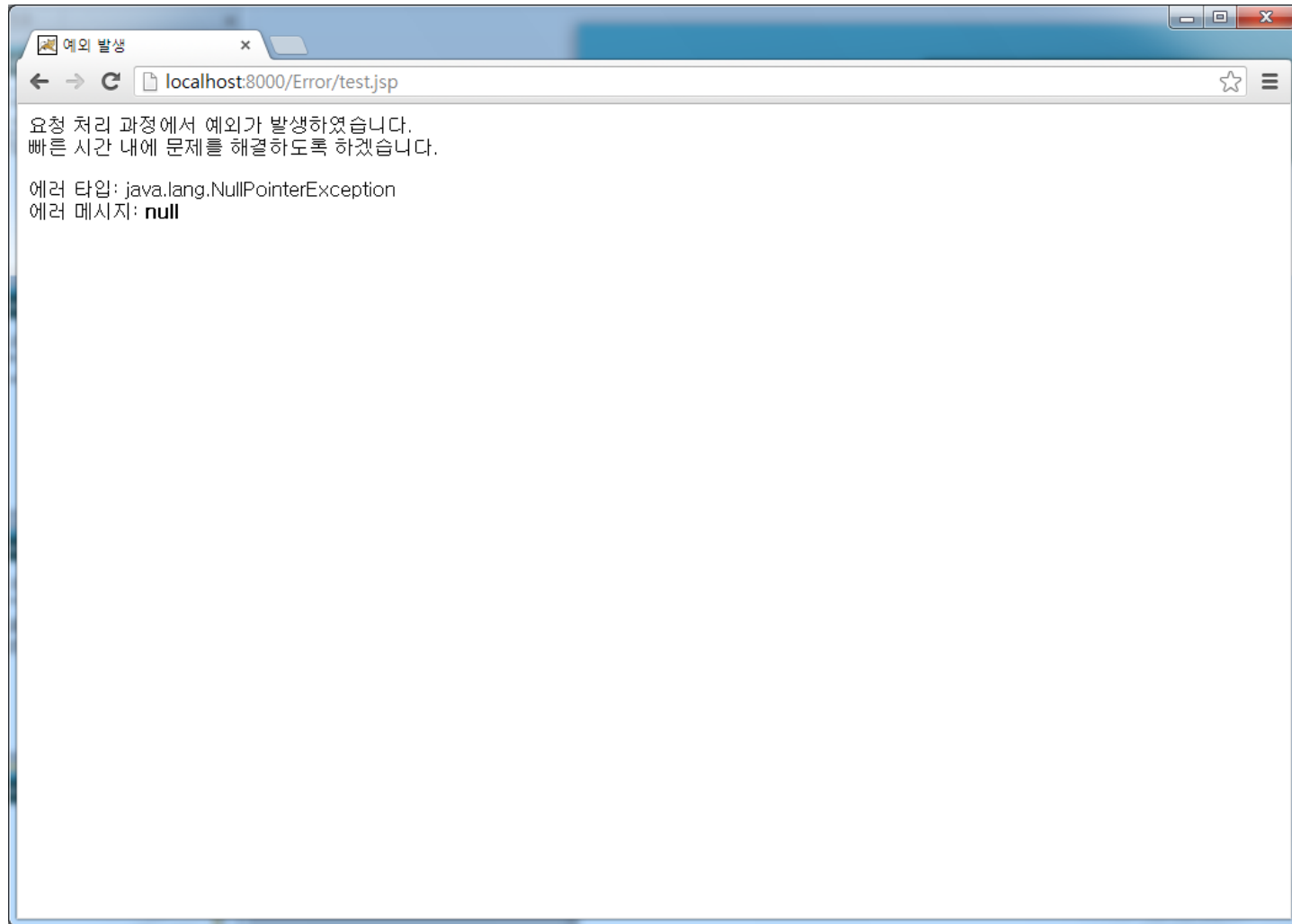
viewErrorMessage.jsp

```
<%@ page contentType = "text/html; charset=UTF-8" %>
<%@ page isErrorPage = "true" %>
<html>
<head> <title>예외 발생 </title> </head>
<body>
```

