

JSP Programming

표현 언어(Expression Language)



EL(Expression Language)

카페

- 표현 언어란 "자바 언어를 사용하지 않고 JSP 페이지를 작성하겠다."라는 취지로 나온 개념이다.

항목	설 명
기본 개념(사상)	JSP의 기본 문법을 보완하는 스크립트 언어이다. 간결한 소스 작성을 위해 탄생한 개념이다. 자바스크립트와 XPath를 베이스로 설계되었다. Xpath : XML 문서의 정보에 접근하기 위해 사용되는 언어이다
특징	EL 자체의 내장 객체가 제공된다. 연산자가 제공(산술 연산과 비교 연산등이 가능)된다. 자바스크립트와 비슷한 방법으로 객체 내부의 자원에 접근이 가능하다. dot(.) 과 bracket([]) 모두를 이용해서 접근할 수 있다. JSP 2.0 스펙에 포함이 되었다. JSP의 영역(page, request, session, application)에 저장된 어떤 속성(attribute) 및 Java Bean에서도 EL의 변수로서 사용 가능하다. null 값을 갖는 변수에 대해 좀더 관대하다. JSP 스크립트릿 내부에서는 사용 불가능하다.
표현 방법	\$으로 시작한다. {...} 사이에 어떤 변수를 표현한다.

EL의 표현 방법

Syntax

`${expr}`

expr – 표현 언어가 정의한 문법에 따라 값을 표현하는 수식이다.

Example

`${test}` //변수 test의 값을 출력한다.

//skin이라는 객체의 id를 읽어 온다.

```
<jsp:include page="/module/${skin.id}/header.jsp" flush="true" />
```

//세션 영역의 member 객체의 id를 읽어 온다.

```
<b>${sessionScope.member.id}</b>님 환영합니다.
```

`${header.cookie}` 와 `${header['cookie']}` 는 같은 결과 값이다.

스크립트 요소(스크립트릿, 표현식, 선언문)를 제외한 나머지 부분에서 사용 가능하다.
표현식을 통해서 **간결한 표현**이 가능해진다.

표현 언어(EL) 맛보기

카페

- 파일 이름 : helloMessage.jsp
- 표현 언어를 이해하기 위하여 간단한 메시지를 출력하는 예제를 하나 보도록 하자.
- 다음 예제는 다양한 방법으로 메시지를 출력하고 있다.

출력 결과

표현 언어 방식 : 안녕하세요
표현식 : 안녕하세요
out 내장 객체 : 안녕하세요.

```
<body>
  <!-- ${} 사이에 쌍따옴표를 이용하여 문자열을 출력하고 있다. -->
  표현 언어 방식 : ${"안녕하세요"} <br>
  <!-- 표현 식을 이용하여 문자열을 출력하고 있다. -->
  표현식 : <%= "안녕하세요" %> <br>
  out 내장 객체 :
  <%
    //out 내장 객체를 이용하여 문자열을 출력하고 있다.
    out.println("안녕하세요.");
  %>
</body>
```

EL의 데이터 타입

카페

데이터 타입	설 명
Boolean	true 와 false
정수 타입	0~9로 이루어진 정수 값 음수의 경우 '-'가 붙음
실수 타입	0~9로 이루어져 있으며, 소수점('.')을 사용할 수 있고, 3.24e3과 같이 지수형으로 표현 가능.
문자열 타입	따옴표(' 또는 ")로 둘러싼 문자열. 만약 작은 따옴표(')를 사용해서 표현할 경우 값에 포함된 작은 따옴표는 \' 와 같이 \ 기호와 함께 사용해야 한다. \ 기호 자체는 \\ 로 표시한다.
널 타입	null

EL 연산자

카페

연산자	설명
.	빈 또는 맵에 접근하기 위한 연산자
[]	배열 또는 리스트에 접근하기 위한 연산자
()	연산자의 우선 순위 변경시 사용
+ -	덧셈/뺄셈
* /(div) %(mod)	곱셈/나눗셈(division)/나머지
x ? a : b	삼항 연산자
empty	값이 Null이면 True을 리턴한다.
or	논리 연산자(논리합)
&& and	논리 연산자(논리곱)
!(not)	부정
== 또는 eq	두 항이 같으면 True(equal)
!= 또는 ne	두 항이 다르면 True(not equal)
< 또는 lt	[보다 작다]의 의미(less than)
> 또는 gt	[보다 크다]의 의미(greater than)
<= 또는 le	[보다 작거나 같다]의 의미(less than or equal)
>= 또는 ge	[보다 크거나 같다]의 의미(greater than or equal)

연산자 사용 예시

항목	설 명
문자열 + 숫자	객체를 숫자 값으로 변환 후 연산자를 수행 <code>\${"10"}+1</code> → <code>\${10+1}</code>
숫자 변환 불가능 문자 + 숫자	에러 발생 <code>\${"열"}+1</code> → 에러
수치 연산에서의 null	수치 연산자에서 사용되는 객체가 null이면 0으로 처리: <code>\${null + 1}</code> → <code>\${0+1}</code>
문자열 비교	<code>\${str == '값'}</code> 은 <code>str.compareTo("값") == 0</code> 과 동일
<code>==</code> 연산자	성별이 남자인지 여자인지 판단하여 해당 성별에 체크하기 위한 문장 <code><input type="radio" name="gender" value="남자" checked="\${bean.gender == '남자'}">남자</code> <code><input type="radio" name="gender" value="여자" checked="\${bean.gender == '여자'}">여자</code>
jQuery를 이용한 콤보 박스에서의 항목 선택	<pre>\$(document).ready(function(){ var cate = '\${bean.category}'; /* 상품의 카테고리를 읽어 와서 */ \$('#category').val(cate); /* 해당하는 상품의 카테고리를 선택한다 . */ });</pre>

empty 연산자

- empty 연산자는 부정적인 개념인 경우 True인 값을 리턴해준다.
- 일반적으로 다음과 같은 경우에 True를 반환한다.
 - <값>이 null이면 true를 리턴한다.
 - <값>이 빈 문자열("")이면 true를 리턴한다.
 - <값>이 길이가 0인 배열이면 true를 리턴한다.
 - <값>이 빈 Map이면 true를 리턴한다.
 - <값>이 빈 Collection이면 true를 리턴한다.
 - 이 외의 경우에는 false를 리턴한다.

empty 연산자, 조건 연산자

Syntax

empty <값>

Syntax

<수식> ? <값1> : <값2>

<수식>의 결과값이 true이면 <값1>을 리턴하고, false이면 <값2>를 리턴

표현 언어(EL) 연산자

카페

- 파일 이름 : elOperator.jsp
- EL에서는 변수도 표현하지만, **연산자**와 **내장 객체**도 제공하고 있다.
- EL도 일종의 스크립트 언어로서
- 자료 타입, 수치 연산자, 논리 연산자,
- 관계 연산자 등을 제공하고 있다.
- EL의 언어적인 측면과 규칙들을 숙지하기
- 위하여 EL에 대한 **기본 연산자**를
- 익히도록 하자.

```
<body>
  <h3>EL 연산자 실습</h3>
  <!-- 덧셈 연산을 수행한다. -->
  \${14+5} : \${14+5} <br>
  \${8-3} = \${8-3} <br>
  \${6*3} = \${6*3} <br>

  <!-- 자바와는 다르게 14/5 = 2.8이 된다 -->
  \${14/5} : \${14/5} <br>
  \${14 mod 5} : \${14 mod 5}<br>
  \${14 > 5} : \${14 > 5} <br>
  \${10==9} = \${10==9} <br>
  \${5 gt 10} : \${5 gt 10}<br>

  <!-- gt는 greater than의 의미로 > 연산자를 의미한다. -->
  \${(14 > 5) ? 14 : 5} : \${(14 > 5) ? 14 : 5}<br>

  <!-- 논리 연산자 및 관계 연산자 모두 가능하다. -->
  \${(14 > 5) || (5 < 10)} : \${(14 > 5) || (5 < 10)}<br>
  <%
    String input=null;
  %>
  <!-- null인 값에 대하여 empty 연산자를 사용하게 되면 true를 반환한다. -->
  \${empty input} : \${empty input}<br>
</body>
```

출력 결과

EL 연산자 실습

```
\${14+5} : 19
\${8-3} = 5
\${6*3} = 18
\${14/5} : 2.8
\${14 mod 5} : 4
\${14 > 5} : true
\${10==9} = false
\${5 gt 10} : false
\${(14 > 5) ? 14 : 5} : 14
\${(14 > 5) || (5 < 10)} : true
\${empty input} : true
```

EL의 기본 객체

카페

기본 객체	설명
pageContext	JSP의 page 기본 객체와 동일하다.
pageScope	pageContext 기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map 객체.
requestScope	request 기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map 객체.
sessionScope	session 기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map 객체.
applicationScope	application 기본 객체에 저장된 속성의 <속성, 값> 매핑을 저장한 Map 객체.
param	요청 파라미터의 <파라미터이름, 값> 매핑을 저장한 Map 객체. 파라미터 값의 타입은 String 으로서, <code>request.getParameter(이름)</code> 의 결과와 동일하다.
paramValues	요청 파라미터의 <파라미터이름, 값배열> 매핑을 저장한 Map 객체. 값의 타입은 String[] 으로서, <code>request.getParameterValues(이름)</code> 의 결과와 동일하다.
header	요청 정보의 <헤더이름, 값> 매핑을 저장한 Map 객체. <code>request.getHeader(이름)</code> 의 결과와 동일하다.
headerValues	요청 정보의 <헤더이름, 값 배열> 매핑을 저장한 Map 객체. <code>request.getHeaders(이름)</code> 의 결과와 동일하다.
cookie	<쿠키 이름, Cookie> 매핑을 저장한 Map 객체. <code>request.getCookies()</code> 로 구한 Cookie 배열로부터 매핑을 생성한다.
initParam	초기화 파라미터의 <이름, 값> 매핑을 저장한 Map 객체. <code>application.getInitParameter(이름)</code> 의 결과와 동일하다.

사용 예시

카페

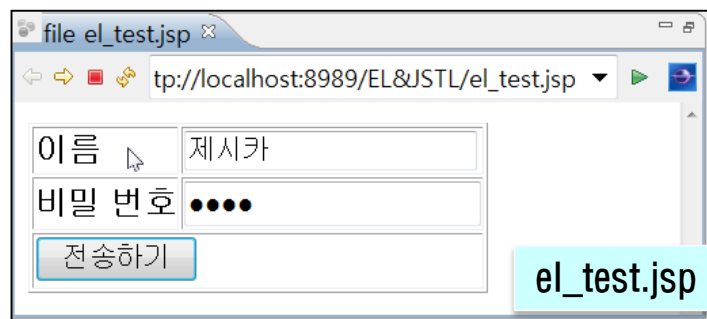
항목	설명
JSP	<code><%=session.getAttribute("member").getName()%></code> 표현식을 이용한 member 이라는 객체의 getName() 메소드를 사용하는 예시
JSTL	<code><c:out value="\${sessionScope.member.name}" /></code> 태그 라이브러리를 이용하여 member이라는 객체의 name이라는 프로퍼터를 이용
EL	\$로 시작하고 { ... } 사이에 표현하고자 하는 데이터 표시 \${member.name}의 의미는 member이라는 객체 내의 name 변수(프로퍼티)를 읽어 온다.

