# 7. 표현 언어(Expression Language)와 JSTL(JSP Standard Tag Library)

# jstl.jar/standard.jar 라이브러리 설치

http://jakarta.apache.org

Tomcat

Apache Taglib

Last Published: 26 February 2015

ApacheCon | Apache | Tom

**Apache Taglibs**
- Home
- Tutorial
- Building/Source
- News Archives

**Our Taglibs**
- RDC
- Standard

**Get Involved**
- Security Issues
- Mailing Lists
- Bugs

**ASF**
- License
- Sponsorship
- Thanks

## Standard Taglib

### JSP(tm) Standard Tag Library implementations

Apache hosts the Apache Standard Taglib, an implementation of the JSP Standard Tag Library (JSTL) specification. Various versions are available.

| Version | JSTL version | Requirements | Getting the Taglib |
|---|---|---|---|
| Standard 1.2.3 | JSTL 1.2 | Servlet 2.5, JavaServer Pages 2.1 | download (javadoc) |
| Standard 1.1 | JSTL 1.1 | Servlet 2.4, JavaServer Pages 2.0 | download |
| Standard 1.0 | JSTL 1.0 | Servlet 2.3, JavaServer Pages 1.2 | download |

# Index of /dist/jakarta/taglibs/standard

| Name | Last modified | Size | Descr |
|---|---|---|---|
| Parent Directory | | - | |
| binaries/ | 2005-10-05 20:39 | - | |
| source/ | 2005-10-05 20:38 | - | |

jakarta-taglibs-standard-1.1.2.tar.gz

jakarta-taglibs-standard-1.1.2.tar.gz

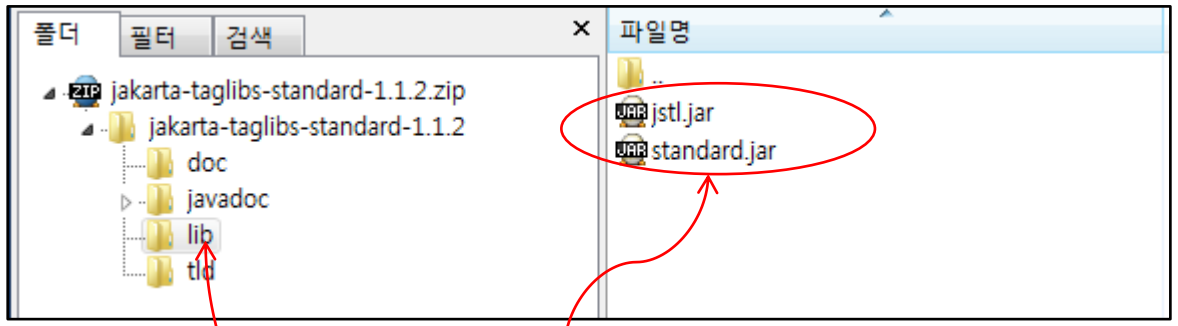
jakarta-taglibs-standard-1.1.2.zip 2004-10-25 20:57 933K