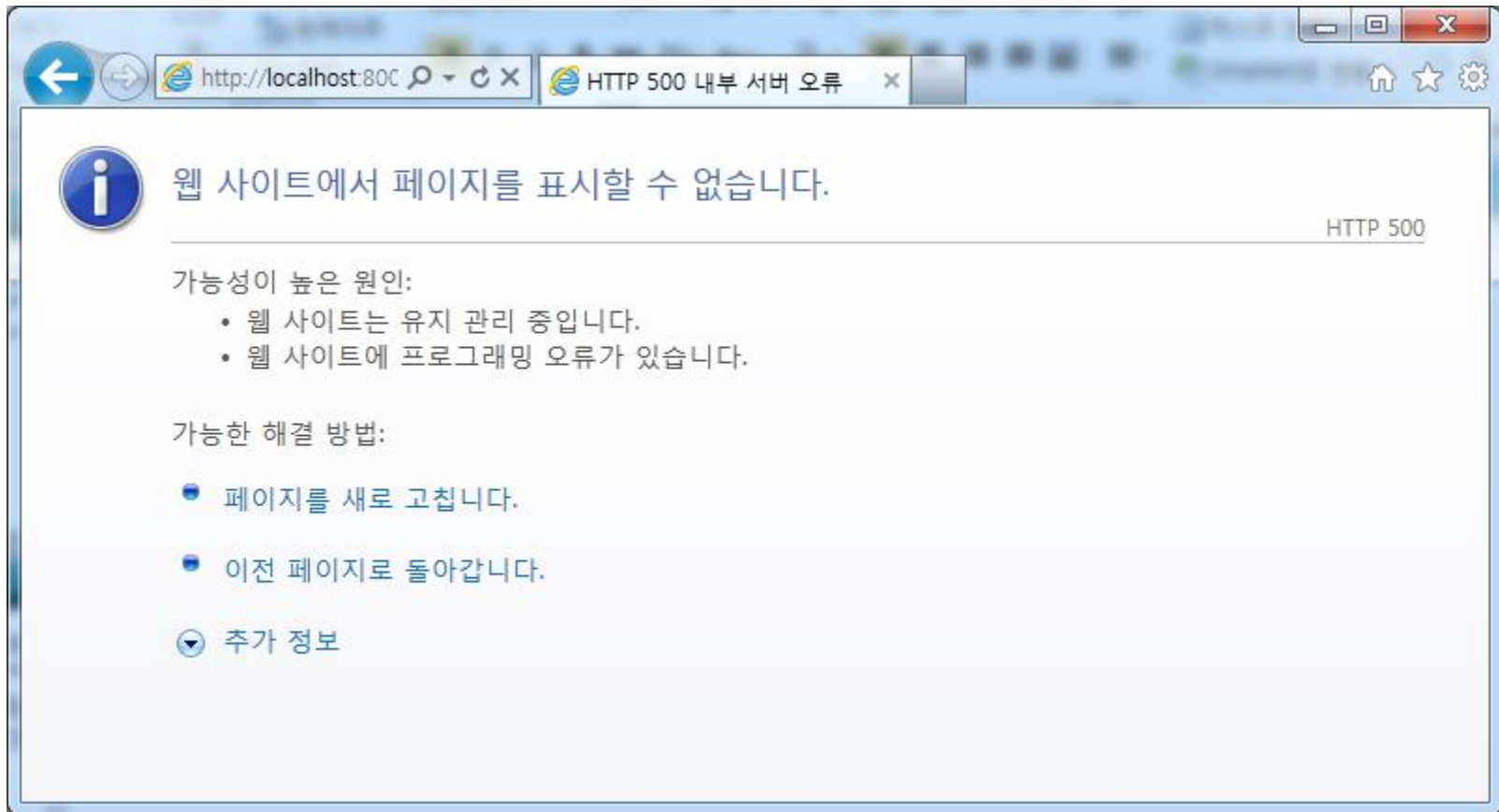
요청 처리 과정에서 예외가 발생하였습니다.
빠른 시간 내에 문제를 해결하도록 하겠습니다.

```
<p>
에러 타입: <%= exception.getClass().getName() %> <br>
에러 메시지: <b><%= exception.getMessage() %></b>
</body>
</html>
```

결과(크롬)



결과(explorer)



error/viewErrorMessage.jsp 수정

```
<%@ page contentType = "text/html; charset=UTF-8" %>
<%@ page isErrorPage = "true" %>
<html>
<head> <title>예외 발생</title> </head>
<body>
요청 처리 과정에서 예외가 발생하였습니다.<br>
빠른 시간 내에 문제를 해결하도록 하겠습니다.
<p>
에러 타입: <%= exception.getClass().getName() %> <br>
에러 메시지: <b><%= exception.getMessage() %></b>
</body>
</html>
<%--
만약 에러 페이지의 길이가 513 바이트보다 작다면
인터넷 익스플로러는 이 페이지가 출력하는 에러 페이지를 출력하지 않고
자체적으로 제공하는 'HTTP 오류 메시지' 화면을 출력할 것입니다.
만약 에러 페이지의 길이가 513 바이트보다 작으면
에러 페이지의 내용이 인터넷 익스플로러에서도 올바르게 출력되길 원한다면
응답 결과에 이 주석과 같은 내용을 포함시켜서
에러 페이지의 길이가 513 바이트 이상이 되도록 해 주어야 합니다.
-->
--%>
```

❖ JSP 페이지에서 에러 페이지 호출하기

두 수를 더하는 JSP 페이지 – page 지시자의 `errorPage` 애트리뷰트 사용

```
<% @page contentType= "text/html; charset=euc-kr" errorPage=
"DataError.jsp" %>
```

```
<%
```

```
    String str1 = request.getParameter( "NUM1 " );
```

```
    String str2 = request.getParameter( "NUM2 " );
```

```
    int num1 = Integer.parseInt(str1);
```

```
    int num2 = Integer.parseInt(str2);
```

```
    int result = num1 + num2;
```

```
%>
```

```
<HTML>
```

```
    <HEAD><TITLE>덧셈 프로그램</TITLE></HEAD>
```

```
    <BODY>
```

```
        <%= num1 %> + <%= num2 %> = <%= result %>
```

```
    </BODY>
```