EL 사용 예시	JSP에서의 사용 방법
<code>\${requestScope.name}</code>	request 영역의 값을 출력한다. <code><%=request.getAttribute("name")%></code>
<code>\${sessionScope.profile}</code>	<code>Object xxx = session.getAttribute("profile");</code>
<code>\${param.productId}</code>	<code>request.getParameter("productId");</code>
<code>\${paramValues.hobby}</code>	<code>request.getParameterValues("hobby");</code>

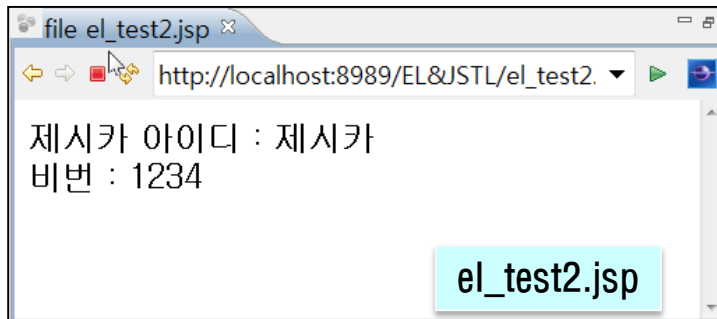
도전 과제(param 내장 객체)

카페

- EL은 스크립트 요소(스크립트릿, 표현식, 선언부)를 제외한 나머지 부분에서 사용 가능하다.
- 표현식을 통해서 **간결한 표현**이 가능해진다.
- EL에 내장되어 있는 객체를 사용하여 다음 문제를 풀어 보도록 하세요.



el_test.jsp



el_test2.jsp

```
<form action="el_test2.jsp" method="post">
  <input type="text" name="name" value="제시카"></td>
  <input type="password" name="password" value="1234">
  <input type="submit" value="전송하기">
</form>
```

```
<body>
  <%=request.getParameter("name") %>
  아이디 : ${param.name }<br>
  비번 : ${param.password }<br>
</body>
```

내장 객체 사용

카페

- EL에 내장되어 있는 객체를 사용한다면 복잡한 자바 코드를 사용하지 않아도 파라미터들에 관한 정보를 쉽게 사용할 수 있다.
- 또한 여러 종류의 scope 내에 존재하는 객체를 쉽게 표현할 수 있다.

출력 결과

회원 정보

아이디	<input type="text" value="jessica"/>
이름	<input type="text" value="제시카"/>
비밀 번호	<input type="password" value="••••"/>
나이	<input type="text" value="30"/>
<input type="button" value="전송하기"/>	

출력 결과

당신이 입력한 정보입니다(고전적인 방식)
제시카
아이디 : jessica
이름 : 제시카
비번 : 1234
나이 : 30

```
<!-- EL의 내장 객체인 param을 이용하여 해당 파라미터의 값을 출력한다. -->
${param.id}<br>
${param.name}<br>
${param.password}<br>
${param.age}<br>
```

파일 이름	설명
elFrom.jsp	회원의 기본적인 정보들을 입력하는 form 페이지이다.
elTo.jsp	요청 정보에 대한 결과를 보여 주는 페이지이다.

자바 빈과 EL

카페

- EL과 JavaBean을 연동하게 되면 코드를 좀 더 효율적으로 작성할 수 있다.

회원 정보 전송

이름	<input type="text" value="제시카"/>
나이	<input type="text" value="20"/>
몸무게	<input type="text" value="70"/>
키	<input type="text" value="175"/>
성별	<input type="text" value="남"/>
<input type="button" value="전송하기"/>	

이름 : 제시카
나이 : 20
몸무게 : 70
키 : 175
성별 : 남

```
package mypkg;

public class MemberBean2 {
    private String name ; //이름
    private int age ; //나이
    private String weight ; //몸무게
    private String height ; //키
    private String gender ; //성별

    //getter, setter 구현
}
```

```
<!-- Bean 클래스 MemberBean을 이용하여 객체 member을 생성한다. -->
<jsp:useBean id="member" class="mypkg.MemberBean2"></jsp:useBean>

<!-- 모든 프로퍼티에 대한 setter을 호출한다. -->
<jsp:setProperty property="*" name="member"/>
```

```
<!-- 이전 방식인 표현식을 이용하여 출력한다. -->
이름 : <%=member.getName()%><br>

<!-- member 객체와 EL을 이용하여 각 정보를 출력한다. -->

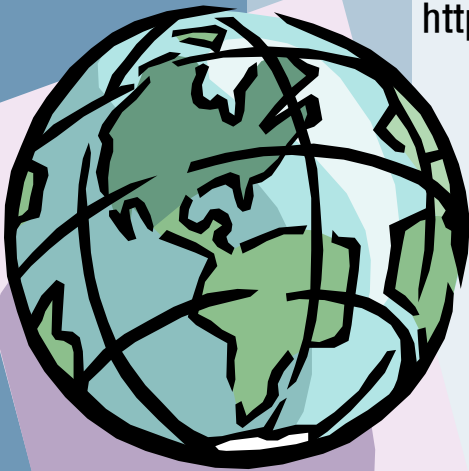
이름 : ${member.name} <br>
나이 : ${member.age} <br>
몸무게 : ${member.weight} <br>
키 : ${member.height} <br>
성별 : ${member.gender} <br>
```

파일 이름	설명
MemberBean2.java	회원의 각 정보들에 대한 변수를 저장하고 있는 Bean 클래스이다.
memberFrom.jsp	회원 정보를 입력 받는 폼 페이지이다.
memberTo.jsp	해당 결과를 보여 주는 페이지이다.

JSP Programming

JSTL (JSP Standard Tag Library)

http://docs.oracle.com/cd/E17802_01/products/products/jsp/jstl/1.1/docs/tlddocs/index.html



JSTL(JSP Standard Tag Library)

카페

항목	설 명
특징	<p>개발자를 위한 표준화된 Custom Tag이다.</p> <p>자바 문법없이 태그만으로 JSP 문장을 작성하는 개념이다.</p> <p>개발자가 가장 필요로 하는 구현 내용을 표준 태그로 제공한다.</p> <p>JSP 코드가 깔끔해지고 가독성을 향상시키고자 하는 목적이다.</p> <p>XML 문법을 준수해야 한다.</p>
역할	<p>데이터베이스 처리등과 관련된 표준 custom tag를 제공</p> <p>JSP 페이지에서 논리적인 판단 및 반복문의 처리(자바의 변수 선언, if, for 구문 등의 로직)</p> <p>다른 JSP 페이지 호출</p> <p>XML 문서의 처리</p> <p>데이터베이스의 입력, 수정, 삭제, 조회 등등</p> <p>날짜, 시간, 숫자 등의 포맷</p> <p>다국어 버전의 JSP 만들기</p> <p>문자열을 처리하는 함수 호출</p>
필요한 jar	<p>http://tomcat.apache.org/taglibs/standard/ 사이트에서</p> <p>jstl.jar, standard.jar 파일을 다운로드하여 추가 요망</p>

JSTL이 제공하는 태그

카페

라이브러리	기능 기능	접두사	관련 URI
코어(core)	변수 지원, 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 (formatting)	지역, 메시지 형식, 숫자 및 날짜 형식	fmt	http://java.sun.com/jsp/jstl/fmt
데이터베이스	데이터베이스의 입력/수정/조회/삭제	sql	http://java.sun.com/jsp/jstl/sql
함수(Functions)	컬렉션 처리, String 처리	fn	http://java.sun.com/jsp/jstl/functions

항목	설명
JSTL 사용	<pre><%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %> <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %> <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %></pre>
태그 라이브러리 지시어 사용	<p>prefix : 어떤 타입의 태그 라이브러리를 작성할 것인가?</p> <p>uri : 실제 존재하고 있는 TLD 파일의 위치 정보</p> <p>TLD 파일 : 커스텀 태그 정보를 담고 있는 라이브러리 파일</p>

JSTL(JSP Standard Tag Library)

카페

- 처리 영역(4개의 커스텀 태그와 문자열을 다루는 함수들)

Core 라이브러리	
일반적인 것	<code><c:out></code> <code><c:set></code> <code><c:remove></code> <code><c:catch></code>
조건	<code><c:if></code> <code><c:choose></code> <code><c:when></code> <code><c:otherwise></code>
URL 관련	<code><c:import></code> <code><c:url></code> <code><c:redirect></code> <code><c:param></code>
반복	<code><c:forEach></code> <code><c:forEachToken></code>

포매팅 라이브러리	
국제화	<code><fmt:message></code> <code><fmt:setLocale></code> <code><fmt:bundle></code> <code><fmt:setBundle></code> <code><fmt:param></code> <code><fmt:requestEncoding></code>
포매팅	<code><fmt:timeZone></code> <code><fmt:setTimeZone></code> <code><fmt:formatNumber></code> <code><fmt:parseNumber></code> <code><fmt:parseDate></code>

함수들 지원

XML 라이브러리	
코어 XML 액션	<code><x:parse></code> <code><x:out></code> <code><x:set></code>
XML 흐름제어	<code><x:if></code> <code><x:choose></code> <code><x:when></code> <code><x:otherwise></code> <code><x:forEach></code>
변환 액션	<code><x:transform></code> <code><x:param></code>

SQL 라이브러리	
데이터베이스 접근	<code><sql:query></code> <code><sql:update></code> <code><sql:setDataSource></code> <code><sql:param></code> <code><sql:dateParam></code>

JSTL(core)

카페

- 기본적인 기능들의 구현에 이용된다.
- 출력, 수식, 제어 흐름 ,URL 처리 등에 관련된 작업을 수행한다.
- 자바 코드 작성없이 쉽게 구현이 가능하다.
- `<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>`
 - prefix : 접두어 c는 네임스페이스 개념
 - uri : TLD 파일이 존재하는 위치

기능	사용 가능한 태그
출력 태그	<code><out></code>
변수 설정/삭제 태그	<code><set></code> , <code><remove></code>
예외 처리 태그	<code><catch></code>
조건 처리 태그	<code><if></code> , <code><choose></code> , <code><when></code> , <code><otherwise></code>
반복 처리 태그	<code><forEach></code> , <code><forToken></code>
페이지 처리 태그	<code><import></code> , <code><redirect></code> , <code><url></code> , <code><param></code>

core 라이브러리

카페

기능 분류	태그	설명
변수 지원	set	JSP에서 사용될 변수를 설정한다. scope의 setAttribute() 메소드 기능을 한다.
	remove	설정한 변수를 제거한다. scope의 removeAttribute() 메소드 기능을 한다.
흐름 제어	if	조건에 따라 내부 코드를 수행한다.
	choose	switch 구문과 동일하다. 조건에 문자열도 가능하다. 다중 조건을 처리할 때 사용된다. 한 개 이상의 <c:when>과 한 개의 <c:otherwise> 서브 태그를 갖는다.
	forEach	반복 실행문에 사용된다. 컬렉션이나 Map의 각 항목을 처리할 때 사용된다.
	forEachTokens	구분자로 분리된 각각의 토큰을 처리할 때 사용된다.
URL 처리	import	URL을 사용하여 다른 자원의 결과를 삽입한다. 즉, 웹 애플리케이션의 내부/외부 자원을 사용할 때 쓴다.
	redirect	지정한 경로로 리다이렉트한다. response.sendRedirect() 기능을 한다.
	url	URL을 재작성한다. 요청 파라미터로부터 URL을 생성한다.
기타 태그	catch	body에서 실행이 되는 예외 처리에 사용된다.
	out	JspWriter에 내용을 알맞게 처리한 후 값을 출력하는 데 사용한다.

core(c:out)

- JSP의 표현식을 대체하며, 화면에 해당 **변수 값**을 **출력**한다.
- 수식을 **평가**하고, 평가된 결과를 JspWriter에 **출력**한다.

