

1. 다운로드 및 설치

<https://tomcat.apache.org/download-taglibs.cgi>에서 jar 파일 4개를 다운받자.

- [zip \(pgp, sha512\)](#)

Jar Files

- [Binary README](#)
- Impl:
 - [taglibs-standard-impl-1.2.5.jar \(pgp, sha512\)](#)
- Spec:
 - [taglibs-standard-spec-1.2.5.jar \(pgp, sha512\)](#)
- EL:
 - [taglibs-standard-jstlel-1.2.5.jar \(pgp, sha512\)](#)
- Compat:
 - [taglibs-standard-compat-1.2.5.jar \(pgp, sha512\)](#)

2. 압축을 풀고 lib디렉토리로 이동. jar 파일4개를 복사하여
내가 만든 톰캣 프로젝트(예를 들어 test)의 WEB-INF/lib/아래에 붙여 넣자.

2. 표현언어(Expression Language)란?

1) JSTL(JSP standard Tag Library)1.0 규약에 소개된 내용으로서 JSP2.0규약에 새롭게 추가된 기능.

2) 값을 표현하는데 사용되는 새로운 스크립트 언어로서 JSP의 기본 문법을 보완하는 역할을 한다.

3. 표현언어가 제공하는 기능

- 1) JSP의 네가지 기본 객체가 제공하는 영역의 속성 사용
- 2) 집합 객체에 대한 접근 방법 제공
- 3) 수치 연산, 관계 연산, 논리 연산자 제공
- 4) 자바 클래스 메소드 호출 기능 제공
- 5) 표현 언어만의 기본 객체 제공

4. 표현언어의 표현 방법

\$와 표현식 그리고 괄호('{ ' 와 ' } ')를 사용하여 값을 표현한다.

`${expr}`

expr부분에는 표현언어가 정의한 문법에 따라 값을 표현하는 식이 온다.

예] `${sessionScope.memeber.id}`님 환영합니다.

5. 표현언어의 기본 객체

JSP의 기본 내장객체(예, request, session 등)와 마찬가지로 EL(표현언어)에서도 11개의 기본 객체를 제공하고 있다.


N0	기본 객체	설 명
1	pageContext	JSP의 page 기본객체와 동일
2	pageScope	pageContext기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map객체
3	requestScope	request 기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map객체

N0	기본 객체	설 명
4	sessionScope	session 기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map객체
5	applicationScope	application 기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map객체
6	requestScope	request 기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map객체
7	param	요청 파라미터의 <파라미터 이름, 값> 매핑을 저장한 Map 객체, 파라미터 값의 타입은 String으로서, request.getParameter(이름)의 결과와 동일
8.	paramValues	요청 파라미터의 <파라미터 이름, 값 배열> 매핑을 저장한 Map 객체, 파라미터 값의 타입은 String[]으로서, request.getParameterValues(이름)의 결과와 동일

9	header	요청정보의 <헤더 이름, 값> 매핑을 저장한 Map 객체, request.getHeader(이름)와 동일
10	headerValues	요청정보의 <헤더 이름, 값 배열> 매핑을 저장한 Map 객체. request.getHeaders(이름)와 동일
11	cookie	<쿠키 이름, Cookie> 매핑을 저장한 Map 객체. request.getCookies()로 구한 Cookie 배열로부터 매핑을 생성한다.
12	initParam	초기화 파라미터의 <이름, 값> 매핑을 저장한 Map 객체 application.getInitParameter(이름)의 결과와 동일

예제

test1.jsp
<pre> <%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%> <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <% request.setAttribute("id","Swan"); request.setAttribute("name","백성애"); %> 아이디 : \${requestScope.id}
 이름 : \${requestScope.name}
 <hr color=green> <form name="f"> 주소: <input type=text name=addr>
 전화번호: <input type=text name=tel>
 주소: <input type=submit>
 </form> <hr color=red> 주소: \${param.addr}
 전화번호: \${param.tel}
 </pre>

<p>실행결과</p>	 <ul style="list-style-type: none"> • 아이디 : Swan • 이름 : 백성애 <hr/> <p>주소: <input type="text"/></p> <p>전화번호: <input type="text"/></p> <p>주소: <input type="button" value="쿼리 전송"/></p> <hr/> <ul style="list-style-type: none"> • 주소: 서울 영등포구 당산동 • 전화번호: 111-4111
-------------	---

5. EL의 데이터 타입

- boolean타입: true와 false
- 정수 타입: 0~9로 이루어진 정수 값.
- 실수 타입: 0~9로 이루어져 있으며, 소수점(.)을 사용할 수 있고, 7.24e5와 같이 지수형으로 표현 가능하다.
- 문자열 타입: 따옴표(' 또는 ")로 둘러싼 문자열.
- 널 타입: null

6. 연산자

6.1. 수치 연산자

덧셈: +
 뺄셈: -
 곱셈: *
 나눗셈: / 또는 div
 나머지: % 또는 mod

자바 연산자와 동일하며 나눗셈과 나머지 연산의 경우 div와 mod를 추가적으로 사용할 수 있다.