jakarta-taglibs-standard-1.1.2.zip.asc 2004-10-25 20:57 186

다운로드후 압축푼다

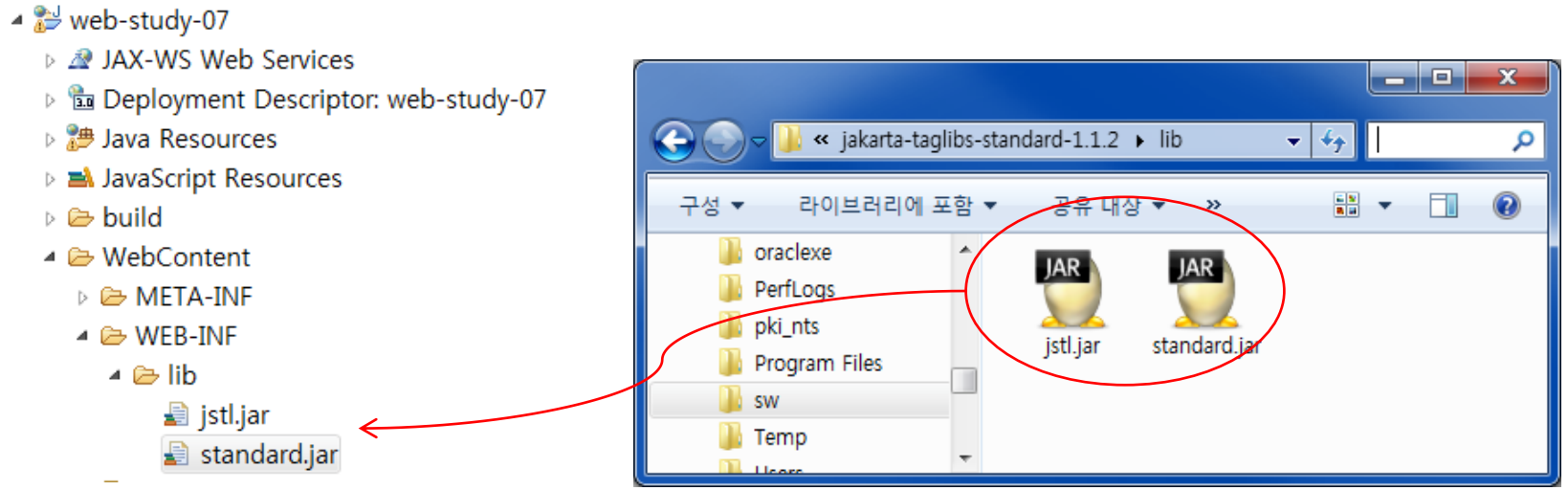
# jstl.jar/standard.jar 라이브러리 설치

압축 푼 결과




jakarta-taglibs-standard-1.1.2/lib
디렉터리에서 jstl.jar와 standard.jar 파일 확인

웹프로젝트>webapp>WEB-INF> 폴더에 복사



# 표현 언어(Expression Language)

JSP 출력에 대한 부분을 쉽게 하기 위해 개발한 tag(태그)

■ 예 : 표현식

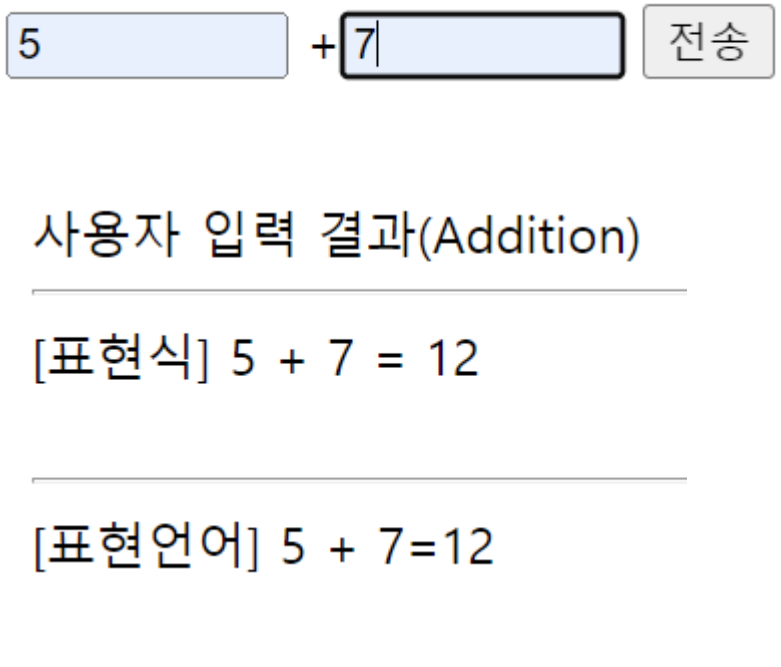
```
<%=expr%>
```

■ 예 : 표현 언어

```
${expr}
```

# 표현 언어로 요청 파라미터 처리하기 1

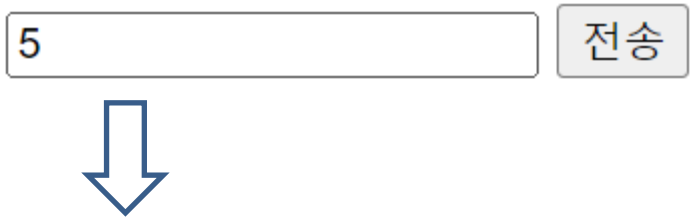
**request.getParameter("num") vs ${param.num}**




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

%>
[표현식]
<%=num1 %> + <%=num2 %> = <%=num1+num2 %><br>

<hr>

[표현언어]
${param.num1} + ${param.num2 }=${param.num1}
```

# 표현 언어(Expression Language)

사용자 입력값 -> 서블릿 처리 -> JSP 출력




EvenOdd.java Controller(로직 담당)