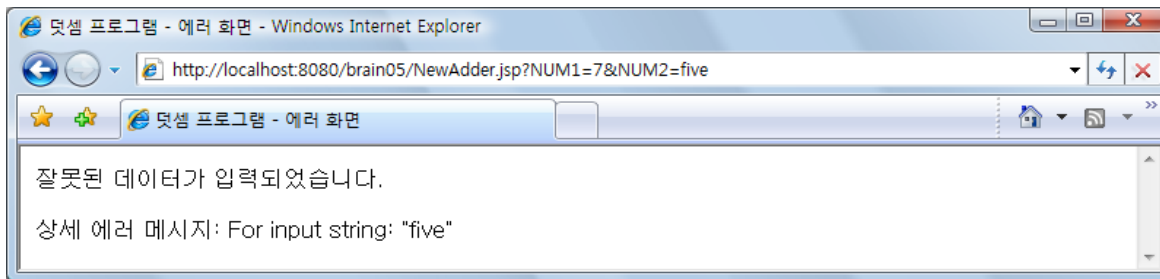
```
</HTML>
```

❖ JSP 페이지에서 에러 페이지 호출하기

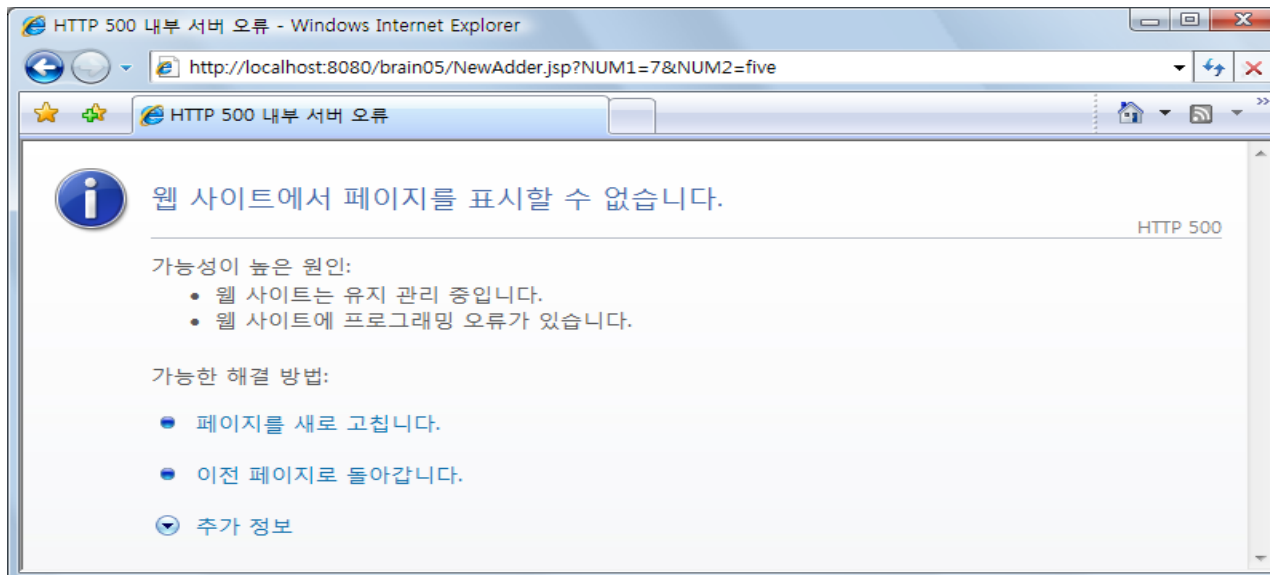
DataError.jsp

```
<% @page contentType= "text/html; charset=euc-kr" isErrorPage= "true" %>
<HTML>
    <HEAD><TITLE>덧셈 프로그램 - 에러 화면</TITLE></HEAD>
    <BODY>
        잘못된 데이터가 입력되었습니다. <BR><BR>
        상세 에러 메시지: <%= exception.getMessage() %>
    </BODY>
</HTML>
```

❖ JSP 페이지에서 에러 페이지 호출하기



[그림 5-3] 예제 5-6, 5-7의 실행 결과(1)



[그림 5-3] 예제 5-6, 5-7의 실행 결과(2)

❖ JSP 페이지에서 에러 페이지 호출하기

- [그림 5-4]는 웹 브라우저에 내장되어 있는 에러 표시용 웹 페이지이며, 이런 결과가 나오는 이유는 [예제 5-7]이 생성한 HTML 문서와 함께 웹 브라우저로 전달된 HTTP 상태 코드 때문이다.
- HTTP 상태 코드란 HTTP 응답 메시지의 시작 행에 표시되는 3자리의 숫자인데, 메시지에 포함된 HTML 문서가 정상적인 실행의 결과인지 에러 발생의 결과인지 구분하는 역할을 한다.

❖ JSP 페이지에서 에러 페이지 호출하기

HTTP 응답 메시지

정상적인 실행 결과일 때

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=euc-kr
Content-Length: 110
Date: Thu, 18 Sep 2008 19:52:11 GMT

<HTML>
  <HEAD><TITLE>덧셈 프로그램 - 결과 화면</TITLE></HEAD>
  <BODY>
    5 + 3 = 8
  </BODY>
</HTML>
```

HTTP 응답 메시지

실행 도중 에러가 발생했을 때

```
HTTP/1.1 500 Internal Server Error
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=euc-kr
Content-Length: 133
Date: Thu, 18 Sep 2008 19:58:53 GMT
Connection: close

<HTML>
  <HEAD><TITLE>덧셈 프로그램 - 에러 화면</TITLE></HEAD>
  <BODY>
    잘못된 데이터가 입력되었습니다.
  </BODY>
</HTML>
```

[그림 5-5] HTTP 응답 메시지의 상태 코드

에러 페이지 만들어서 호출하기

❖ JSP 페이지에서 에러 페이지 호출하기

- 웹 브라우저는 상태 코드 값이 500이면 HTTP 응답 메시지에 포함된 HTML 문서의 내용을 무시하고 웹 브라우저 자체에 내장된 에러 메시지를 출력한다.
- 인위적으로 HTTP 상태 코드의 값을 200으로 바꿔주기 위해서는 response 내장 변수에 대해 setStatus라는 메서드를 호출하면서 200이라는 파라미터값을 넘겨주면 된다.

```
response.setStatus(200);
```

HTTP 상태 코드

DataError.jsp

```
<% @page contentType= "text/html; charset=euc-kr" isErrorPage= "true " %>
<% response.setStatus(200); %>
<HTML>
  <HEAD><TITLE>뎃셈 프로그램 - 에러 발생</TITLE></HEAD>
  <BODY>
    잘못된 데이터가 입력되었습니다. <BR><BR>
    상세 에러 메시지: <%= exception.getMessage() %>
  </BODY>
</HTML>
```


❖ 서블릿 클래스에서 에러 페이지 호출하기

NewAdderServlet.java

```
import javax.servlet.http.*; import javax.servlet.*; import java.io.*;
public class NewAdderServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        try {
            String str1 = request.getParameter( "NUM1 ");
            String str2 = request.getParameter( "NUM2 ");
            int num1 = Integer.parseInt(str1);          int num2 = Integer.parseInt(str2);
            int result = num1 + num2;
            response.setContentType( "text/html;charset=euc-kr ");
            PrintWriter out = response.getWriter();
            out.println( "<HTML> ");
            out.println( "<HEAD><TITLE>덧셈 프로그램</TITLE></HEAD> ");
            out.println( "<BODY> ");
            out.printf( "%d + %d = %d ", num1, num2, result);
            out.println( "</BODY> ");
            out.println( "</HTML> ");
        }
        catch (NumberFormatException e) {
            RequestDispatcher dispatcher = request.getRequestDispatcher( "data-error ");
            dispatcher.forward(request, response);
        }
    }
}
```

❖ 서블릿 클래스에서 에러 페이지 호출하기

DataErrorServlet.java

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class DataErrorServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType( "text/html;charset=euc-kr ");
        PrintWriter out = response.getWriter();
        out.println( "<HTML> ");
        out.println( "<HEAD><TITLE>뎃셈 프로그램 - 에러
화면</TITLE></HEAD> ");
        out.println( "<BODY> ");
        out.println( "잘못된 데이터가 입력되었습니다. " );
        out.println( "</BODY> ");
        out.println( "</HTML> ");
        return;
    }
}
```