숫자가 아닌 객체와 수치 연산자를 사용할 경우, 객체를 숫자 값으로 변환한 후 연산자를 수행한다.

ex)

```
${"5"+3}
```

위의 코드는 문자열 "5"를 먼저 숫자 5로 변환한 후 덧셈을 수행한다.
따라서 결과는 8을 출력한다.

만약 아래와 같이 숫자로 변환할 수 없는 객체와 수치 연산자를 함께 사용하면 에러를 발생시킨다.

```
${"오"+3}
```

만약 수치 연산자에서 사용되는 객체가 null이면 0으로 처리된다. 예를 들어 다음의 코드 결과는 3이다.

```
${null+3}
```

6.2. 비교 연산자

EL이 제공하는 비교 연산자는 6개가 있다.

- 1) == 또는 eq
- 2) != 또는 ne
- 3) < 또는 lt
- 4) > 또는 gt
- 5) <= 또는 le
- 6) >= 또는 ge

6.3 논리 연산자

- 1) && 또는 and
- 2) || 또는 or
- 3) ! 또는 not

6.4 empty 연산자

empty <값>

empty 연산자는 검사할 객체가 텅 빈 객체인지 검사하기 위해 사용한다.

여기서 <값>에 따라서 리턴 되는 값은 다음과 같이 결정된다

- 1) 값이 null이면 true를 리턴
- 2> 값이 빈 문자열("")이면 true를 리턴
- 3> 값이 길이가 0인 배열이면 true를 리턴
- 4> 값이 빈 Map이면 true를 리턴
- 5> 값이 빈 Collection이면 true를 리턴
- 6> 이외의 경우는 false를 리턴 한다.

6.5 비교 선택 연산자

<수식>? <값1> : <값2>

<수식>의 결과 값이 true이면 <값1>을 리턴 하고 false이면 <값2>를 리턴 한다.

6.6 연산자 우선 순위

```
[ ]  
( )  
-(단일) not ! empty  
* / div % mod  
+ -  
< > <= >= lt gt le ge  
== != eq ne  
&& and  
|| or  
? :
```

- JSTL은 계속적으로 사용이 증가되고 있는 추세다.
- 많이 사용되는 사용자 정의 태그를 모아서 JSTL이라는 규약이 만들어짐.
- 스크립틀릿, 표현식을 사용하는 것보다 훨씬 간결한 문법 구조를 지원.
- JSTL은 5가지의 태그를 지원합니다.

라이브러리 기능 접두어 관련 URL

Core	변수지원, 흐름 제어, URL 처리	c	http://java.sun.com/jsp/jstl/core ★
XML	XML 코어, 흐름 제어, XML 변환	x	http://java.sun.com/jsp/jstl/xml
국제화	지역, 메시지 형식, 숫자 및 날짜 형식	fmt	http://java.sun.com/jsp/jstl/fmt
데이터베이스	SQL	sql	http://java.sun.com/jsp/jstl/sql
함수	컬렉션 처리, String 처리	fn	http://java.sun.com/jsp/jstl/functions ★

☒ Core Tag

기능 태그명 기능설명

변수 지원 **set** jsp 에서 사용될 변수를 설정합니다.

remove 설정한 변수를 제거합니다.

흐름 제어 **if** 조건에 따라 내부 코드를 수행합니다.

choose 다중 조건을 처리할 때 사용됩니다.

forEach Collection 의 각 항목을 처리할 때 사용합니다.

forTokens 구분자로 분리된 각각의 토큰을 처리할 때 사용합니다.

URL 처리 **import** URL 을 사용하여 다른 자원의 결과를 삽입합니다.

redirect 지정한 경로로 이동합니다.

url URL 을 재 작성합니다.

기타 태그 `catch` 예외 처리에 사용합니다.

`out` `jspWriter`에 내용을 알맞게 처리한 후 출력합니다.

7.1. set/remove 태그

`<c:set/>`의 기본형식은 다음과 같다. `scope` 속성이 생략될 경우 기본값은 `page`이다.