Syntax

```
<c:out value="value" escapeXml="{true|false}" default="defaultValue" />
```

항목	설명
var	속성 이름
value	출력할 값을 의미(String과 같은 문자열) 만약 value의 값이 java.io.Reader의 한 종류라면 out 태그는 Reader로부터 데이터를 읽어와 JspWriter에 값을 출력. 속성 이름 값이 Null 인 경우에는 공백으로 출력.
escapeXml	특수 문자 처리 유무 escapeXml = true(기본 값)이면 <, >, &, ', " 를 < > & ' " 등으로 출력이 된다.
default	기본 값을 지정한다.
예시	이름 : <c:out value="\${param.name}" /> <%=request.getParameter("name")%>와 동일한 문장.

core(c:set)

카페

- 변수에 값을 설정(기본형 데이터, 객체 타입)한다.
- JSP의 **영역.setAttribute()** 메소드와 같은 역할을 수행한다.

<형태>

Syntax

1. 단순 변수에 값 설정

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

2. 단순 변수에 태그의 내용 값 설정

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

3. 객체에 value 값 설정

```
<c:set value="홍길동" target="obj" property="name" />
JSP이라면 <% obj.setName("홍길동") ; %>
```

속성	설 명
var	속성 이름(변수명)
value	지정한 속성에 들어 있는 값
target	자바 빈의 객체 이름이나 Map 객체 이름
property	자바빈 객체나 Map 객체의 값을 설정할 속성 이름
scope	속성의 공유 범위의 유효 기간을 설정

참고 사항

JSP 페이지의 4가지 범위(page/request/session/application) 내에서 설정이 가능하다.

선언한 속성/변수는 EL 식에서 사용할 수 있다.

스크립팅 요소 안에서는 사용할 수 없다.

왜냐하면 <c:set> 커스텀 액션을 이용해서 선언한 변수는 자바 변수가 되는 것이 아니라, page 데이터 영역의 attribute가 되기 때문이다.

사칙 연산 수행하기

- 사칙 연산을 이용하여 **JSTL**을 사용하는 방법에 대하여 숙지해보도록 한다.

출력 결과

```
14과 5의 더하기? 19
14과 5의 빼기? 9
14과 5의 곱하기? 70
14과 5의 나눗셈? 2.8
```

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

```
<!-- 변수 num1의 값을 14로 지정하고 있다. -->
<c:set var="num1" value="14" />
<!-- 변수 num2의 값을 5로 지정하고 있다. -->
<c:set var="num2" value="5" />
```

자바와의 비교

```
int num1 = 14 ;
int num2 = 5 ;

int result = num1+num2 ;
```

```
<body>
```

```
<!-- 두 수의 덧셈의 합을 result 변수에 지정하고 있다. -->
<c:set var="result" value="${num1+num2}" />
${num1}과 ${num2}의 더하기? ${result} <br>
```

```
<!-- 두 수의 덧셈의 차를 result 변수에 지정하고 있다. -->
<c:set var="result" value="${num1-num2}" />
${num1}과 ${num2}의 빼기? ${result} <br>
```

```
<!-- 두 수의 덧셈의 곱을 result 변수에 지정하고 있다. -->
<c:set var="result" value="${num1*num2}" />
${num1}과 ${num2}의 곱하기? ${result} <br>
```

```
<!-- 두 수의 덧셈의 나눗기를 result 변수에 지정하고 있다. -->
<c:set var="result" value="${num1/num2}" />
${num1}과 ${num2}의 나눗셈? ${result} <br>
```

```
</body>
```

파일 이름	설명
Arithmetic.jsp	2개의 정수를 이용하여 간단한 산술 연산을 수행하는 페이지이다.

회원 정보 보기

- 개인의 이름과 나이 정보를 Bean 클래스와 <c:set /> core 태그를 이용하여 다뤄 보기로 한다.

출력 결과

회원 정보 보기

회원 이름: 강감찬

회원 나이: 30

```
package mypkg;

public class Member {
    private String name; //이름
    private int age ; //나이
}
```

```
<!-- JSTL Core태그 라이브러리를 사용하기 위하여 taglib 지시어를 사용한다. --%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%
    //Member 클래스에 대한 member 객체를 생성한다.
    Member member = new Member();
%>
```

```
<h1>회원 정보 보기</h1>
<!-- member 객체를 변수 member에 설정한다. -->
<c:set var="xxx" value="<%=member%>" />

<!-- name 프로퍼티의 값을 "강감찬"으로 설정한다. -->
<c:set target="${xxx}" property="name" value="강감찬" />
<c:set target="${xxx}" property="age" value="30" />

<h3>회원 이름: ${xxx.name}</h3>
<h3>회원 나이: ${xxx.age}</h3>
```

파일 이름	설명
coreSetExam01.jsp	<c:set>를 이용한 회원의 이름과 나이를 보여 주는 페이지
mypkg.Member.java	회원의 이름과 나이를 저장할 수 있는 Bean 클래스

동일 코드

```
request.setAttribute("xxx", member);

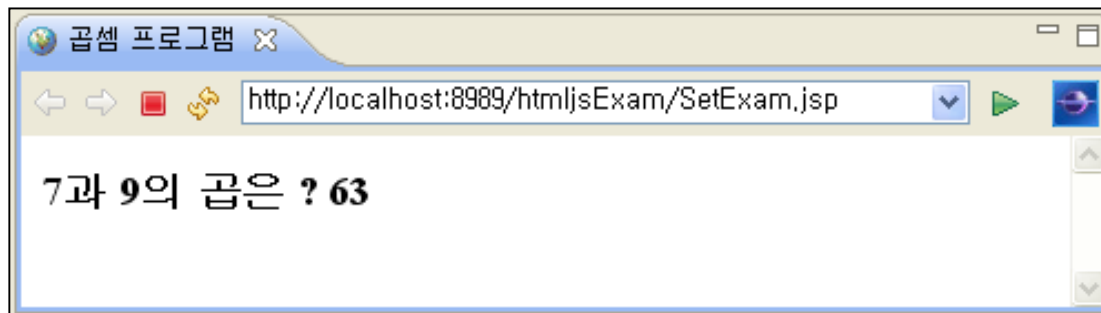
Member member = (Member)request.getAttribute("xxx");
member.setName("강감찬") ;
request.setAttribute("member", member);
```

core(c:set)

- 사용 예시
 - 2개의 정수를 이용하여 곱셈을 하는 예시

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <c:set var="num1" value="7" />
6 <c:set var="num2" value="9" />
7 <c:set var="result" value="${num1 * num2}" />
8
9 <html>
10 <head><title>곱셈 프로그램</title></head>
11 <body>
12   <h3>${num1 }과 ${num2 }의 곱은 ? ${result }</h3>
13 </body>
14 </html>
```

SetExam.jsp



core(c:remove)

- JSP의 **영역.removeAttribute()** 메소드와 같은 역할을 수행한다.
- scope에 저장된 **속성 값을 제거**한다.

Syntax

```
<c:remove var="varName" scope="{page|request|session|application}" />
```

속성	설 명
var	속성 이름(변수명)
scope	속성의 공유 범위의 유효 기간을 설정한다.
특징	scope의 기본 값은 page이다. scope가 맞지 않으면 값이 제거되지 않는다.
예시	<c:remove var="user_id" scope="request" />

core(c:catch)

- 예외 처리를 위한 태그이다.
- 태그 몸체에서 발생한 예외를 변수에 저장한다.

Syntax

```
<c:catch var="err">   예러가 발생하면 예러가 err 변수에 저장된다.  
    <%=10/0%>  
</c:catch>
```

```
<h3>&lt;c:catch&gt;로 잡아낸 오류 : <c:out value="\${err}"/></h3>
```

- var - 예외 객체를 저장할 속성 이름

core(c:if)

카페

- 조건이 맞는 경우 해당하는 태그 내용을 처리한다.
- 자바의 else 구현은 `<c:choose>` 또는 연산자를 이용하여 처리하면 된다.
- 특징
 - test 속성에 조건식을 작성한다.
 - else 구문은 지원되지 않는다.

Syntax

1. 태그 내용이 **없**는 경우
`<c:if test="조건식" var="속성 이름"
[scope="{page|request|session|application}"] />`
2. 태그 내용이 **있**는 경우
`<c:if test="testCondition" var="varName"
[scope="{page|request|session|application}"] >
body content
</c:if>`

※ 특징

if 문의 결과 값이 var 속성 값으로 설정된다.

속성	설 명
test	조건식 판별식
var	속성 이름 (조건식 결과 값이 저장됨)
scope	변수의 공유 범위

core(c:if)

- <c:if> 사용 예시

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

```
<c:if test="true">
```

```
    무조건 수행<br>
```

```
</c:if>
```

Example

```
<c:if test="${param.name == 'bk'}">
```

```
    name 파라미터의 값이 ${param.name} 입니다.<br>
```

```
</c:if>
```

```
<c:if test="${18 < param.age}">
```

```
    당신의 나이는 18세 이상입니다.
```

```
</c:if>
```

큰 정수 구하기

카페

- `<c:if />` core 태그와 EL의 내장 객체인 `param`을 이용하여 2수중에서 큰 수를 구해보기로 한다.
- 참고 사항 :
 - `else` 구문이 따로 없다.
 - 연산자를 이용하여 처리하도록 한다.

출력 결과

큰 수 구하기

숫자1 : 5

숫자2 : 3

전송

큰 수는 5 입니다.