응답 상태 코드 별 에러 페이지

❖ web.xml 파일에 작성

```
<error-page>  
    <error-code>에러코드</error-code>  
    <location>에러페이지의 경로</location>  
</error-page>
```

❖ 주요 응답 상태 코드

- ❖ 200 : 요청이 정상적으로 처리
- ❖ 307 : 임시로 페이지가 리다이렉트
- ❖ 400 : 클라이언트의 요청이 잘못된 구문
- ❖ 401 : 접근이 허용되지 않음
- ❖ 404 : 자원이 존재하지 않음
- ❖ 405 : 요청된 메서드는 허용되지 않음
- ❖ 500 : 서버 내부 에러(JSP에서 예외 발생)
- ❖ 503 : 서버가 일시적으로 서비스 할 수 없음(과부하나 임시 보수 중)

실습

⏪ ⏩ 🛑 💰 http://localhost:8000/Error/test1.jsp

요청한 페이지는 존재하지 않습니다:

주소를 올바르게 입력했는 지 확인해보시기 바랍니다.

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app 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
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">
  <error-page>
    <error-code>404</error-code>
    <location>/error/error404.jsp</location>
  </error-page>

</web-app>
```

error/error404.jsp

```
<%@ page contentType = "text/html; charset=UTF-8" %>
```

```
<html>
```

```
<head><title>404 에러 발생</title></head>
```

```
<body>
```

```
<strong>요청한 페이지는 존재하지 않습니다:</strong>
```

```
<br><br>
```

주소를 올바르게 입력했는 지 확인해보시기 바랍니다.

```
</body>
```

```
</html>
```

```
<!--
```

만약 에러 페이지의 길이가 513 바이트보다 작다면,
인터넷 익스플로러는 이 페이지가 출력하는 에러 페이지를 출력하지 않고
자체적으로 제공하는 'HTTP 오류 메시지' 화면을 출력할 것이다.

만약 에러 페이지의 길이가 513 바이트보다 작으데
에러 페이지의 내용이 인터넷 익스플로러에서도 올바르게 출력되길 원한다면,
응답 결과에 이 주석과 같은 내용을 포함시켜서
에러 페이지의 길이가 513 바이트 이상이 되도록 해 주어야 한다.

참고로 이 주석은 456바이트이다.

```
-->
```

예외 타입 별 에러 페이지

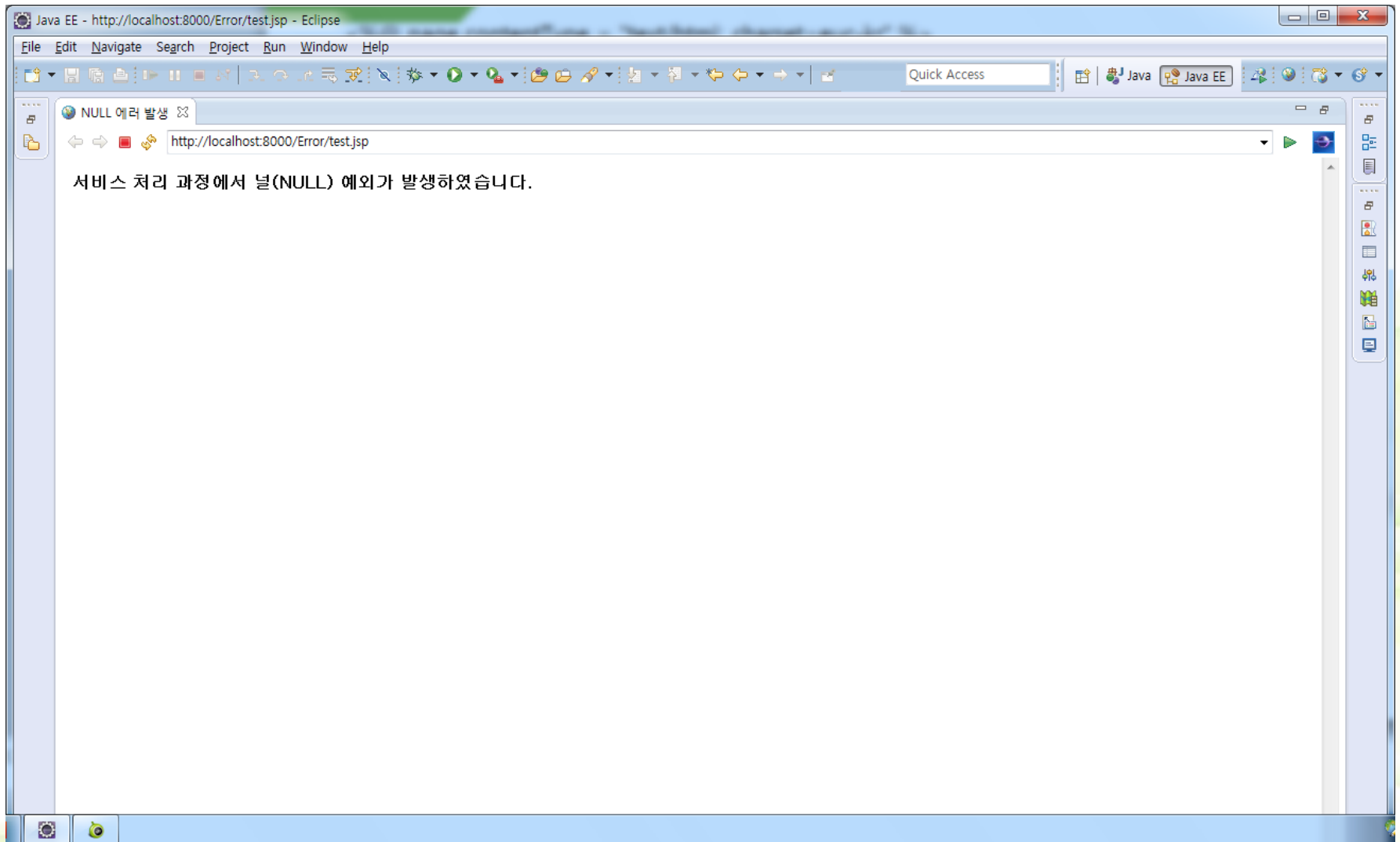
❖ web.xml 파일에 작성