Syntax 1: `scope`에 해당하는 변수에 속성 값을 정한다.

```
<c:set value="value"
      var="varName" [scope="{page|request|session|application}"]/>
```

Syntax 2: `scope`에 해당하는 변수에 `body` 값을 정한다.

```
<c:set var="varName" [scope="{page|request|session|application}"]>
  body content
</c:set>
```

Syntax 3: 속성 값으로 `target` 객체의 프로퍼티 값을 정한다.

```
<c:set value="value"
      target="target" property="propertyName"/>
```

Syntax 4: `body` 값으로 `target` 객체의 프로퍼티 값을 정한다.

```
<c:set target="target" property="propertyName">
  body content
</c:set>
```

변수에 값을 할당한다. 빈과 같은 객체에 할당하기 위해서는 `target` 과 `property` 속성을 이용한다.

`<c:remove/>`는 JSP의 `removeAttribute()`와 같은 역할을 한다. 해당 `scope`에 있는 변수를 제거하는 역할을 한다.

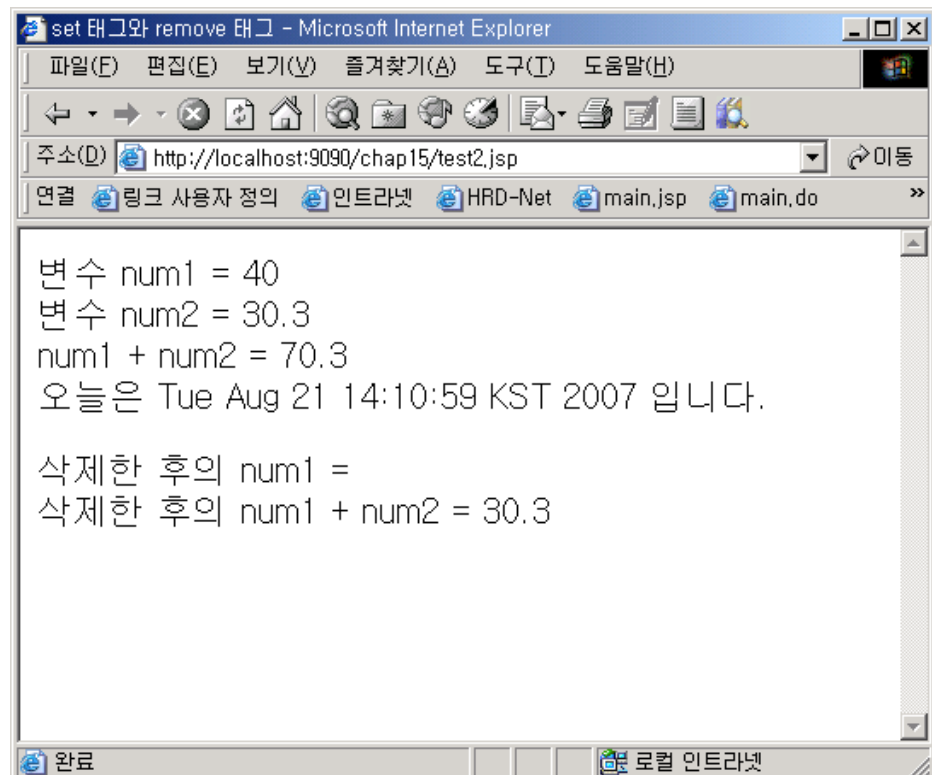
test2.jsp : set 태그와 remove 태그

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="num1" value="${40}" />
<c:set var="num2">30.3</c:set>
<c:set var="today" value="<%= new java.util.Date() %>" />
<html>
<head>
    <title>set 태그와 remove 태그</title>
</head>
<body>
변수 num1 = ${num1} <br>
변수 num2 = ${num2} <br>
num1 + num2 = ${num1 + num2} <br>
오늘은 ${today} 입니다.