```
int num = Integer.parseInt(request.getParameter("num"));
String result = "홀수";
if(num%2 != 0) {
   result = "홀수";
}else {
   result = "짝수";
}
request.setAttribute("result", result);
RequestDispatcher disp =
request.getRequestDispatcher("view.jsp");
disp.forward(request, response);
```

add.jsp View

홀수입니다.
홀수

```
<%=request.getAttribute("result") %>입니다.<br>
${result}<br>
<hr>
```

# 표현 언어(Expression Language)

사용자 정보 서블릿 처리 -> jsp출력하기

```
request.setAttribute("name", "홍길동");
request.setAttribute("age", 22);
RequestDispatcher disp = request.getRequestDispatcher("view.jsp");
disp.forward(request, response);
```

```
<%=request.getAttribute("name") %>/<%=request.getAttribute("age") %>입니다.
<br>
${name}/${age }<br >
```

# 표현 언어 장점

표현 언어에서는 파라미터를 찾지 못하면 공백으로 처리한다.

=>예외 발생하지 않음

JAVA 언어에서는 파라미터를 찾지 못하면 NullPointException 오류 발생

# 표현 언어로 내장 객체 접근

| JSP 내장 객체 | 표현 언어의 내장 객체 |
|---|---|
| pageContext | pageScope |
| request | requestScope |
| session | sessionScope |
| application | applicationScope |

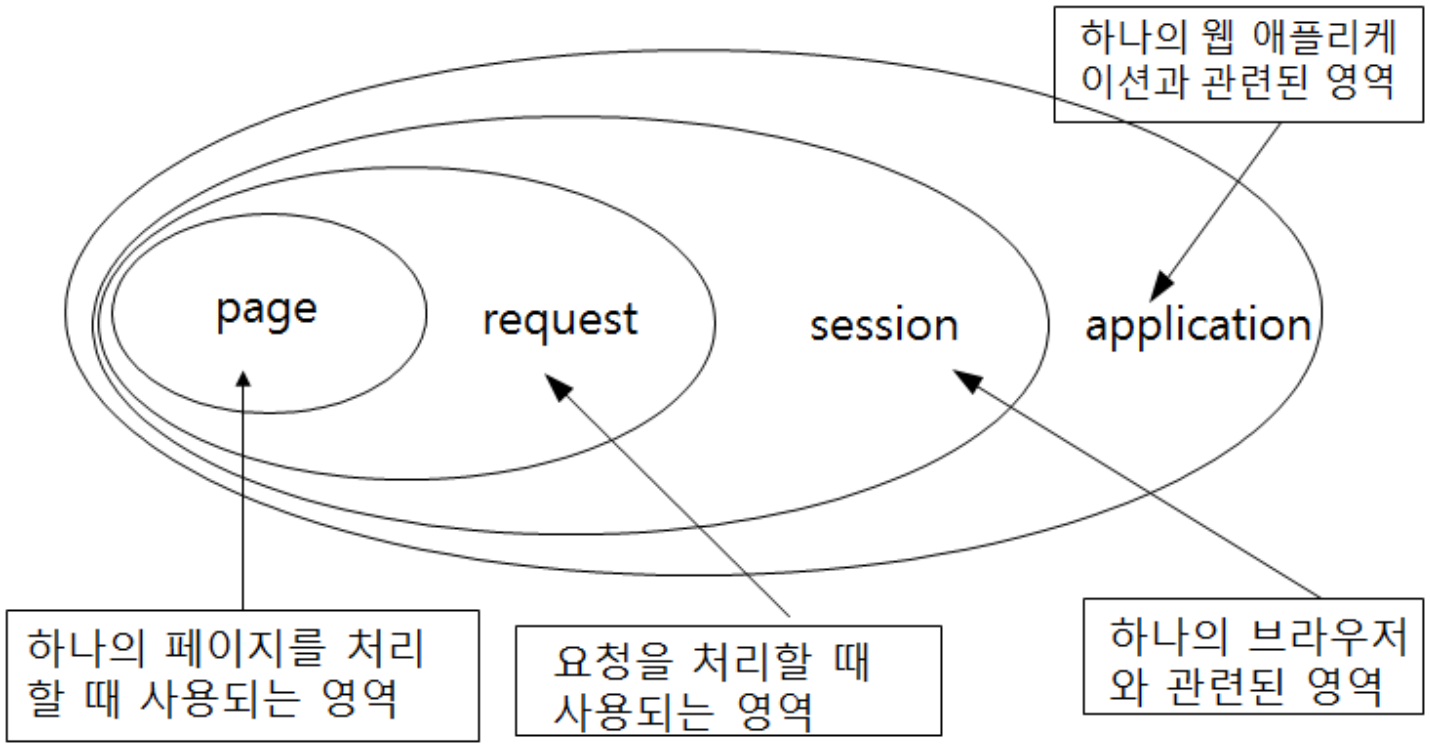
내장 객체 접근 메소드

request.getAttribute("num1");　　${requestScope.num1}

session.getAttribute("num1");　　${sessionScope.num1}

# 표현 언어로 내장 객체 접근: ${name} 출력 결과는?