```
<error-page>
```

```
    <exception-type>예외 이름</exception-type>
```

```
    <location>에러 페이지 경로</location>
```

```
</error-page>
```



web.xml 수정

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<web-app 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  
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"  
  version="3.0">
```

```
  <error-page>
```

```
    <error-code>404</error-code>
```

```
    <location>/error/error404.jsp</location>
```

```
  </error-page>
```

```
  <error-page>
```

```
    <exception-type>java.lang.NullPointerException</exception-type>
```

```
    <location>/error/errorNullPointerException.jsp</location>
```

```
  </error-page>
```

```
</web-app>
```

error/errorNullPointerException.jsp

```
<%@ page contentType = "text/html; charset=UTF-8" %>
```

```
<%
```

```
response.setStatus(HttpServletResponse.SC_OK);
```

```
%>
```

```
<html>
```

```
<head> <title>NULL 에러 발생 </title> </head>
```

```
<body>
```

```
<strong>서비스 처리 과정에서 널(NULL) 예외가 발생하였습니다.</strong>
```

```
<%--
```

```
<!--
```

만약 에러 페이지의 길이가 513 바이트보다 작다면

인터넷 익스플로러는 이 페이지가 출력하는 에러 페이지를 출력하지 않고
자체적으로 제공하는 'HTTP 오류 메시지' 화면을 출력할 것입니다.

만약 에러 페이지의 길이가 513 바이트보다 작는데

에러 페이지의 내용이 인터넷 익스플로러에서도 올바르게 출력되길 원한다면

응답 결과에 이 주석과 같은 내용을 포함시켜서

에러 페이지의 길이가 513 바이트 이상이 되도록 해 주어야 합니다.

```
-->
```

```
--%>
```

```
</body>
```

```
</html>
```

test.jsp 파일 수정

```
<%@ page contentType = "text/html; charset=UTF-8" %>
```

```
<html>
```

```
<head> <title>파라미터 출력</title> </head>
```

```
<body>
```

```
name 파라미터 값: <%= request.getParameter("name").toUpperCase() %>
```

```
</body>
```

```
</html>
```

에러 페이지의 우선 순위와 지정

❖ 우선 순위

- ❖ Page 디렉티브의 `errorPage` 속성에서 지정한 페이지
- ❖ JSP 페이지에서 발생한 예외 타입이 `web.xml`에서 지정된 경우
- ❖ JSP 페이지에서 발생한 에러 코드가 `web.xml`에서 지정된 경우
- ❖ 웹 컨테이너의 에러 페이지

❖ 지정 방법

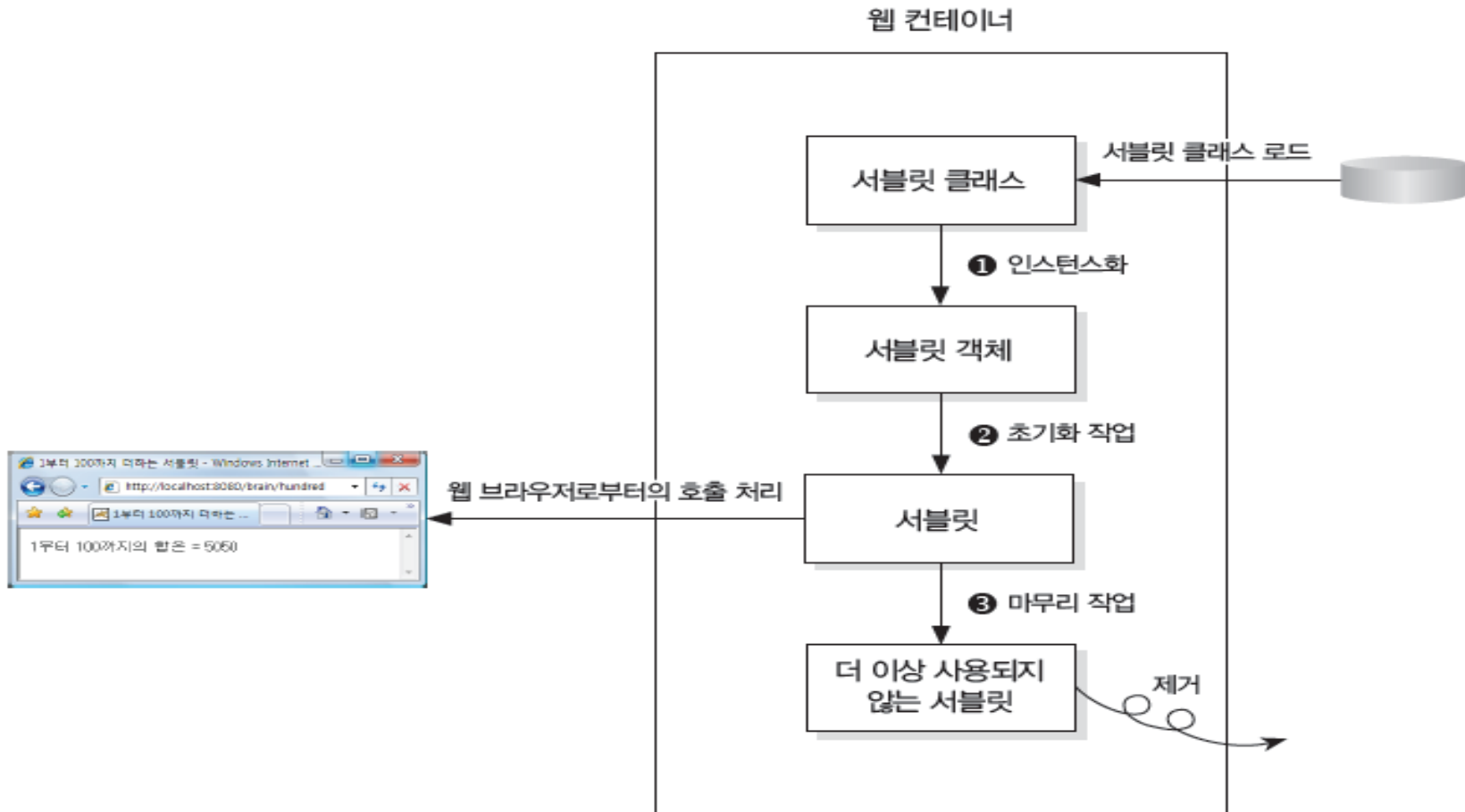
- ❖ 별도의 에러 페이지가 있어야 하는 경우 `page` 디렉티브 속성을 이용
- ❖ 범용적인 에러코드(404, 500)는 `web.xml`에서 처리
- ❖ 별도로 처리해야 하는 심각한 예외 타입에 대해서는 `web.xml` 파일을 이용해서 별도의 에러 페이지를 지정