<c:remove var="num1" scope="page" />

<p>
삭제한 후의 num1 = ${num1} <br>
삭제한 후의 num1 + num2 = ${num1 + num2}
</body>
</html>
```

실행결과



test3.jsp : set 태그와 remove 태그, forEach태그

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="map" value="<%= new java.util.HashMap() %>" />
<html>
<head><title>test3.jsp [set태그와 remove, forEach 태그]</title></head>
<body>
<c:set target="{map}" property="name" value="홍길동" />
<c:set target="{map}" property="addr" value="강남구 삼성동" />
<c:set target="{map}" property="tel" value="02-123-4567" />

<UL>
    <LI>변수 map에 저장된 name 값: ${map.name}
    <LI>변수 map에 저장된 addr 값: ${map.addr}
    <LI>변수 map에 저장된 tel 값: ${map.tel}
</UL>

<FONT SIZE="5" COLOR="blue">
<c:forEach var="i" items="{map}">
    ${i.key} = ${i.value}<br>
</c:forEach>
</FONT>

<hr>
<c:set var="tb" value="<%= new java.util.Hashtable() %>" />
<c:set target="{tb}" property="one" value="하나" />
<c:set target="{tb}" property="two" value="둘" />
<UL>
    <LI>변수 tb에 저장된 one 값: ${tb.one}
    <LI>변수 tb에 저장된 two 값: ${tb.two}
</UL>

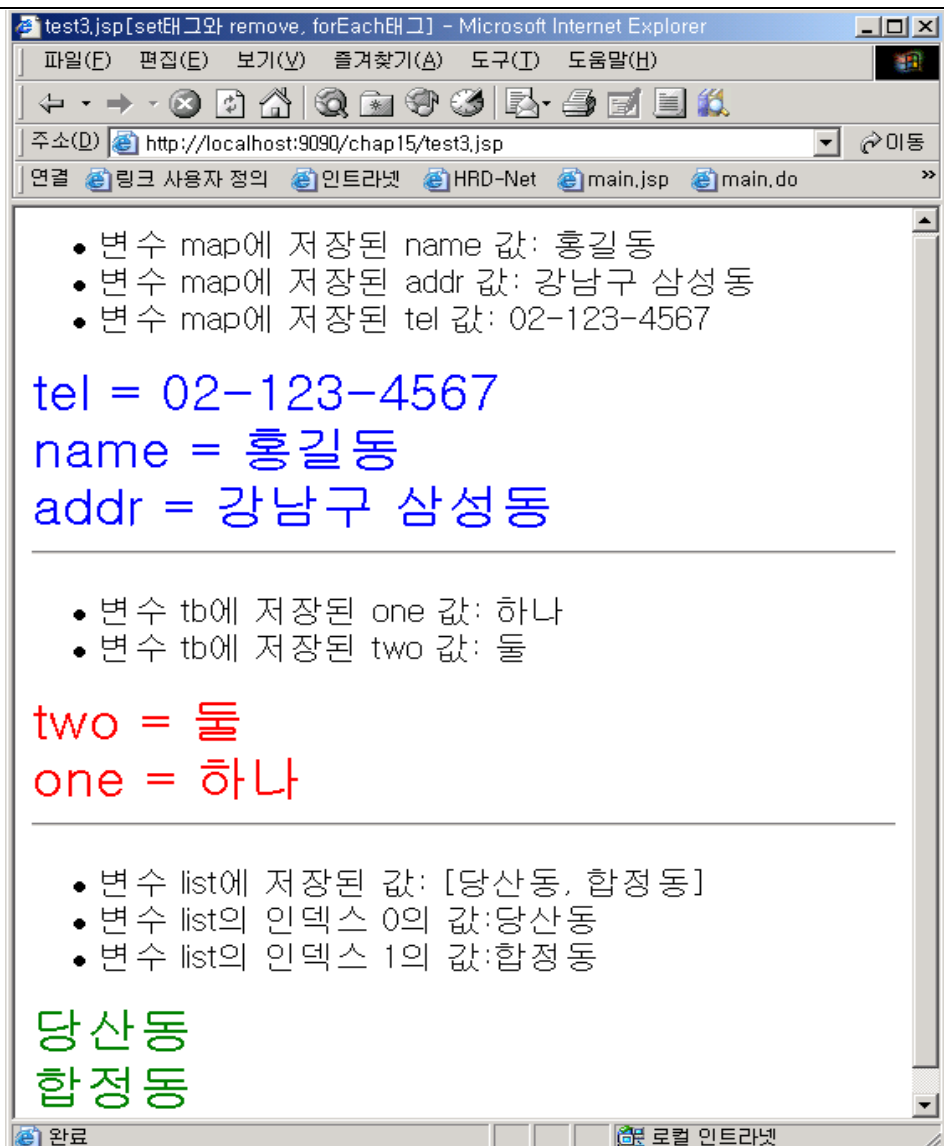
<FONT SIZE="5" COLOR="red">
<c:forEach var="i" items="{tb}">
    ${i.key} = ${i.value}<br>
</c:forEach>
</FONT>
<hr>
<%
java.util.ArrayList arr=new java.util.ArrayList();
arr.add("당산동");
```

```

arr.add("합정동");
%>
<c:set var="list" value="<%=arr%>" />
<UL>
    <LI>변수 list에 저장된 값: ${list}
    <li>변수 list의 인덱스 0의 값:${list[0]}
    <li>변수 list의 인덱스 1의 값:${list[1]}
</UL>
<FONT SIZE="5" COLOR="green">
<c:forEach var="i" items="${list}">
    ${i}<br>
</c:forEach>
</FONT>
</body>
</html>

```

실행결과



7.2. forEach태그

<c:forEach/> 는 강력한 반복실행 태그이다. 형식은 다음과 같다.