```

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

<body>
    <!-- c:if 태그를 이용하여 큰 수를 판단한다. -->
    <!-- 넘겨진 파라미터의 차를 구해본다. -->
    큰 수는
    <c:if test="${param.num1 - param.num2 >= 0}">
        ${param.num1}
    </c:if>
    <c:if test="${param.num1 - param.num2 < 0}">
        ${param.num2}
    </c:if>
    입니다.
</body>
    
```

`<c:if>` 태그는 `test` 속성에 조건식을 넣는다.
else 절이 따로 없어서 연산자로 처리해야 한다.

파일 이름	설명
maxFrom.jsp	2개의 정수를 파라미터로 넘기는 form 페이지이다.
maxTo.jsp	<code><c:if></code> 태그를 이용하여 큰 수를 구하는 페이지이다.

core(c:choose)

카페

- 조건이 맞는 경우 처리(**switch ... case와 유사**)
- `<c:choose>`, `<c:when>`, `<c:otherwise>`
 - `when`은 자바의 `case` 구문과 유사하다
 - `Otherwise`은 자바의 `default` 구문과 유사하다.
- 조건식에 문자열 사용이 가능하다.

※ 특징

조건식이 맞으면 body 내용을 수행한다.
`<c:when ...>`은 여러 개 사용 가능.
만족하는 조건식이 없는 경우에는 `otherwise`를 수행한다.

사용 예시

```
<c:choose>
  <c:when test="${country} == 'Korea'">
    <c:out value="${country}" />의 겨울은 춥다.
  </c:when>
  <c:when test="${country} == 'Canada'">
    <c:out value="${country}" />의 겨울은 너무 춥다.
  </c:when>
  <c:otherwise>
    그 외의 나라들의 겨울은 알 수 없다.
  </c:otherwise>
</c:choose>
```


상황에 따른 인사말

카페

- <c:choose>, <c:when>, <c:otherwise> core 태그를 사용하여 상황에 맞는 간단한 인사말을 남기는 예시이다.

\$(param.name)님

<!-- 넘겨진 파라미터 greeting의 값을 이용하여 다양한 인사말을 만드는 문장이다. -->

<!-- 초면은 0이고, 구면은 1의 값을 가진다. -->

<c:choose>

<c:when test="\${param.greeting == 0}">

처음 뵙겠습니다.

</c:when>

<c:when test="\${param.greeting == 1}">

반갑습니다.

</c:when>

<c:otherwise>

누구신지요

</c:otherwise>

</c:choose>

간단한 설문 조사

이름 : 강감찬

인사 : ☒ 초면 ☐ 구면 ☐ 기타

과일을 선택하세요

메론

전송

간단한 설문 조사

이름 : 김말똥

인사 : ☐ 초면 ☒ 구면 ☐ 기타

과일을 선택하세요

메론

전송

파일 이름	설명
surveyFrom.jsp	특정 대상에게 인사말을 하기 위한 form 페이지이다. 좋아하는 과일이 어떤 것인지 설문 조사도 수행한다.
surveyTo.jsp	초면인지 구면인지에 따른 인사말을 출력하는 페이지이다.

강감찬님 처음 뵙겠습니다.
좋아하는 과일은 메론 이군요.

김말똥님 반갑습니다.
좋아하는 과일은 메론 이군요.

core(c:forEach)

- 반복문을 표현한다.(자바의 일반, 확장 for 구문이다.)
- 배열 및 Collection에 저장된 요소를 차례 대로 순회하면서 처리한다.

1. 아이템들에 대해서 반복 수행

```
<c:forEach [var="속성_이름"] items="반복할컬렉션"
           [varStatus="반복 횟수의 속성 이름"]
           [begin="시작값"] [end="마지막값"] [step="증가값"]>
```

Syntax

```
body content
</c:forEach>
```

2. 카운트에 대해서 반복 수행(items 속성이 빠짐)

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

※특징

반복 횟수는 **varStatus**의 속성_이름.count 프로퍼티를 이용하여 구한다.
item이 Map인 경우 변수에 저장되는 객체는 Map.Entry이다.
따라서, 변수 값을 사용할 때는 **\${변수.key}**와 **\${변수.value}**를 사용해서 맵에
저장된 항목의 <키, 값> 매핑에 접근할 수 있다.

속성	설 명
items	반복할 객체 이름
begin	시작 값
end	마지막 값
step	증감치
var	변수명
varStatus	별도의 변수를 줄 때 사용

구구단 문제

카페

- 사용자가 입력단 구구단의 단수를 이용하여 해당 단수를 출력해주는 프로그램을 작성하세요.
- 예를 들어 3이 넘겨지면 3단을 출력하면 된다.

단입력:

파일 이름 : exFrom3.jsp

[문제3] dan 변수를 입력한 해당 구구단 출력

```
<form action="exTo3.jsp" method="post">  
  단입력: <input type="text" name="dan" value="3"><br>  
  <input type="submit" value="단전송">  
</form>
```

결과 보기

http://localhost:8989/EL&JSTL/exTo3.jsp

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27

파일 이름 : exTo3.jsp

```
<c:set var="dan" value="${param.dan}"/>  
  
<c:forEach var="i" begin="1" end="9" step="1">  
  ${dan} * ${i} = ${dan * i} <br>  
</c:forEach>
```

반복문 실습

카페

- 파일 이름 : forEachExam01.jsp
- 자바에서 배운 for 문을 JSTL 태그로 배워 보도록 한다.

[안녕하세요] 메시지 5번 출력하기

안녕하세요
안녕하세요
안녕하세요
안녕하세요
안녕하세요

for 구문 사용 예시

1 곱하기 1 은(는) 1 입니다.
2 곱하기 2 은(는) 4 입니다.
3 곱하기 3 은(는) 9 입니다.
4 곱하기 4 은(는) 16 입니다.
5 곱하기 5 은(는) 25 입니다.
6 곱하기 6 은(는) 36 입니다.
7 곱하기 7 은(는) 49 입니다.
8 곱하기 8 은(는) 64 입니다.
9 곱하기 9 은(는) 81 입니다.

1부터 10까지의 합은 55입니다.
이것을 jstl core를 이용하여 풀어 보세요.
총합은 55 입니다.

[안녕하세요] 메시지 5번 출력하기


```
<!--  
    "안녕하세요"라는 문장을 <c:forEach> 태그를 이용하여 5번 출력하고 있다.  
    글자의 크기를 제어 변수 i를 이용하여 변경한다.  
-->  
<c:forEach var="cnt" begin="1" end="5">  
    <font size="{cnt}">안녕하세요</font><br>  
</c:forEach>  
<br><br>
```

등가의 자바

```
for( int cnt = 1 ; cnt <= 5 ; cnt++ ){  
}
```

```
<!--  
    c:forEach 태그를 이용하여 정수 1부터 9까지의  
    각각의 정수의 제곱의 값을 출력하고 있다.  
-->  
for 구문 사용 예시<br>  
<c:forEach var="i" begin="1" end="9" step="1">  
    <c:out value="{i}" />  
    <c:out value=" 곱하기 " />  
    <c:out value="{i}" />  
    <c:out value="은(는) " />  
    <c:out value="{ i * i }" />  
    <c:out value="입니다." />  
    <br>  
</c:forEach>
```

추가 문제

추가 문제 : 풀어 보세요
1부터 10까지의 합은 55입니다.

이것을 jstl core를 이용하여 풀어 보세요.

일반 for 구문은 var, begin, end, strp 속성을 이용하여 표현하면 된다.

Map 컬렉션(c:forEach)

카페

- 파일 이름 : forEachExam02.jsp
- 자바에서 배운 확장 for 문을 JSTL 태그로 배워 보도록 한다.

출력 결과

1부터 100까지 정수 중 5의 배수의 총합

결과 = 1050

now = Mon Nov 03 14:54:33 KST 2014

name = 을지문덕

gender = 남자

```
<%  
    //HashMap 객체 mapData를 정의하고 있다.  
    HashMap<String, Object> mapData = new HashMap<String,Object>();  
  
    //이름과 성별과 현재 시각을 요소로 추가하고 있다.  
    mapData.put("name", "을지문덕");  
    mapData.put("gender", "남자");  
    mapData.put("now", new java.util.Date());  
%>  
<!-- 선언된 mapData객체를 map 속성에 저장하고 있다. -->  
<c:set var="map" value="<%= mapData %>" />
```

등가의 자바

```
request.setAttribute("map", mapData);
```

```
<!--  
    c:forEach 태그를 이용하여 map 속성에  
    들어 있는 요소들의 키와 값을 출력시킨다.  
-->  
<c:forEach var="item" items="${map}">  
    ${item.key} = ${item.value}<br>  
</c:forEach>
```

varStatus 속성(c:forEach)

카페

- 파일 이름 : sportsList.jsp
- varStatus 속성은 현재 목록의 인덱스 번호(0부터 시작), 카운터 번호(1부터 시작), 목록 내용을 저장하고 있는 속성이고, 명시해 놓으면 JSTL이 자동으로 생성해준다.
- 데이터를 나열할 때 순번을 매기려면 count를 사용하면 좋다.

출력 결과

인덱스 (0부터)	카운터 (1부터)	현재 아이템
0	1	축구
1	2	야구
2	3	테니스
3	4	농구

```
<%
String[] sports = {"축구", "야구", "테니스", "농구"} ;
pageContext.setAttribute("asdf", sports) ;
%>
<table border="1">
  <tr>
    <th>목록의 인덱스<br>(0부터 시작)</th>
    <th>목록의 카운터<br>(1부터 시작)</th>
    <th>현재 아이템</th>
  </tr>
  <c:forEach items="${asdf}" varStatus="qwert">
    <tr>
      <td>${qwert.index}</td>
      <td>${qwert.count}</td>
      <td>${qwert.current}</td>
    </tr>
  </c:forEach>
</table>
```

page scope에
sports를 바인딩한다.

도전 과제

- varStatus 속성을 이용하여 다음 그림과 동일하게 요소들을 출력해 보세요.
- varStatus 속성은 현재 목록의 인덱스 번호, 카운터 번호, 목록 내용을 저장하고 있는 속성이다.

배열 원본 : int[] lotto = {44, 33, 22, 11, 9, 7};

varStatus : 반복 횟수를 위한 변수 이름

배열을 출력

파일 이름 : ex1.jsp

lotto[0] = 44
lotto[1] = 33
lotto[2] = 22
lotto[3] = 11
lotto[4] = 9
lotto[5] = 7

```
<%  
    int[] lotto = { 10, 20, 30, 5, 45, 13} ;  
%>  
  
<c:set var="array" value="<%=lotto%>"></c:set>  
  
<c:forEach items="${array}" var="arr" varStatus="status">  
    lotto[ ${status.index} ] = ${arr} &nbsp;&nbsp;&nbsp;&nbsp;<br>  
</c:forEach>
```

반복문 도전

카피

- 문제1
- 1부터 10까지 중에서 홀수의 합은 25, 짝수의 합은 30이다.
- 파일 이름 : oddEven.jsp

- 문제2
- 1부터 100까지 정수 중 5의 배수의 총합을 구하시오.
- 단, 30의 배수는 배제하시오
- 정답 : 870
- 파일 이름 : exclude30.jsp

상품 정보(session 영역)

카페

- 세션 영역에 저장되어 있는 자바의 컬렉션 정보를 출력해 보고자 한다.
- 실제 게시물에서 목록 보기 형태로 사용되는 예시이다.

<h2>JSTL 사용</h2>

```
<table border="1">
```

```
<tr>
```

```
<td>아이디</td>
<td>이름</td>
<td>재고</td>
<td>단가</td>
<td>분류</td>
```

```
</tr>
```

```
<c:forEach var="product" items="${sessionScope.xxx}">
```

```
<!-- c:forEach 태그를 이용하여 데이터를 보여 주고 있다. -->
```

```
<tr>
```

```
<td><c:out value="${product.id}" /></td>
<td><c:out value="${product.name}" /></td>
<td><c:out value="${product.stock}" /></td>
<td><c:out value="${product.price}" /></td>
<td><c:out value="${product.category}" /></td>
```

```
</tr>
```

```
</c:forEach>
```

```
</table>
```

등가의 자바

```
for( ProductBean product : items){
}
```

확장 for 구문은 var와 items 속성을 이용하여 표현하면 된다.

단순히 \${product.id}라고 표현해도 결과는 동일하다.

```
<jsp:useBean id="dao" class="mypkg.MyDao"></jsp:useBean>
```

```
<%
```

```
List<ProductBean> items = dao.GetProductList() ;
session.setAttribute("xxx", items) ;
response.sendRedirect("abcdTo.jsp") ;
```

```
%>
```

```
package mypkg;
```

```
public class ProductBean {
    private int id; //아이디
    private String name; //이름
    private int stock; //재고
    private int price; //단가
    private String category; //카테고리
}
```

출력 결과

JSTL 사용

아이디	이름	재고	단가	분류
1	아이폰	20	300	IT
2	갤럭시	30	400	IT

파일 이름

설명

mypkg.Product.java

상품 정보를 저장하고 있는 Bean 클래스이다.

mypkg.ProductDao.java

GetProductList() 메소드를 구현한다.

abcdFrom.jsp

2가지의 상품을 session 영역에 저장하고, abcdTo.jsp 페이지로 이동한다.

abcdTo.jsp

상품의 목록을 Table 형태로 보여 주는 페이지이다.