서블릿의 라이프 사이클

강사 : 강병준

서블릿의 라이프 사이클

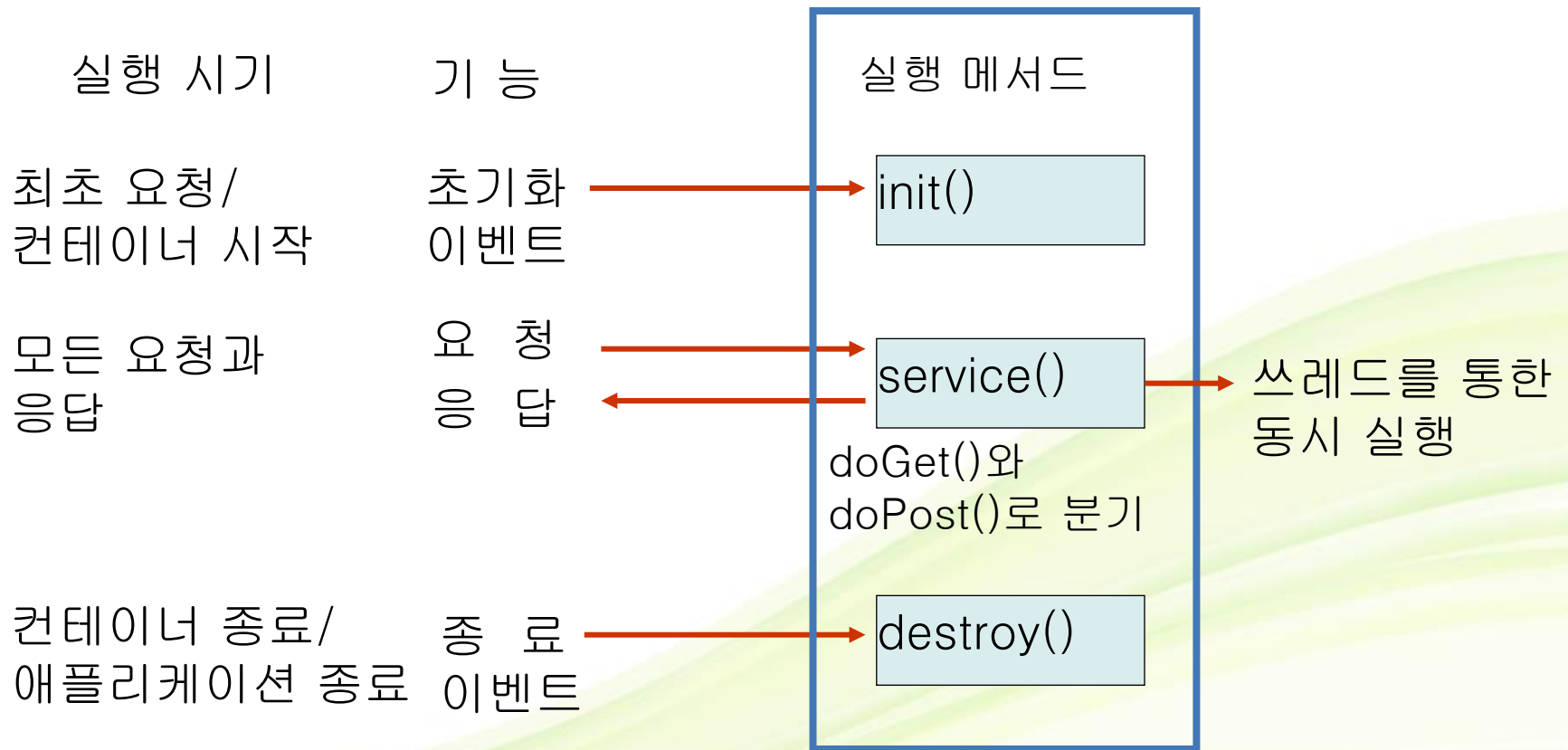
- 서블릿 클래스는 웹 브라우저에 의해 바로 호출되는 것이 아니라 서블릿 클래스로부터 서블릿 객체가 만들어지고, 그 객체가 웹 컨테이너에 의해 초기화된 다음에 호출된다.
- 웹 브라우저의 요청을 처리할 수 있는 상태의 서블릿 객체를 서블릿이라고 한다



서블릿의 라이프 사이클

❖ 서블릿 구조

■ 서블릿 생명 주기



- 웹 컨테이너는 서블릿을 언제 제거할까? 웹 컨테이너는 자신이 종료되기 전이나 웹 애플리케이션을 리로드(unload) 하기 전에 그에 속하는 모든 서블릿을 제거한다.
- 서블릿 라이프 사이클 전체에 걸쳐서 한번만 실행되어야 할 코드는 서블릿 클래스 안에 **init 메서드**와 **destroy**라는 메서드를 선언하고 그 안에 써 놓으면 된다.

```
public class SomeServlet extends HttpServlet {  
    public void init() throws ServletException {
```

```
          
    }  
}
```