```
pageContext.setAttribute("name", "page man");
request.setAttribute("name", "request man");
session.setAttribute("name", "session man");
application.setAttribute("name", "application man");
```



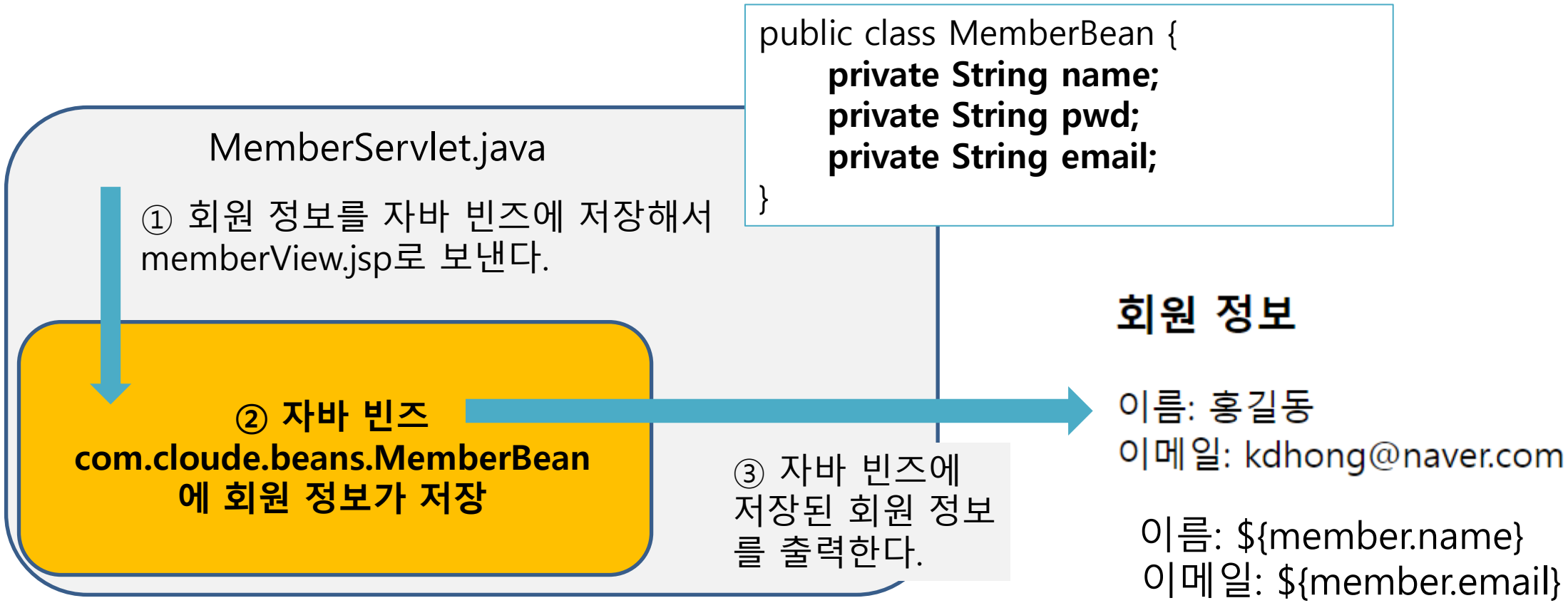
# 표현 언어로 내장 객체 접근