```
public List<ProductBean> GetProductList(){
```

```
List<ProductBean> lists = new ArrayList<>();
```

```
lists.add(new ProductBean(1, "아이폰", 20, 300, "IT")) ;
lists.add(new ProductBean(2, "갤럭시", 30, 400, "IT")) ;
```

```
return lists ;
```

```
}
```

core(c:forTokens)

- 문자열을 토큰들로 잘라 반복 수행한다.
- 자바의 StringTokenizer 클래스와 동일한 개념이다.

<형태>

```
<c:forTokens items="stringTokens" delims="delimiters"
    [var="varName"]
    [varStatus="varStatusName"]
    [begin="begin"] [end="end"] [step="step"]>
    body content
</c:forTokens>
```

속성	설 명
items	반복할 객체명
delims	문자열을 구분할 구분자
begin	반복할 시작값
end	반복할 마지막 값
step	증감치
var	변수명
varStatus	별도의 변수를 줄 때 사용

```
<c:forTokens var="alphabet" items="a,b,c,d,e,f,g,h,i,j,k,l,m,n" delims=",">
    <b>${alphabet}</b>&nbsp;
</c:forTokens>
```

도전 과제

- 파일 이름 : ex2.jsp
- varStatus 속성을 이용하여 다음 그림과 동일하게 요소들을 출력해 보세요.
- varStatus 속성은 현재 목록의 인덱스 번호, 카운터 번호, 목록 내용을 저장하고 있는 속성이다.

변수 원본 : 신촌,여의도,독도,제주도,울릉도

도시[0] = 신촌
도시[1] = 여의도
도시[2] = 독도
도시[3] = 제주도
도시[4] = 울릉도

```
<c:set var="area" value="서울,제주,경북,전남,강원"></c:set>
```

```
<c:forEach var="oneitem" items="${area}" delims="," varStatus="status">
```

```
    지역[ ${status.count} ] : ${oneitem} &nbsp;&nbsp;&nbsp;<br>
```

```
</c:forEach>
```

문자열을 구분자로 나누어 보기

- forToken core 태그는 자바의 StringTokenizer 클래스와 동일한 개념을 가지고 있다.

첫 번째 토큰

가가
나나
다다
라라

두 번째 토큰

제시카
티파니
효연
태연
써니

```
public String GetToken2(){
    String result = "제시카,티파니,효연,태연,써니" ;
    return result ;
}
```

```
public String GetToken1(){
    String result = "" ;
    String[] arr = {"가가", "나나", "다다", "라라"};

    for (int i = 0; i < arr.length; i++) {
        if ( i == arr.length -1 ) {
            result += arr[i] ;
        } else {
            result += arr[i] + "," ;
        }
    }
    System.out.println( result );
    return result ;
}
```

```
<jsp:useBean id="dao" class="mypkg.MyDao"></jsp:useBean>
<%
    String mytoken1 = dao.GetToken1() ;
    String mytoken2 = dao.GetToken2() ;
%>
```

```
<h3>첫 번째 토큰</h3>
<c:forTokens var="one" items="<%=mytoken1%>" delims=",">
    ${one}<br>
</c:forTokens>

<hr>

<h3>두 번째 토큰</h3>
<c:forTokens var="one" items="<%=mytoken2%>" delims=",">
    ${one}<br>
</c:forTokens>
```

구분자는 콤마와 마침표
2개를 사용하고 있다.

파일 이름	설명
coreFortokensExam01.jsp	문자열을 / 또는 , 또는 .으로 분리하는 예시 파일이다.
MyDao.java	GetToken1(), GetToken2() 메소드를 구현한다.

forToken 사용(자바의 StringTokenizer 클래스 공부)
체크 박스를 만들되 홀수번째 요소는 체크 되도록 설정하세요.
☒가나 ☐다라 ☒마바 ☐사아

2개 이상의 항목을 구분자로 사용하는 경우 :
소시 멤버를 이용하여 콤보 박스 형태로 만드시오.(서현이 기본 선택되게):

서현 ▼

core(c:import)

- 이미 만들어 놓은 URL 자원(*.jsp 파일)의 내용을 import한다.
- HTTP, FTP 같은 외부 사이트의 내용도 포함할 수 있다.

<형태>

1. Reader가 주어지지 않은 경우

```
<c:import url="url" [context="context"]  
          [var="varName"]  
          [scope="{page|request|session|application}"]  
          [charEncoding="charEncoding"]>  
    optional body content for <c:param> subtags  
</c:import>
```

2. Reader가 주어지지 않은 경우

```
<c:import url="url" [context="context"]  
          varReader="varReader"  
          [charEncoding="charEncoding"]>  
    body content wherer varReader is consumed by another action  
</c:import>
```

예시

```
<c:import url="/date.jsp" var="url" />
```

속성	설 명
url	url
var	읽어올 데이터를 저장할 변수명
scope	변수의 공유 범위
varReader	리소스의 내용을 Reader 객체로 읽어 오는 경우
charEncoding	읽어 온 데이터의 Character set을 지정

core(c:import)

지정한 URL에 연결하여 결과를 지정한 변수에 저장한다.

Syntax

```
<c:import url="URL" charEncoding="캐릭터인코딩" var="변수명" scope="범위" >  
  <c:param name="파라미터이름" value="파라미터값" />  
</c:import>
```

- url - 결과를 읽어올 URL
- charEncoding - 읽어온 결과를 저장할 때 사용할 캐릭터 인코딩
- var - 읽어온 결과를 저장할 변수명
- scope - 변수를 저장할 영역
- <c:param> 태그는 url 속성에 지정한 사이트에 연결할 때 전송할 파라미터를 입력한다.

Example

```
<c:set var="url" value="http://www.naver.com"/>  
  
<c:import url="${url}" var="u" charEncoding="EUC-KR" />  
  
<c:out value="${url}" /> import 하기.  
  
<hr>  
<pre><c:out value="${u}" /></pre>
```

core(c:url)

- URL을 생성해서 변수에 저장한다.
- URL rewriting방법을 이용해서 URL을 기술

Syntax

```
<c:url var="변수명" value="재작성할URL" scope="저장범위" >  
  <c:param name="파라미터이름" value="파라미터값" />  
</c:url>
```

- var - 생성한 URL이 저장될 변수명이다.
- value - 생성할 URL
- scope - 변수를 저장할 범위를 지정한다.

Example

```
<c:url var="url1" value="../shopping.do" />  
<c:url var="url2" value="/shopping.do" >  
  <c:param name="Add" value="isdn-001" />  
</c:url>  
<c:url var="url3" value="http://localhost:8080/jstl/setTag.jsp" />
```

core(c:redirect)

- HTTP 요청을 클라이언트를 통해 지정한 페이지로 리다이렉트 한다.
- `response.sendRedirect()`와 동일한 개념으로 이해하면 된다.

Syntax

```
<c:redirect url="리다이렉트할URL">  
  <c:param name="파라미터이름" value="파라미터값" />  
</c:redirect>
```

- url - 리다이렉트 URL
- <c:param>은 리다이렉트할 페이지에 전달할 파라미터 지정

Example

```
<c:redirect url="/ifTag.jsp">  
  <c:param name="name" value="bk" />  
</c:redirect>
```


core(c:param)

- <c:import>, <c:url>, <c:redirect>에서 파라미터 전달

<형태>

1. 태그 내용이 없는 경우

```
<c:param name="name" value="value" />
```

2. 태그 내용이 있는 경우

```
<c:param name="name" value="value" >
```

```
    parameter value
```

```
</c:param>
```

<c:redirect> 태그 사용하기

- <c:redirect> core 태그를 이용하여 "MultiplyTo.jsp" 페이지로 이동시키는 예시이다.

출력 결과

곱셈 프로그램

14과 5의 곱은? 70

```
<!-- 두 개의 파라미터 su1, su2를 사용하여 MultiplyTo.jsp 페이지로 이동한다. -->
<c:redirect url="MultiplyTo.jsp" >
    <c:param name="su1" value="14" />
    <c:param name="su2" value="5" />
</c:redirect>
```

```
<!-- 넘어오는 2개의 파라미터를 이용하여 곱셈을 연산한다. -->
<!-- 그 결과는 변수 result에 설정한다. -->
<c:set var="result" value="${param.su1*param.su2}" />

<body>
    <h3>곱셈 프로그램</h3>
    <!-- 해당 결과물을 출력한다. -->
    ${param.su1}과 ${param.su2}의 곱은? ${result}
</body>

</html>
```

파일 이름	설명
MultiplyFrom.jsp	2개의 정수 파라미터를 <c:redirect> 태그를 이용하여 "MultiplyTo.jsp" 페이지로 이동시키는 페이지이다.
MultiplyTo.jsp	2개의 정수 파라미터를 이용하여 곱셈을 연산하는 페이지이다.

format 라이브러리 소개

카페

항목	설 명
특징	국제화/형식화 의 기능을 제공한다. 특정 지역에 따라 알맞은 메시지를 출력하는 경우에 사용한다. 로컬과 언어를 지정하기 위한 방법들을 제공
국제화	다국어 버전을 지원한다.
형식화	날짜 및 숫자 등에 대한 서식 처리 를 한다. 예) 숫자는 3자리 마다 콤마, 소수점은 2자리 까지 표시하라 등등
표현 방법	<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %> prefix : 접두어 c는 네임스페이스 개념 uri : TLD 파일이 존재하는 위치

국제화 태그

카페

- JSTL fmt는 국제화 지역화 태그로 JSP 페이지에서 다양한 언어를 지원 받을 수 있도록 할 수 있고, 날짜와 숫자 형식 등을 formatting하는 데 사용된다.

기능 분류	태그	설명
로케일 지정	setLocale	Locale을 지정한다.
	requestEncoding	요청 파라미터의 캐릭터 인코딩을 지정한다.
메시지 처리	bundle	사용할 번들을 지정한다.
	message	지역에 알맞은 메시지를 출력한다.
	setBundle	리소스 번들을 읽어와 특정 변수에 저장한다.
숫자 및 날짜 포매팅	formatNumber	숫자를 포매팅한다.
	formatDate	Date 객체를 포매팅한다.
	parseDate	문자열로 표시된 날짜를 분석해서 Date 객체로 변환
	parseNumber	문자열로 표시된 날짜를 분석해서 숫자로 변환
	setTimeZone	시간대 정보를 특정 변수에 저장한다.
	timeZone	시간대를 지정한다.

format(fmt:requestEncoding)

- HTTP 요청의 **문자 인코딩**을 설정하기 위해 사용.
- 즉, request 객체로부터 전달 받은 값을 인코딩한다.
- `<fmt:requestEncoding value="euc-kr" />`은
- `<% request.setCharacterEncoding="euc-kr" ; %>`와 등가이다.

Syntax

`<fmt:requestEncoding value="캐릭터인코딩" />`

- value - 캐릭터 인코딩

format(fmt:setLocale)

- 로케일(Locale)이란?
 - 사용자의 언어, 국가뿐 아니라 사용자 인터페이스에서 사용자가 선호하는 사항을 지정한 매개 변수의 모임.
 - 예를 들어, 한국은 \(\원)을 사용하고, 숫자의 3자리마다 콤마를 붙인다.
- 국제화 태그에 적용될 **로케일**을 **지정**한다.
- 다국어 페이지의 언어를 지정하는 태그이다.

사용 형식 : `<fmt:setLocale value="값" variant="" scope="영향을_미치는_범위" />`

사용 예시 :