Syntax 1: 객체 전체에 걸쳐서 반복

```
<c:forEach [var="varName"] items="collection"
           [varStatus="varStatusName"]
           [begin="begin"] [end="end"] [step="step"]>
    body content
</c:forEach>
```

Syntax 2: 지정한 횟수만큼 반복

```
<c:forEach [var="varName"]
           [varStatus="varStatusName"]
           begin="begin" end="end" [step="step"]>
    body content
</c:forEach>
```

<c:forEach/> 태그는 여러 가지로 활용이 가능하다. 원하는 구간만큼 반복할 수도 있고, 객체를 받아와서 그 객체의 길이만큼 반복할 수도 있다. begin, end 속성은 시작번호와 끝 번호를 지정하고, step 속성을 이용해서 증가 구간을 정할 수 있다. var 속성에서 정한 변수로 반복되는 내부 구간에서 사용할 수 있다. varStatus는 javax.servlet.jsp.jstl.core.LoopTagStatus객체 인스턴스 변수를 만든다. LoopTagStatus객체에는 count라는 프로퍼티가 있어 지금이 몇번째 회전인지 알 수 있다. 즉 자바 for문[예: for(int i=0;i<arr.length;i++){ }]의 i와 같은 것.

7.3. if 태그

<c:if/> 는 흔히 보는 조건문이다. 형식은 다음과 같다.

Syntax 1: Body 없는 경우

```
<c:if test="testCondition"
    var="varName" [scope="{page|request|session|application}"]/>
```

Syntax 2: Body 있는 경우

```
<c:if test="testCondition"
    [var="varName" ] [scope="{page|request|session|application}"]>
    body content
</c:if>
```

<c:if/> 에서 나온 결과를 varName 변수에 넣고, 나중에 활용이 가능하다. 변수의 scope 는 임의로 지정할 수 있고, 생략될 경우 기본값은 page 이다

test4.jsp : forEach태그

[illegible]

```

        </c:forEach>
    </tr>
</c:forEach>
</table>

<h4>int형 배열</h4>

<c:forEach var="i" items="${intArray}" begin="2" end="4">
    [${i}]
</c:forEach>

<h4>Map</h4>

<c:forEach var="i" items="${map}">
    ${i.key} = ${i.value}<br>
</c:forEach>

</body>
</html>

```

실행결과

forEach 태그 - Microsoft Internet Explorer

주소(D) http://localhost:9090/chap15/forEachTag.jsp

연결 링크 사용자 정의 인트라넷 HRD-Net main.jsp main.do 서버를 찾지 못하였습니다.

1부터 100까지 짝수의 합

결과 = 2550

구구단: 5단

5 * 1 = 4 5 * 2 = 8 5 * 3 = 12 5 * 4 = 16 5 * 5 = 20 5 * 6 = 24 5 * 7 = 28 5 * 8 = 32 5 * 9 = 36

구구단 테이블

2* 1 = 2	3* 1 = 3	4* 1 = 4	5* 1 = 5	6* 1 = 6	7* 1 = 7	8* 1 = 8	9* 1 = 9
2* 2 = 4	3* 2 = 6	4* 2 = 8	5* 2 = 10	6* 2 = 12	7* 2 = 14	8* 2 = 16	9* 2 = 18
2* 3 = 6	3* 3 = 9	4* 3 = 12	5* 3 = 15	6* 3 = 18	7* 3 = 21	8* 3 = 24	9* 3 = 27
2* 4 = 8	3* 4 = 12	4* 4 = 16	5* 4 = 20	6* 4 = 24	7* 4 = 28	8* 4 = 32	9* 4 = 36
2* 5 = 10	3* 5 = 15	4* 5 = 20	5* 5 = 25	6* 5 = 30	7* 5 = 35	8* 5 = 40	9* 5 = 45
2* 6 = 12	3* 6 = 18	4* 6 = 24	5* 6 = 30	6* 6 = 36	7* 6 = 42	8* 6 = 48	9* 6 = 54
2* 7 = 14	3* 7 = 21	4* 7 = 28	5* 7 = 35	6* 7 = 42	7* 7 = 49	8* 7 = 56	9* 7 = 63
2* 8 = 16	3* 8 = 24	4* 8 = 32	5* 8 = 40	6* 8 = 48	7* 8 = 56	8* 8 = 64	9* 8 = 72
2* 9 = 18	3* 9 = 27	4* 9 = 36	5* 9 = 45	6* 9 = 54	7* 9 = 63	8* 9 = 72	9* 9 = 81

int형 배열

[3] [4] [5]