```
pageContext.setAttribute("name", "page man");
request.setAttribute("name", "request man");
session.setAttribute("name", "session man");
application.setAttribute("name", "application man");
```

모든 객체에 저장된 name을 출력하려면?
구체적인 범위 지정한다

예)  ${sessionScope.name}

# 표현 언어로 회원 자바 빈 정보 저장 및 조회



# JSTL

JSP Standard Tag Library의 약어

JSP에서 사용 가능한 표준 태그 라이브러리

JSTL에서 제공하는 태그를 사용하면 JSP 코드가 깔끔하고 가독성 향상됨

| 기능 | prefix | 기본 URI |
|---|---|---|
| 기본 기능 | c | http://java.sun.com/jsp/jstl/core |
| 형식화 | fmt | http://java.sun.com/jstl/fmt |
| 데이터베이스 작업 | sql | http://java.sun.com/jstl/sql |
| XML 처리 | x | http://java.sun.com/jstl/xml |
| 함수 처리 | fn | http://java.sun.com/jsp/jstl/fn |

# JSTL

- JSP 문서에서 태그 라이브러리 사용법

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"  prefix="c"%>
```

  - uri 속성 값: 어떤 라이브러리를 사용하는지를 구별
  - prefix 속성 값: 태그에서 사용할 접두사

예) **<c:out value="Hello World!"/>  ➔  <%="Hello World"%>**

# JSTL core 라이브러리의 종류

| 태그 | 설명 |
| --- | --- |
| <c:set> | 변수에 값 설정 |
| <c:remove> | 변수에 설정된 값 제거 |
| <c:if> | 조건문 |
| <c:choose> | 조건이 여러 개 일 경우 사용 -> switch 문과 유사 |
| <c:forEach> | 반복 처리문 |
| <c:forTokens> | 구분자로 분리된 각각의 토큰을 처리할 때 사용 |
| <c:Import> | 외부의 자원을 url을 지정하여 가져다 사용 |
| <c:redirect> | 지정한 경로로 이동 |
| <c:url> | url을 재작성 |
| <c:out> | 데이터를 출력할 때 사용하는 태그로 표현식인 <%= %>로 대체 가능 |
| <c:catch> | 예외처리에 사용 |

# 요청 파라미터에 한글 포함할 경우

이름: 홍길동 SEND

이름: íOOê„ëOO

```
<h3>한글 처리(기존 방식)</h3>
<%
    request.setCharacterEncoding("UTF-8");
%>
<%=request.getParameter("name") %>
<hr>
<h3>한글 처리(EL 방식)</h3>
<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>
<fmt:requestEncoding value="UTF-8"/>
이름:${param.name }
```

**한글 처리(기존 방식)**

홍길동

---

**한글 처리(EL 방식)**

이름: 홍길동

# <c:forEach>태그

배열(Array)이나 컬렉션(Collection) 또는 맵(Map) 등과 같은 집합체에 저장되어 있는 값들을 순차적으로 처리할 때 사용할 수 있는 태그

**<c:forEach var="속성이름" items="집합체 이름" varStatus="반복상태속성이름">**

```
String [] fruits = {"apple", "banana", "kiwi"};
request.setAttribute("fruits", fruits);
RequestDispatcher disp = request.getRequestDispatcher("view.jsp");
disp.forward(request, response);
```

ArrayEx.java
(서블릿 코드)

```
<c:forEach var="fruit" items="${fruits }" varStatus="status">
${status.count }. ${fruit }
</c:forEach>
```

1. apple 2. banana 3. kiwi

view.jsp

# <c:forEach>태그 속성

| 프로퍼티 | 설명 |
|---|---|
| index | items에 지정한 집합체의 현재 반복 중인 항목의 index를 알려준다. 0부터의 순서가 부여된다. |
| count | 루핑을 돌 때 현재 몇 번째를 반복 중인지 알려준다. 1부터의 순서가 부여된다. |
| first | 현재 루프가 처음인지 여부를 알려준다. 첫 번째일 경우에는 true를 아니면 false를 되돌린다. |
| last | 현재 루프가 마지막인지 여부를 알려준다. 마지막일 경우에는 true를 아니면 false를 되돌린다. |
| begin | 반복에 사용될 것 중 첫 번째 항목의 Index |
| end | 반복에 사용될 것 중 마지막 항목의 Index |
| step | 증가치 |

```
<c:forEach var="cnt" begin="1" end="10" >
   ${cnt}
</c:forEach>
```

```
<fmt:requestEncoding value="UTF-8"/>
<fieldset><legend>사용자 정보</legend>
<table style="width:500px;border: 1px solid;">
<tr>
<th>이름</th><th>나이</th><th>성별</th><th>좋아하는 음식</th><th>관심분야</th>
</tr>
<tr>
    <td>${param.name}</td>
    <td>${param.age}</td>
    <td>${param.gender}</td>
    <td>${param.food}</td>
    <td style="text-align:center;">
<c:forEach items="${paramValues.interest}" var="interest">
    ${interest }<br>
</c:forEach>
</td>
</tr>
</table>
</fieldset>
```