```
<fmt:setLocale value="ko" scope="request" />
```

```
<fmt:setLocale value="${param.locale}" scope="request" />
```

- value 속성
 - 어떤 언어를 사용할지(Locale 지정) 지정하는 속성
 - 두 글자로 된 언어 코드를 반드시 지정해주어야 한다.
 - 두 글자로 된 국가 코드를 추가로 지정할 수 있다.
 - 한국(ko), 영어(en)
- scope 속성
 - 지정한 Locale이 영향을 미치는 범위를 지정한다.
- variant 속성 : 브라우저의 스펙 지정

format(fmt:bundle)

- properties 확장자를 가진 파일의 리소스를 로딩할 때 사용한다.
- 첫 태그와 끝 태그 사이의 영역에서만 사용 가능하다.
- 다국어 처리를 위한 용도로 사용한다.
- 주의 사항 :
 - `<fmt:bundle>` 다국어 버전 관련 코드는 여기에 작성하세요 `</fmt:bundle>`
- 사용 방법
 - `/WEB-INF/classes` 폴더 내에 파일을 생성한다.
 - `basename` 속성 작성시 하위 패키지가 존재하면 `dot(.)`으로 구분하면 된다.
 - 예시
 - `/WEB-INF/classes/abcd/resource.properties` 파일이 존재한다면 `<fmt:bundle basename="abcd.resource">` 이라고 작성하면 된다.

출력 결과

```
<fmt:bundle basename="리소스번들" prefix="기본키점두어">
...
  <fmt:message key="키값1" />
...
  <fmt:message key="키값2" />
...
</fmt:bundle>
```

속성	설 명
basename	사용할 리소스 번들의 이름 (* .properties 파일)
prefix	message 태그의 key 속성의 값 앞에 자동으로 붙게 될 문자열
key	리소스 번들에 저장된 메시지 키 값

format(fmt:setBundle)

- 지정한 리소스 번들로부터 메시지를 읽어와 출력.
- properties 확장자를 가진 파일의 리소스 로딩시 사용
- 태그는 페이지 전체에 적용

Syntax

```
<fmt:setBundle var="변수명" basename="리소스번들" scope="범위" />
```

...

```
<fmt:message bundle="${변수명}" key="키값" />
```

- var : 불러올 리소스 내용을 가지고 있을 변수 이름
- basename : 리소스 번들의 이름
- bundler : 참조할 리소스 번들 변수명
- key : 리소스 번들에 저장된 메시지 키 값
- scope : 범위 지정시 사용

* bundler 속성을 사용하면, message 태그가 bundle 태그의 몸체에 중첩될 필요가 없다.

format(fmt:message)

- properties 파일의 리소스를 읽을 때 사용한다.
- <형태>
 - <fmt:message key="키값" bundle="bundle변수" var="변수명" scope="범위">
 - key 속성으로 구분된 내용을 가지고 온다.
 - bundle 속성에는 var 변수를 입력할 수 있다.

속성	설 명
key	읽어올 메시지의 키
var	해당 메시지를 저장할 변수 이름
scope	변수가 저장 되는 영역
bundle	번들의 이름

- 사용 예시
 - <fmt:message key="title" var="mytitle"/>
 - <fmt:message key="address" />

```
resource.properties.src
1 title=웹 프로그래밍
2 address=서울 강남구
3 company=한빛
4 me=홍길동
5 visitor=귀하의 아이디는 {0}입니다.
```

format(fmt:param)

- message 태그내의 파라미터를 전달하는 데 사용
- <형태>
 - <fmt:param value="파라미터값">

다국어 버전

- 파일 이름 : msgBundleForm.jsp, msgBundleTo.jsp, resource.properties, resource.properties.src
- 순서
- properties 파일을 만든다.
- native2ascii.exe 파일을 이용하여 한글에 대한 리소스를 만들도록 한다.
- msgBundleForm.jsp 파일을 작성한다.
- msgBundleTo.jsp 파일을 작성한다.

format(fmt:timeZone)

- 날짜와 시간에는 시간대란 것이 존재한다.
- 한국과 홍콩의 시간은 차이가 나며, JSTL의 <fmt:timeZone /> 태그가 사용된다.
- 첫 태그와 끝 태그 사이의 body 부분의 내용에서만 사용된다.
- 지정한 지역의 값으로 시간대를 맞추는 경우에 사용한다.

Syntax

```
<sql:timeZone value="어떤_값" />
```

항목	설명
value	시간대가 들어간다.

사용 예시

```
<fmt:timeZone value="Tokyo" >  
    <fmt:formatDate value="${now}" type="both" dateStyle="full" timeStyle="full" />  
</fmt:timeZone>
```

세계 각국의 시각(TimeZone) 출력하기

- 날짜와 시간에는 시간대(TimeZone)란 것이 존재한다.
- 한국과 홍콩의 시간은 차이가 나며, JSTL의 <fmt:timeZone /> 태그가 사용된다.

Etc/GMT-13
MIT
Pacific/Apia
Pacific/Enderbury
Pacific/Fakaofa
Pacific/Tongatapu
Etc/GMT-14
Pacific/Kiritimati

서울 시간 : 2015년 6월 22일 월요일 오전 12시 59분 29초 EST
홍콩 시간 : 2015년 6월 22일 월요일 오전 12시 59분 29초 EST
호주 시간 : 2015년 6월 23일 화요일 오전 12시 59분 29초 EST
런던 시간 : 2015. 6. 22 오후 3:59:29
뉴욕 시간 : 2015. 6. 22 오전 10:59:29

```
<!--
전세계의 모든 TimeZone을 구하기 위하여
TimeZone.getAvailableIds() 메소드를 사용하고 있다.
변수 id에 저장하고, 이것을 출력한다.
-->
timeZone이란 해당 지역의 시간대를 의미한다.<br>
<c:forEach var="id" items="%= java.util.TimeZone.getAvailableIds() %>"
    ${id}<br/>
</c:forEach>

<hr>
<!-- 변수 now에 현재 시간을 설정하고, 서울의 시간을 출력한다. -->
<c:set var="now" value="%= new java.util.Date() %>" />
    서울 시간 : <fmt:formatDate value="${now}" type="both" dateStyle="full" timeStyle="full" />

<!-- 홍콩의 현재 시각을 출력한다. -->
<fmt:timeZone value="Hongkong">
    홍콩 시간 : <fmt:formatDate value="${now}" type="both" dateStyle="full" timeStyle="full" />
</fmt:timeZone>
```

파일 이름	설명
fmtTimezoneExam.jsp	여러 지역의 TimeZone을 이용하여 현재 시각 출력하는 페이지이다.

format(fmt:setTimeZone)

- <fmt:timeZone /> 태그가 동일한 기능을 한다.
- 다만 페이지 전체에 적용되는 것이 차이점이다.

Syntax

```
<sql:setTimeZone value="어떤_값" var="변수_이름" scope="범위"/>
```

항목	설명
var	지정한 시간대 값이 저장된다.
value	시간대가 들어간다.
scope	속성의 공유 범위의 유효 영역을 설정한다.

format(fmt:formatNumber)

카페

- 숫자 형식의 패턴/서식(format 라고 부른다.)을 설정할 때 사용한다.

- <형태>

- <fmt:formatNumber value="값" type="숫자의타입">
- <fmt:formatNumber value="{price}" type="percent" />

속성	표현식/EL	타입	설 명(빨간 색이 기본 값이다.)
value	사용 가능	String 또는 Number	입력 양식에 맞춰서 출력할 숫자이다.
type	사용 가능	String	출력할 양식(숫자, 통화, 퍼센트) number : 숫자 타입, percent : % 형식, currency : 통화 형식
pattern	사용 가능	String	지정한 값을 어떠한 값으로 변화시킬지... 숫자가 출력되는 양식 java.text.DecimalFormat
currencyCode	사용 가능	String	통화 코드를 지정한다.(ISO 4217에 정의되어 있다.) 한국의 "원"화에 대한 통화 코드는 KRW이다. type의 속성이 currency일 때만 의미가 있다.
currentSymbol	사용 가능	String	통화를 표현할 때 사용할 기호를 지정한다. type의 속성이 currency일 때만 의미가 있다.
groupingUsed	사용 가능	boolean	콤마와 같이 단위를 구분할 때 사용되는 기호를 사용할 지의 여부를 설정한다. true false
var	사용 불가	String	포매팅한 결과를 저장할 변수이다.(String 타입) var 속성을 사용하지 않으면 결과가 바로 출력된다.
scope	사용 불가	String	변수가 공용되는 범위 지정(page , request, session, application)

format(fmt:formatNumber)

카페

- fmt:formatNumber 사용 예시

예시	<fmt:formatNumber pattern="###,###" value="\${order.orderLists[cnt].price}"/>
설명	order 객체의 멤버 변수인 orderLists 컬렉션의 0번째 요소의 price 프로퍼티의 값을 3자리마다 콤마 형식으로 표현하세요.
예시	<fmt:formatNumber pattern="0" value="\${product.price}" /> 원
설명	product 객체의 price 프로퍼티의 값을 정수형으로 표현하세요.
예시	<fmt:formatNumber pattern="0.00" value="\${product.point}" /> point
설명	product 객체의 point 프로퍼티의 값을 소수점 2째 자리까지 표현하세요.
예시	<fmt:formatNumber pattern="###,###.00" value="\${sumTotalPrice}"/>원
설명	sumTotalPrice 프로퍼티의 값을 3자리 마다 콤마 유형으로 소수점 2째 자리까지 표현하세요.

숫자에 대한 서식 지정하기

카페

- 파일 이름 : fmtNumberExam.jsp
- 숫자에 대한 서식을 지정하는 태그로써 <fmt:formatNumber>가 존재한다.
- 이것을 이용하여 다음 문제를 풀어 보도록 한다.
- 문제
 - 어떤 제품이 단가가 1250000이고, 할인율이 0.35이다.
 - 이에 대한 판매가를 구하는 프로그램을 <fmt:formatNumber/> 태그로 만들어 보자.
 - 또한, 판매가가 100만원이 넘으면 비싸군요, 아니요 싸네요를 출력하는 문장을 만들어 보자.
 - 포매팅을 위한 패턴 공부도 해보도록 한다.

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

<%
    request.setAttribute("price", 1250000); //단가
    request.setAttribute("discount", 0.35); //할인율
    //판매가 : 812500
%>
```

```
원가 : <fmt:formatNumber type="number" value="${price}"/><br>
할인율 : <fmt:formatNumber type="number" value="${discount}"/><br>
할인율(%) : <fmt:formatNumber type="percent" value="${discount}"/><br>
판매가 : <fmt:formatNumber value="${price*(1-discount)}"/><br>
<hr>
패턴 1 : <fmt:formatNumber pattern="###,###,###.00" value="${price}"/><br>
패턴 2 : <fmt:formatNumber pattern="000,000,000.00" value="${price}"/><br>
<hr>
<%-- 판매가가 100만원이 넘으면 비싸군요, 아니요 싸네요를 출력하세요. --%>
<c:set var="sale" value="${price*(1-discount)}"></c:set>
<c:if test="${sale >= 1000000}">
    비싸군요
</c:if>
<c:if test="${sale < 1000000}">
    싸네요
</c:if>
```