Map

name = 임혁정
today = Wed Aug 22 14:24:55 KST 2007

완료 로컬 인트라넷

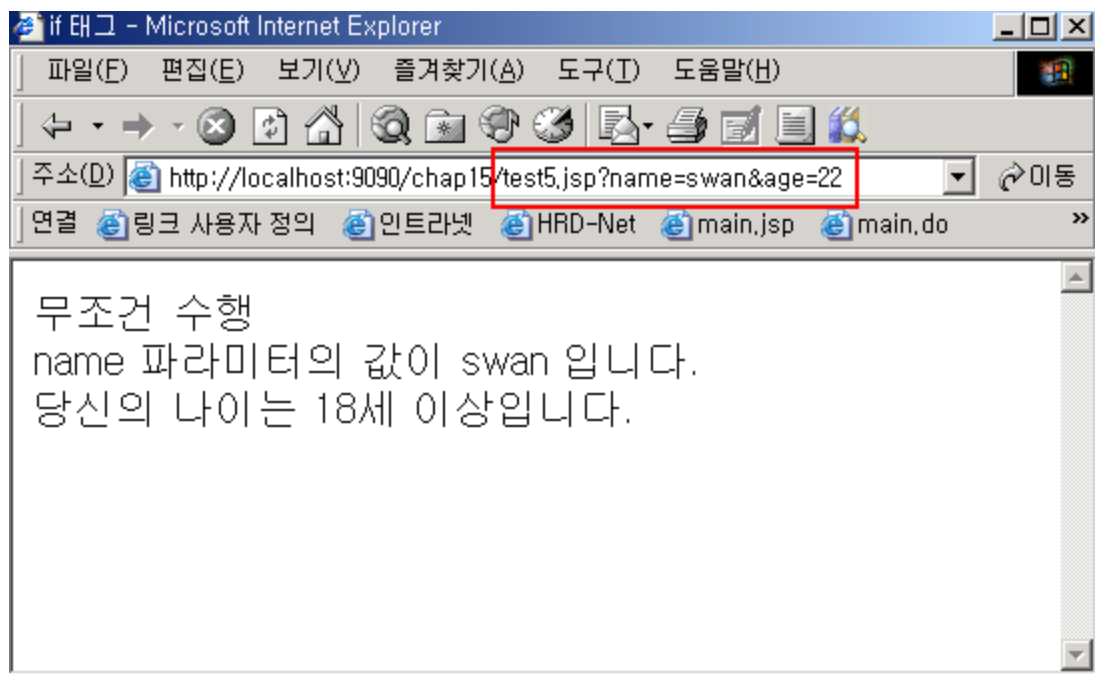
test5.jsp : if태그

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head><title>if 태그</title></head>
<body>
<c:if test="true">
무조건 수행<br>
</c:if>

<c:if test="${param.name == 'swan'}">
name 파라미터의 값이 ${param.name} 입니다.<br>
</c:if>

<c:if test="${18 < param.age}">
당신의 나이는 18세 이상입니다.
</c:if>
</body>
</html>
```

실행결과



7.4. choose ~when ~otherwise 태그

<c:choose/> 태그는 java 의 switch 문과 같지만, 조건에 문자열 비교도 가능하고 쓰임의 범위가 넓다. 또한 <c:if/> 태그에 else 가 없기 때문에 이의 대체 기능도 수행한다. 형식은 다음과 같다.

test6.jsp : choose 태그

```
<c:choose>
body content
(하나 이상의 <when> 과 하나 이하의 <otherwise> 서브태그)
    <c:when test="조건">
        body content
    </c:when>

    <c:otherwise>
        conditional block
    </c:otherwise>
</c:choose>
```

조건 판단을 수행하는 간단한 예제이다.

예제 5. test6.jsp

```
<% response.setContentType("text/html"); %>
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c"%>
<h3>조건</h3>
파라미터 없음:<c:out value="${empty param.name}" />
<h4>&lt;c:if test=""></h4>
<c:if test="${empty param.name}">
<form>
이름을 적어주세요.<br>
    <input type="text" name="name">
    <input type="submit" value="확인">
</form>
</c:if>
<c:if test="${!empty param.name}">
    안녕하세요. <c:out value="${param.name}" />님.
</c:if>