서블릿이 초기화될 때 해야 할 일을
기술했는 부분

```
    public void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws IOException, ServletException {  
        out.println( "<HTML> ");  
        out.println( "<HEAD><TITLE>Hello</TITLE></HEAD> ");  
        out.println( "<BODY>Hello, Everyone.</BODY> ");  
        out.println( "</HTML> ");  
    }  
}
```

```
public void destroy() {
```

```
      
}
```

서블릿이 제거되기 전에 해야 할 일을
기술했는 부분

서블릿 클래스의 init 메서드와 destroy 메서드

- JSP 기술에서는 초기화 작업과 마무리 작업 단계에 해야 할 일을 jspInit와 jspDestroy 메서드 안에 써놓으면 웹 컨테이너에 의해 자동으로 호출된다.

```
<%!  
    public void jspInit() {  
          
    }  
%>  
<HTML>  
    <HEAD><TITLE>Hello</TITLE></HEAD>  
    <BODY>  
        Hello, Everyone.  
    </BODY>  
</HTML>  
<%!  
    public void jspDestroy() {  
          
    }  
%>
```

JSP 페이지로부터 변환된 서블릿이
초기화될 때 해야 할 일을 기술하는 부분

JSP 페이지로부터 변환된 서블릿이
제거되기 전에 해야 할 일을 기술하는 부분

JSP 페이지의 jspInit 메서드와 jspDestroy 메서드

서블릿 클래스의 init 메서드와 destroy 메서드

❖ init 메서드의 작성 방법

- **init 메서드는 파라미터가 없는 메서드로 선언해야 하고, 리턴 타입은 void로 지정해야 하며, public 메서드로 선언해야 한다.**

```
public void init() throws ServletException {
```

$$\}$$

우리가 작성할 코드가
들어갈 부분

- 위의 점선으로 표시된 부분에 서블릿 클래스의 초기화 작업 중에 실행해야 할 코드를 써 놓으면 init 메서드가 완성된다.

```
<html><head>
<script language="javascript">
function chk() {
    if (document.form1.num.value=="") {
        alert("데이터를 입력하세요"); document.form1.num.focus();
        return false;    }
    if (isNaN(document.form1.num.value)) {
        alert("숫자를 입력하세요"); document.form1.num.value="";
        document.form1.num.focus();    return false;
    }
    return true;
}
</script></head><body><h3>숫자를 입력하세요</h3>
<form action="FibonacciServlet" name="form1" onSubmit="return chk()">
    <input type="text" name="num"></input><br></br>
    <input type="submit" value="완료"></input>
</form>
</body>
</html>
```

❖ init 메서드의 작성 방법

FibonacciServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import java.math.BigInteger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class Fibona extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private BigInteger arr[];
    public Fibona() {      super();  }
    public void init(){
        arr=new BigInteger[100];
        arr[0]=new BigInteger("1"); arr[1]=new BigInteger("1");
        for (int i=2;i<arr.length;i++){
            arr[i]=arr[i-2].add(arr[i-1]);
        }
    }
}
```

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
    String str = request.getParameter( "NUM " );
    int num = Integer.parseInt(str);
    if (num > 100)
        num = 100;
    response.setContentType( "text/html;charset=euc-kr " );
    PrintWriter out = response.getWriter();
    out.println( "<HTML> " );
    out.println( "<HEAD><TITLE>피보나치 수열</TITLE></HEAD> " );
    for (int cnt = 0; cnt < num; cnt++)
        out.println(arr[cnt] + " ");
    out.println( "</BODY> " );
    out.println( "</HTML> " );
}
```

❖ destroy 메서드의 작성 방법

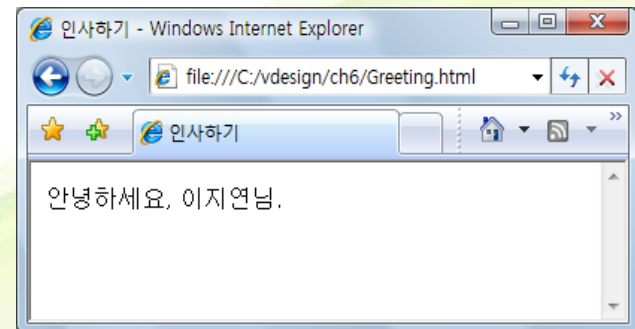
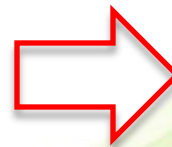
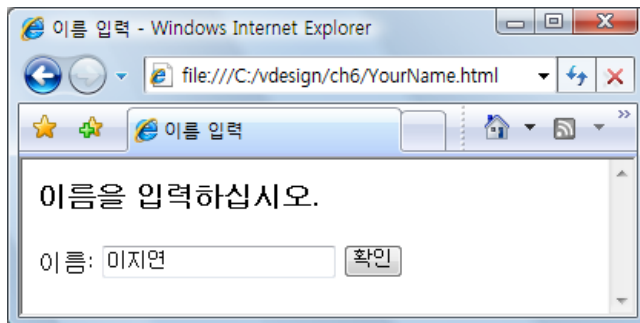
- destroy 메서드의 작성 방법은 init 메서드와 비슷하다. 파라미터가 없어야 하고, 리턴 타입은 void여야 하고, public 메서드로 선언해야 한다.
- 하지만 init 메서드와 달리 throws 절을 쓸 수 없다.

```
public void destroy() {
```