도전 과제

- 문제4
- Product 클래스 변수 리스트
 - 이름, 수량, 단가, 할인율
- MyDao 클래스
 - GetList() 메소드 구현
 - 상품 2개의 컬렉션 객체를 만든다.
- jsp 파일
 - GetList() 메소드를 호출하여 정보를 읽어 온 다음에 session에 바인딩하도록 한다.
- 이 세션 영역의 정보를 이용하여
- 표를 다음과 같이 만들되, 결제 금액은 반드시 jsp에서 JSTL을 이용하여 구해보도록 한다.
- 숫자에 대한 콤마 및 퍼센트도 모두 JSTL을 이용하도록 한다.
- | 이름 | 수량 | 단가 | 할인율 | 판매 금액 |
|----|----|-------|-------|--------|
| 사과 | 10 | 2,000 | 10% | 18,000 |
| 감 | 30 | 1,000 | 15% | 25,500 |
| | | 결제 금액 | 43500 | |
-

숫자에 대한 서식 지정

카페

- 파일 이름 : fmtNumberExam.jsp

자바의 DecimalFormat 클래스와 유사


```
<!-- 변수 price에 값을 설정한다. -->
<:set var="price" value="99999" />
<fmt:formatNumber value="${price}" type="number" var="numberType" />
<br>
통화: <fmt:formatNumber value="${price}"
      type="currency" currencySymbol="원" />
```

```
<!--
      groupingUsed 속성의 값이 true(기본 값)이면
      콤마와 같이 단위를 구분할 때 사용할 기호가 표시된다.
      예시에서는 false이므로 콤마가 표현되지 않는다.
-->
퍼센트: <fmt:formatNumber value="${price}"
      type="percent" groupingUsed="false" />
```

출력 결과

자바의 DecimalFormat 클래스와 유사

통화: 원 99,999.00
퍼센트: 9999900%
숫자: 99,999
패턴 1: 099,999.00
패턴 2: 99999.00
패턴 3: 99,999.00
패턴 4: 99999.00
첫번째 수: 1,234,500
두번째 수: 3.14
세번째 수: 10.50

```
패턴1: <fmt:formatNumber value="${price}" pattern="000,000.00"/>
<br>
패턴2: <fmt:formatNumber value="${price}" pattern="000.00"/>
<br>
패턴3: <fmt:formatNumber value="${price}" pattern="###,###.00"/>
<br>
패턴4: <fmt:formatNumber value="${price}" pattern="###.00"/>
<br>
```

```
첫번째 수: <fmt:formatNumber value="1234500" groupingUsed="true" />
<br>
두번째 수: <fmt:formatNumber value="3.14158" pattern="#.##" />
<br>
세번째 수: <fmt:formatNumber value="10.5" pattern="#.00" />
```

format(fmt:parseNumber)

- 문자열에서 숫자 형식으로 parsing(파싱)한다.

```
<c:set var="sometext" value="123,456" />
<fmt:parseNumber var="su" value=${sometext} />

<c:set var="hap" value=${su+100} />
<fmt:formatNumber value=${hap} pattern="###,###.00" />
```

속성	표현식/EL	타입	설 명(빨간 색이 기본 값이다.)
value	사용 가능	String	Number로 파싱할 문자열 형의 숫자를 의미한다.
type	사용 가능	String	value 속성의 문자열 타입을 지정한다.(number, currency, percentage)
pattern	사용 가능	String	사용자가 지정할 파싱 형식(패턴)을 지정한다.
parseLocale	사용 가능	String 또는 java.util.Locale	파싱할 때 사용할 Locale을 지정한다. Locale이란 숫자, 통화, 퍼센트 등에 대한 기본 형식의 패턴을 말한다.
integerOnly	사용 가능	boolean	주어진 값에서 integer 부분만 파싱할 것인가의 여부를 설정한다. (false, true)
var	사용 불가	String	파싱한 결과를 저장할 변수 이름을 설정한다.
scope	사용 불가	String	변수가 공용되는 범위 지정(page, request, session, application)

format(fmt:formatDate)

- 날짜 형식의 패턴을 설정할 때 사용하는 태그이다.
- 형태
 - `<fmt:formatDate value="값" type="타입" dateStyle="값" timeStyle="타입" pattern="패턴" timeZone="값" var="변수명" scope="범위">`
- 사용 예시
 - `<fmt:formatDate value="${now}" pattern="z a h : mm"/>`
 - `<fmt:formatDate value="${boardVO.regdate}" pattern="yyyy-MM-dd HH:mm"/>`

속성	표현식/EL	타입	설 명(빨간 색이 기본 값이다.)
value	사용 가능	java.util.Date	포매팅할 날짜 및 시간의 데이터(Date 타입)
type	사용 가능	String	format type(기본 값 : date) 날짜(date)/시간(time) 또는 날짜와 시간 둘 다 포함(both)한 타입 지정 가능.
dateStyle	사용 가능	String	날짜의 출력 스타일 지정(default, short, medium, long, full)
timeStyle	사용 가능	String	시간의 출력 스타일 지정(default, short, medium, long, full)
pattern	사용 가능	String	직접 출력 패턴을 지정(java.text.DateFormat 클래스 타입의 패턴)
var	사용 불가능	String	파싱한 결과를 지정할 변수 이름을 지정한다.
scope	사용 불가능	String	변수가 공용되는 범위 지정(page, request, session, application)
timeZone	사용 가능	String 또는 java.util.TimeZone	날짜와 시간이 표시될 시간대 시간대를 변경하고자 하는 경우 지정

format(fmt:parseDate)

- 문자열을 날짜와 시간의 형태로 변환하는 태그이다.
- 자바 또는 데이터베이스에서 넘어온 문자열 형식의 데이터를 날짜 형식으로 바꿀 때 사용하면 좋다.

```
<!-- 어떤 문자 -->  
<c:set var="mydate" value="2016/12/25 13:25:55" />  
<fmt:parseDate var="now" value=${mydate} " pattern="yyyy/MM/dd HH:mm:ss" />
```

속성	표현식/EL	타입	설 명(빨간 색이 기본 값이다.)
value	사용 가능	String	파싱할 문자열을 의미한다.
type	사용 가능	String	날짜(date)/시간(time) 또는 날짜와 시간 둘 다 포함(both)한 타입 지정 가능.
dateStyle	사용 가능	String	날짜의 출력 스타일 지정(default, short, medium, long, full)
timeStyle	사용 가능	String	시간의 출력 스타일 지정(default, short, medium, long, full)
pattern	사용 가능	String	직접 출력 패턴을 지정(java.text.DateFormat 클래스 타입의 패턴)
timeZone	사용 가능	String 또는 java.util.TimeZone	시간대를 변경할 때 사용한다.
parseLocale	사용 가능	String 또는 java.util.Locale	파싱할 때 사용할 Locale을 지정한다. Locale이란 숫자, 통화, 퍼센트 등에 대한 기본 형식의 패턴을 말한다.
var	사용 불가	String	파싱한 결과를 지정할 변수 이름을 지정한다.
scope	사용 불가	String	변수가 공용되는 범위 지정(page, request, session, application)

숫자에 대한 서식 지정하기

- 파일 이름 : fmtTest.jsp
- 요구 사항
 - 어떤 문자열 "2016/12/25 13:25:55"가 있다.
 - 이것을 날짜 형식으로 파싱하고, 여러 가지 타입으로 해당 날짜 정보를 출력해 보도록 한다.
 - 어떤 숫자 형식으로 구성된 문자열 "123,456"이 있다.
 - 이것을 숫자 형식으로 파싱하고, 더하기 100을 연산한 값을 출력해 보도록 한다.

```
<!-- 어떤 문자 -->
<c:set var="mydate" value="2016/12/25 13:25:55" />

<fmt:parseDate var="now" value=${mydate} pattern="yyyy/MM/dd HH:mm:ss" />

<fmt:formatDate value=${now} type="date" dateStyle="full" /><br>
<fmt:formatDate value=${now} type="date" dateStyle="short" /><br>

<fmt:formatDate value=${now} type="time" timeStyle="full" /><br>
<fmt:formatDate value=${now} type="time" timeStyle="short" /><br>

<fmt:formatDate value=${now} type="both" dateStyle="full" timeStyle="full" /><br>
<fmt:formatDate value=${now} pattern="Z a h:mm" /><br>

<c:set var="sometext" value="123,456" />
<fmt:parseNumber var="su" value=${sometext} />

<c:set var="hap" value=${su+100} />
<fmt:formatNumber value=${hap} pattern="###,###.00" />
```

```
2016년 12월 25일 일요일
16. 12. 25
오후 1시 25분 55초 KST
오후 1:25
2016년 12월 25일 일요일 오후 1시 25분 55초 KST
KST 오후 1:25
123,556.00
```

sql 라이브러리 소개

카페

- 데이터베이스 관련 웹 응용프로그램을 개발하는 데 사용하는 태그이다.
- sql 관련 기능을 제공(쿼리 전송, 트랜잭션 처리)한다.

태그	설명
<sql:transaction>	트랜잭션을 처리할 때 사용하는 태그이다.
<sql:query>	executeQuery() 메소드와 동일한 기능을 수행하는 태그이다.
<sql:update>	executeUpdate() 메소드와 동일한 기능을 수행하는 태그이다.
<sql:param>	java.sql.PreparedStatement.setString() 메소드와 동일한 기능을 수행하는 태그이다.
<sql:dateParam>	java.sql.PreparedStatement.setTimestamp() 메소드와 동일한 기능을 수행하는 태그이다.
<sql:setDataSource>	javax.sql.DataSource를 지정할 때 사용하는 태그이다.

sql(sql:setDataSource)

- JDBC의 데이터 소스(javax.sql.DataSource)를 지정할 때 사용한다.
- 만약 context에 DataSource가 지정되어 있다면 <sql:query> 태그를 사용하여 DataSource를 사용할 수 있다.

Syntax

```
<sql:setDataSource  
    var="변수이름" driver="DBdriver" url="DBurl" user="DBuser" password="DBpassword" />
```

항목	설명
var	데이터 소스 정보를 저장하고 있는 변수 이름
url	JDBC Url 정보
driver	JDBC Driver 이름
user	데이터 베이스 사용자 계정
password	데이터 베이스 사용자의 비밀번호
dataSource	컨텍스트에 JNDI 설정시 리소스 이름을 지정한다.
scope	속성의 공유 범위의 유효 영역을 설정한다.

sql(sql:query)

- 데이터베이스 질의어(쿼리 문장)을 실행하여, 그 결과를 변수에 저장한다.
- JDBC 프로그래밍의 executeQuery() 메소드와 동일한 결과를 반환한다.

Syntax

```
<sql:query sql="sqlQuery" var="변수이름" scope="어떤_영역" dataSource="dataSource"  
maxRows="최대행수" startRow="시작행번호" />
```

항목	설명
sql	SQL 쿼리 문장을 지정한다.
var	쿼리의 결과를 저장할 변수 이름이다.
scope	var 속성의 공유 범위의 유효 영역을 설정한다.
dataSource	컨텍스트에 J2EE 설정시 리소스 이름을 지정한다.
maxRows	쿼리의 결과에 포함될 최대 행수를 의미한다.
startrow	쿼리의 결과에 포함될 시작 행 번호를 의미하는 데, 0부터 시작한다.

sql(sql:update)

- sql의 insert, update, delete 문장에 수행된다.
- JDBC의 executeUpdate() 메소드와 동일한 역할을 한다.