<h4>&lt;c:choose> &lt;c:when test=""> &lt;c:otherwise></h4>
```

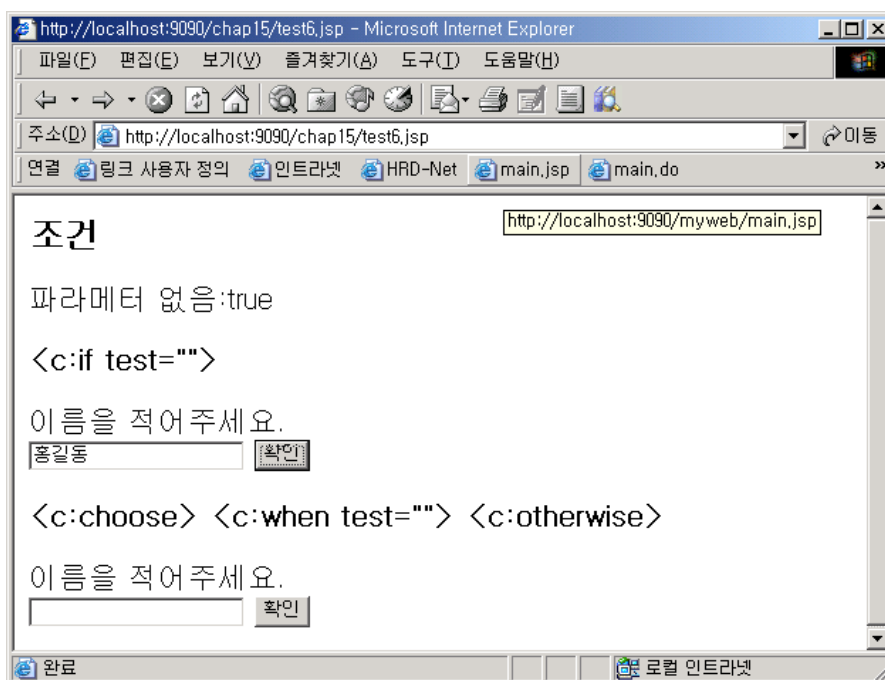


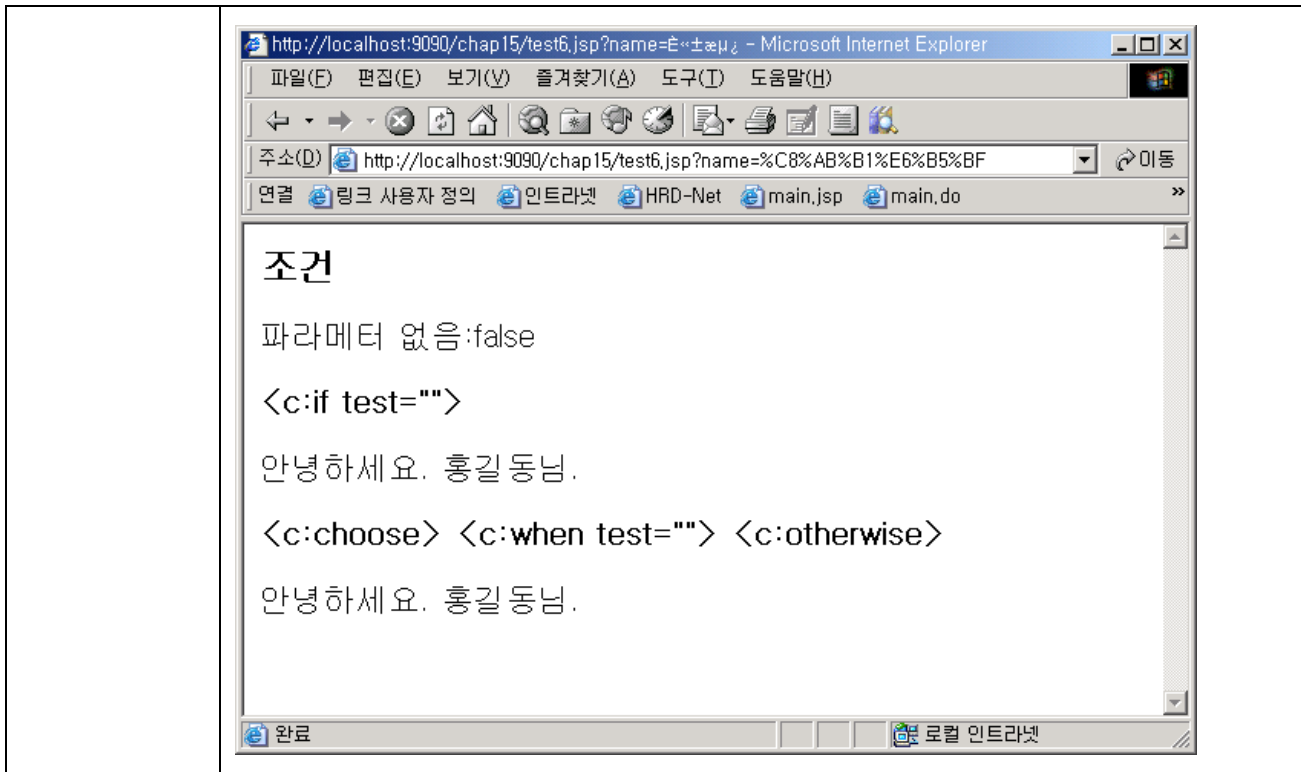
```