우리가 작성할 코드가
들어가는 부분

```
}
```



인사말을 출력하는 웹 애플리케이션의 화면 설계

❖ destroy 메서드의 작성 방법

http://localhost:8181/ch06_web/YourName.html

왼쪽 화면의 URL

http://localhost:8181/ch06_web/greeting

오른쪽 화면의 URL

YourName.html

```
<HTML>
  <HEAD>
    <META http-equiv= "Content-Type" content= "text/html; charset=euc-kr" >
    <TITLE>이름 입력</TITLE>
  </HEAD>
  <BODY>
    <H3>이름을 입력하십시오.</H3>
    <FORM ACTION=greeting>
      이름: <INPUT TYPE=TEXT NAME=NAME>
      <INPUT TYPE=SUBMIT VALUE= '확인' >
    </FORM>
  </BODY>
</HTML>
```

❖ destroy 메서드의 작성 방법

GreetingServlet.java

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
import java.util.*;

public class GreetingServlet extends HttpServlet {
    private PrintWriter logFile;
    public void init() throws ServletException {
        try {
            logFile = new PrintWriter(new FileWriter("c:\\data\\log.txt "));
        }
        catch (IOException e) {
            throw new ServletException(e);
        }
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        String name = request.getParameter( "NAME ");
        String greeting = “안녕하세요, ” + name + “님. ” ;
    }
}
```



```
if (logFile != null) {  
    GregorianCalendar now = new GregorianCalendar();  
    logFile.printf( “%TF %TT - %s %n ”, now, now, name);  
}  
response.setContentType( “text/html;charset=euc-kr ”);  
PrintWriter out = response.getWriter();  
out.println( “<HEAD><TITLE>인사하기</TITLE></HEAD> ”);
```

```
out.println( “<BODY> ”);  
out.println(greeting);  
out.println( “</BODY> ”);  
out.println( “</HTML> ”);  
}  
public void destroy() {  
    if (logFile != null)  
        logFile.close();  
}  
}
```

JSP 페이지의 jspInit 메서드와 jspDestroy 메서드

❖ jspInit 메서드와 jspDestroy 메서드의 작성 방법

- jspInit 메서드의 작성 방법은 서블릿 클래스의 init 메서드와 비슷하다.
- 파라미터가 없는 메서드로 만들어야 하고, 리턴 타입은 void로 지정해야 하고, public 메서드로 선언해야 한다. 하지만 throw 절을 쓸 수 없다는 점은 init 메서드와 다르다.

```
public void jspInit() {
```

```
}
```

우리가 작성할 코드가
들어가는 부분

- jspDestroy 메서드의 작성 규칙은 서블릿 클래스의 destroy 메서드와 비슷하다. 파라미터가 없어야 하고, 리턴 타입은 void로 지정하며, public 메서드로 선언해야 한다.

```
public void jspDestroy() {
```

```
}
```

우리가 작성할 코드가
들어가는 부분

❖ jspInit 메서드와 jspDestroy 메서드의 작성 방법

jspInit, jspDestroy 메서드의 사용 예를 보여주는 JSP 페이지

```
<% @page contentType= "text/html; charset=euc-kr" import= "java.io.*,  
java.util.*" %>  
<%!  
    private PrintWriter logFile;  
    public void jspInit() {  
        String filename = "c:\\data\\datetime_log.txt";  
        try {  
            logFile = new PrintWriter( new FileWriter(filename));  
        }  
        catch (IOException e) {  
            System.out.printf( "%TT - %s 파일을 열 수 없습니다. %n" , new  
GregorianCalendar(), filename);  
        }  
    }  
%>
```

<HTML>

<HEAD><TITLE>**현재의 날짜와 시각**</TITLE></HEAD>

<BODY>

<%

GregorianCalendar now = new GregorianCalendar();

String date = String.format("**현재 날짜**: %TY**년** %Tm**월** %Te**일** ", now, now, now);

String time = String.format("**현재 시각**: %TI**시** %Tm**분** %TS**초** ", now, now, now);

out.println(date + "
 ");

out.println(time + "
 ");

if (logFile != null)

logFile.printf("%TF %TT**에 호출되었습니다.**%n ", now, now);

%>

</BODY>

</HTML>

<%!

public void jspDestroy() {

if (logFile != null)

logFile.close();

}

%>

서블릿의 환경을 표현하는 ServletContext 객체

❖ 서블릿의 환경 정보를 가져오는 방법

- 서블릿 클래스나 JSP 페이지의 환경에 관련된 정보는 javax.servlet 패키지의 ServletContext 인터페이스 타입의 객체를 이용해서 얻을 수 있다.
- 서블릿 클래스에서 이 타입의 객체를 구하기 위해서는 getServletContext라는 메서드를 호출하면 된다.

```
ServletContext context = getServletContext();
```

ServletContext 객체를 리턴하는 메서드

- ServletContext 객체에 대해 getServerInfo라는 메서드를 호출하면 서블릿이 속하는 웹 서버 종류가 리턴된다.

```
String str = context.getServerInfo();
```

웹 서버의 종류를 리턴하는 메서드

❖ 서블릿의 환경 정보를 가져오는 방법

- ServletContext 객체에 대해 getMajorVersion과 getMinorVersion이라는 메서드를 호출하면 웹 컨테이너가 지원하는 서블릿 규격서의 메이저 버전과 마이너 버전이 리턴된다.

```
int num1 = context.getMajorVersion();
```

서블릿의 메이저 버전을 가져오는 메서드

```
int num2 = context.getMinorVersion();
```

서블릿의 마이너 버전을 가져오는 메서드

❖ 서블릿의 환경 정보를 가져오는 방법

ServerInfoServlet.java

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;

public class ServerInfoServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        ServletContext context = getServletContext();
        String serverInfo = context.getServerInfo();
        int majorVersion = context.getMajorVersion();
        int minorVersion = context.getMinorVersion();
        response.setContentType( "text/html;charset=euc-kr ");
        PrintWriter out = response.getWriter();
        out.println( "<HTML> ");
        out.println( "<HEAD><TITLE>웹 서버의 정보</TITLE></HEAD> ");
        out.println( "<BODY> ");
        out.printf( "웹 서버의 종류: %s <BR> ", serverInfo);
        out.printf( "지원하는 서블릿 버전: %d.%d <BR> ", majorVersion, minorVersion);
        out.println( "</BODY> ");
        out.println( "</HTML> ");
    }
}
```