# <c:if>태그: else 태그가 없음

```
<c:if test="${param.userType == 'admin'}">
   ${param.id}(관리자)
</c:if>
<c:if test="${param.userType == 'member'}">
  ${param.id}(회원)
</c:if>
```

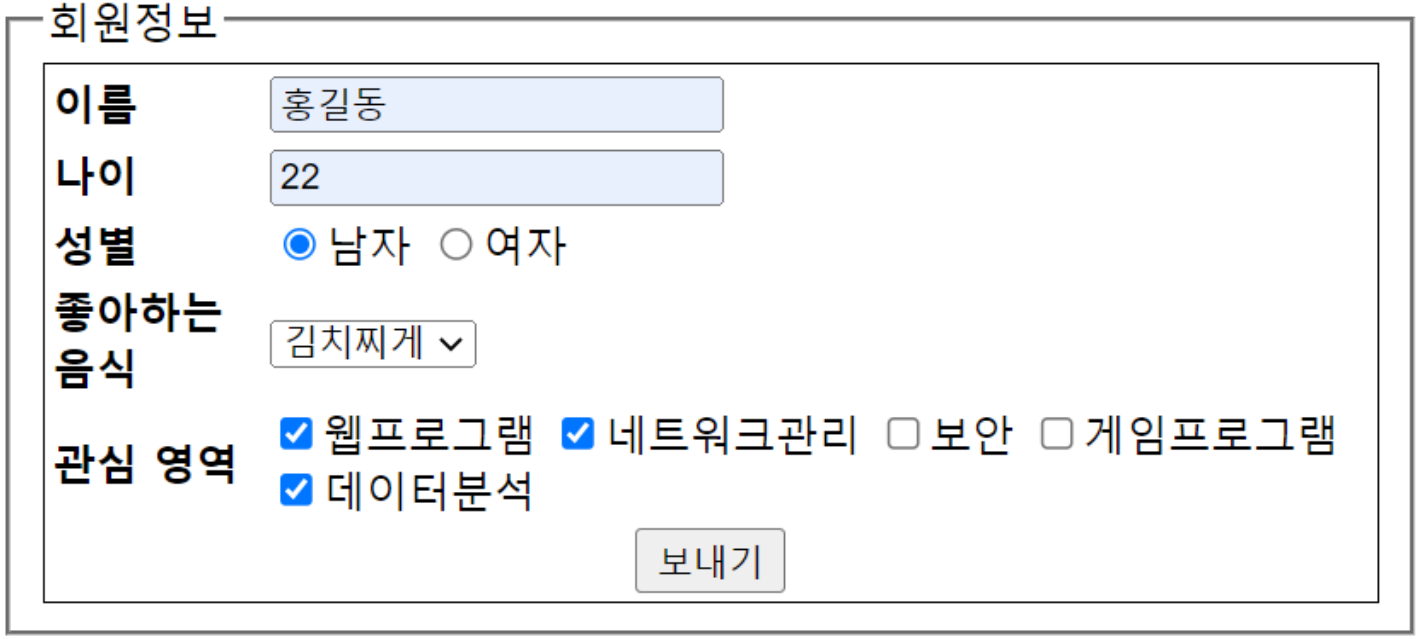
```
if(request.getParameter("userType").equals("admin")){
    out.print(request.getParameter("id")+"(관리자)");
}else{
    out.print(request.getParameter("id")+"(회원)");
}
```

# <c:choose>, <c:when>, <c:otherwise>태그: 자바의 switch 구문과 유사

```
<c:if test="${param.userType == 'admin'}">
   ${param.id}(관리자)
</c:if>
<c:if test="${param.userType == 'member'}">
  ${param.id}(회원)
</c:if>
```

```
<c:choose>
    <c:when test="${param.userType == 'admin'}">
        ${param.id}(관리자)
    </c:when>
    <c:otherwise>
        ${param.id}(회원)
    </c:otherwise>
</c:choose>
```

# 실습: 표현 언어 이용하여 전송데이터 처리하기




사용자 정보

| 이름 | 나이 | 성별 | 좋아하는 음식 | 관심분야 |
|---|---|---|---|---|
| 홍길동 | 22 | 남자 | 김치찌게 | 웹프로그램<br>네트워크관리<br>데이터분석 |