<c:choose>
  <c:when test="${empty param.name}">
    <form>
      이름을 적어주세요.<br>
      <input type="text" name="name">
      <input type="submit" value="확인">
    </form>
  </c:when>
  <c:when test="${param.name=='admin'}">
    안녕하세요. 관리자님.
  </c:when>
  <c:otherwise>
    안녕하세요. <c:out value="${param.name}"/>님.
  </c:otherwise>
</c:choose>

```

실행결과





파라미터 name 값이 없는 경우 입력 폼을 출력한다. 파라미터 name 의 값이 "admin"일 경우 관리자를 표시하고, 그 외에는 파라미터 값을 그대로 출력한다.
파라미터의 유무는 empty 와 !empty 연산자를 통해서 확인할 수 있다.

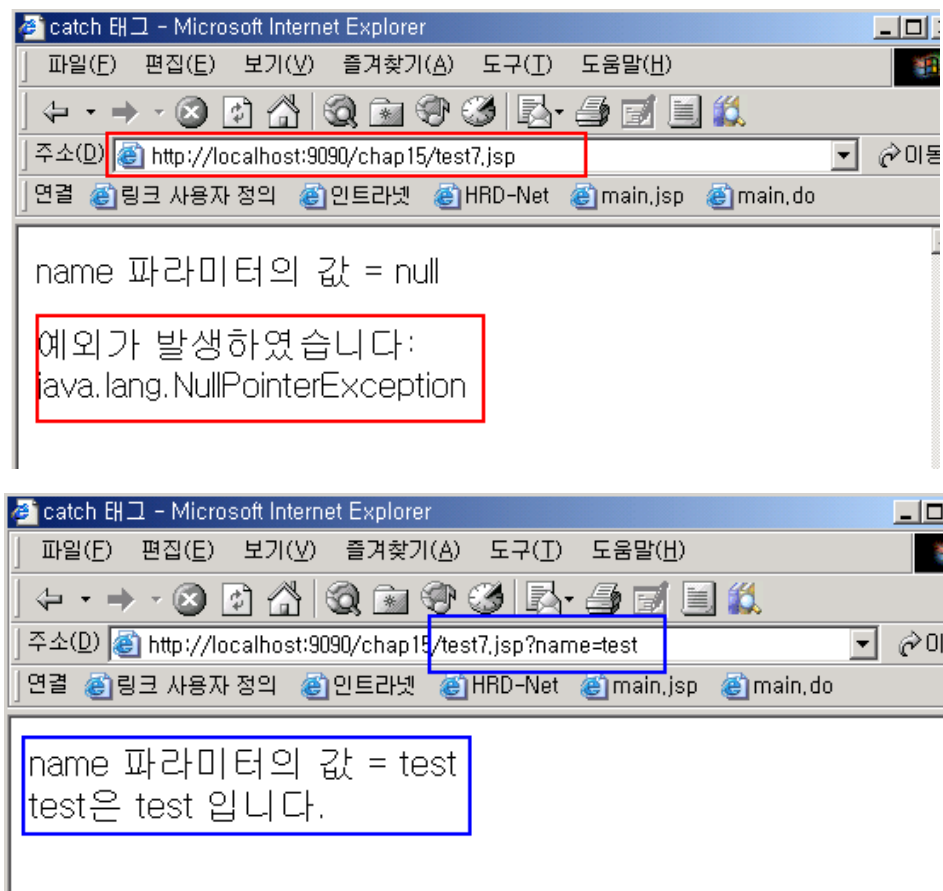
test7.jsp : catch 태그

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head><title>catch 태그</title></head>
<body>

<c:catch var="ex">
name 파라미터의 값 = <%= request.getParameter("name") %><br>
<% if (request.getParameter("name").equals("test")) { %>
${param.name}은 test 입니다.
<% } %>
</c:catch>
<p>
<c:if test="${ex != null}">
예외가 발생하였습니다:<br>
${ex}
</c:if>
```

```
</body>  
</html>
```

실행결과



test8.jsp : redirect 태그

```

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:redirect url="../index.jsp">
    <c:param name="name" value="swan" />
</c:redirect>

```

실행결과