Syntax

```
<sql:update dataSource="dataSource" var="변수이름" scope="어떤_영역" />
```

항목	설명
sql	SQL 쿼리 문장을 지정한다.
var	쿼리의 결과를 저장할 변수 이름이다.
scope	var 속성의 공유 범위의 유효 영역을 설정한다.
dataSource	컨텍스트에 JNDI 설정시 리소스 이름을 지정한다.

sql(sql:param)

- 문자열 파라미터인 sql 문장의 ? 부분을 치환하기 위하여 사용한다.
- PreparedStatement의 setString() 메소드와 동일하다.

Syntax

<sql:param value="어떤_값" />

항목	설명
value	파라미터의 값을 지정한다.

형태

1. 태그 내용이 없는 경우
 <sql:param value="value" />
2. 태그 내용이 있는 경우
 <sql:param>
 parameter value
 </sql:param>

sql(sql:dateParam)

- sql 문장의 ? 부분에 날짜 타입의 데이터를 설정한다.
- 날짜 관련 파라미터를 사용할 수 있다.
- PreparedStatement 의 setTimestamp()메소드와 동일하다.

Syntax

```
<sql:dateParam value="어떤_값" type="어떤_타입"/>
```

항목	설명
value	파라미터의 값을 지정한다.
type	timestamp time date 중 택일이다.

sql(sql:transaction)

- <sql:query>, <sql:update>를 위한 트랜잭션을 설정한다.

Syntax

```
<sql:transaction dataSource="dataSource" isolation="격리_수준">  
    <sql:query> and <sql:update> statements  
</sql:transaction>
```

항목	설명
dataSource	JDBC의 리소스 네임 또는 DriverManager을 위한 파라미터
isolation	"read_committed", "read_uncommittes", "repeatable_read", "serializable" 중 하나이다

XML 라이브러리 소개

- XML에서 자주 사용되는 기능들을 태그 라이브러리로 모아 놓은 것이다.
- XML 문서를 파싱하고 XML 문서의 내용을 추출, 변환한다.
- 데이터를 표현하거나, 제어문 등에 사용될 수 있다.
- Xpath에 사용된다.
- 등록해야 할 코드
 - `<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>`

기능	태그
출력, 변수 설정 태그	<code><out></code> , <code><set></code>
제어 흐름/조건 처리 태그	<code><if></code> , <code><choose></code> , <code><when></code> , <code><otherwise></code>
반복 처리 태그	<code><forEach></code>
변환/기타	<code><transform></code> , <code><param></code> , <code><parser></code> , <code><remove></code>

xml(x:out)

- XPath 식을 평가하고 결과를 출력(내용 출력)한다.

Syntax

```
<x:out select="Xpath_Expression" [escapeXml="{true|false}"] />
```

항목	설명
select	출력하기 위한 Xpath 표현식이다.
escapeXml	특수 기호의 출력 형태를 설정할 때 사용되는 속성이다. false(기본 값), true이면 < 값은 <으로 > 값은 >으로 출력이 된다.

xml(x:set)

- XPath 식을 평가하고 결과 값을 **변수에 저장**한다.
- XML의 내용을 저장하는 태그이다.

Syntax

```
<x:set select="XPathExpression"  
var="변수이름" [scope="{page|request|session|application}"] />
```

항목	설명
var	저장할 변수 이름이다.
select	출력하기 위한 Xpath 표현식이다.
scope	var 속성의 공유 범위의 유효 영역을 설정한다.

xml(x:if)

- select 속성이 참인 경우에 수행되는 태그이다.
- <c:if> 태그와 유사한 기능이다.

Syntax

```
<x:if select="XPathExpression" [var="varName"] [scope="{page|request|session|application}"] >  
    body content  
</x:if>
```

항목	설명
var	조건의 처리의 결과를 저장할 변수 이름이다.
select	조건을 판별하기 위한 Xpath 표현식이다.
scope	var 속성의 공유 범위의 유효 영역을 설정한다.

xml(x:choose...when...otherwise>

- switch 문과 유사하게 여러 가지 조건 중에 만족하는 부분 하나를 수행한다.

Syntax

```
<x:choose>  
  <x:when select="XPathExpression">  
    body content  
  </x:when>  
  <x:otherwise>  
    conditional block  
  </x:otherwise>  
</x:choose>
```

항목	설명
select	조건을 판별하기 위한 Xpath 표현식이다.

xml(x:forEach)

- XPath 식을 평가하고 결과에 대해 **반복** 문을 **수행**한다.

Syntax

```
<x:forEach [var="varName"] select="XPathExpression">  
    body content  
</x:forEach>
```

항목	설명
var	저장할 변수 이름이다.
select	Xpath 표현식을 사용하여 반복문에 사용된다.

xml(x:parser)

- XML 문서를 파싱한다.

Syntax

```
<x:parser xml="XMLDocument" {var="var" [scope="{page|request|session|application}"]}
[systemId="systemId"] [filter="filter"]> />
```

항목	설명
var	파싱할 xml 문서를 저장할 변수 이름이다.
scope	var 속성의 공유 범위의 유효 영역을 설정한다.
systemId	파싱이 되고 있는 문서의 URI를 나타낸다.
filter	파싱하기 전에 저장된 필터의 내용을 걸러 내는 경우에 사용한다.

xml(x:transform)

- XML 문서를 XSL 스타일시트를 이용하여 새로운 문서로 변형시키는 역할을 한다.

Syntax

```
<x:transform var="변수_이름" scope="어떤_영역" result="생성된결과를저장할변수이름"
doc="source" xslt="XSLStyleSheet"/>
```

항목	설명
var	파싱할 xml 문서를 저장할 변수 이름이다.
scope	var 속성의 공유 범위의 유효 영역을 설정한다.
result	생성된 결과를 저장하는 변수이다. 이 속성은 javax.xml.transform.Result 타입을 반환한다.
doc	xml 문서의 입력을 입력한다.
xslt	xsl 스타일 시트의 이름을 입력한다.

xml(x:param)

- 태그 사이에 파라미터 값을 전달하기 위하여 사용한다.

Syntax

```
<x:param name="이름" value="값" />
```

항목	설명
name	전달될 파라미터의 이름을 입력한다.
value	전달될 파라미터의 값을 입력한다.

형태

1. 태그 내용이 없는 경우

```
<x:param name="name" value="value" />
```

2. 태그 내용이 있는 경우

```
<x:param name="name">  
    parameter value  
</x:param>
```

함수(Functions)

카페

- EL에서 사용할 수 있는 함수들의 library이다.
- 주로 문자열을 처리하는 일을 수행한다.
- java.lang.String 클래스에 있는 메소드와 유사한 기능을 제공한다.

Syntax

`${fn:함수명(인자 목록)}`

- [표] JSTL이 제공하는 주요 함수는 다음과 같다.

함수	설명
length(obj)	obj가 List와 같은 Collection인 경우 저장된 항목의 개수를 리턴하고, obj가 문자열일 경우 문자열의 길이를 리턴한다.
toUpperCase(str)	str을 대문자로 변환한다.
toLowerCase(str)	str을 소문자로 변환한다.
substring(str, idx1, idx2)	str.substring(idx1, idx2)의 결과를 리턴한다. idx2가 -1일 경우 str.substring(idx1)과 동일하다.
trim(str)	str 좌우의 공백 문자를 제거한다.

함수(Functions)

- [표] JSTL이 제공하는 주요 함수는 다음과 같다.
- 참조 사이트
 - http://docs.oracle.com/cd/E17802_01/products/products/jsp/jstl/1.1/docs/tlddocs/index.html

함수	설명
<code>replace(str, src, dest)</code>	str에 있는 src를 dest로 변환한다.
<code>split(str1, str2)</code>	str2로 명시한 글자를 기준으로 str1을 분리해서 배열로 리턴한다.
<code>escapeXml(str)</code>	XML의 객체 참조에 해당하는 특수 문자를 처리한다. 예를 들어, ' & '는 ' & amp; '로 변환한다.
<code>String subStringBefore(str1, str2)</code>	String에서 subString이 나타나는 이전의 부분에 있는 문자열을 return
<code>String subStringAfter(str1, str2)</code>	String에서 subString이 나타나는 이후의 부분에 있는 문자열을 return
<code>int indexOf(str1, str2)</code>	string에서 substring이 처음 나타나는 인덱스를 return
<code>boolean startsWith(string, substring)</code>	string이 substring을 포함하면 true
<code>boolean endsWith(string, suffix)</code>	string이 suffix로 끝나면 true
<code>boolean contains(str1, str2)</code>	str1이 str2를 포함하면 True, 아니면 false
<code>boolean containsIgnoreCase(str1, str2)</code>	대소문자 구별 하지 않고 ...
<code>String join(array, str2)</code>	array에 저장된 문자열을 합친다. 이때 각 문자열 사이에는 str2가 붙는다

함수(Functions)

- 체크 박스 중 조건에 만족하는 checkbox 항목만 체크 설정 예시
- `${bean.hobby}`의 값이 "독서,운동,퀵트"라고 가정하면 다음과 같은 코드를 작성할 수 있다.

코드 예시

```
<c:if test="${fn:contains(bean.hobby, '운동')} == true">  
  <input type="checkbox" name="hobby" id="hobby1" value="운동" checked="checked">운동  
</c:if>  
<c:if test="${fn:contains(bean.hobby, '운동')} == false">  
  <input type="checkbox" name="hobby" id="hobby1" value="운동">운동  
</c:if>
```

취미 ☒ 독서 ☒ 운동 ☐ 음악감상 ☒ 퀵트

function 라이브러리 사용하기

- 파일 이름 : fnFunctionExam.jsp
- 함수 기능들에 대하여 살펴 보는 연습 문제이다.
- 자바의 String 클래스와 유사한 기능을 수행하는 태그이다.

```
<!-- 변수 str1, str2에 문자열을 설정한다. -->
<c:set var="str1" value="Hello Java!" />
<c:set var="str2" value="Jav" />
```

```
<!-- 변수 mytoken에 문자열을 설정한다. -->
<c:set var="mytoken" value="1/2/3/4/5/6/7/8/9/10" />
```

항목 설명	실행 결과
문자열 길이	14
대문자	HELLO JAVA!
소문자	hello java!
[Jav] 단어만 추출하기	Jav
[Jav] 단어 이후의 문자열만 추출	a!
[Jav] 단어 이전의 문자열만 추출	Hello
공백 문자열은 잘라 내기	Hello Java!
스페이스를 -으로 치환하기	Hello-Java!---
단어 [Jav]의 발견 위치	6
[He]라는 단어로 시작하는가?	true
[abc]이라는 단어로 끝나는가?	false
[Jav] 단어의 포함 여부	true
[Jav] 단어의 포함 여부(대소문자 구분 않고)	true

mytoken 문자열을 /로 잘라내기 예시
총합 : 55

join(array, "-") = "1-2-3-4-5-6-7-8-9-10"

function 라이브러리 사용하기

- 파일 이름 : fnFunctionExamOld.jsp

```
문자열 길이 : 36
대문자 : "동해물과 <백두산>이 마르고 닳도록... "
소문자 : = "동해물과 <백두산>이 마르고 닳도록... "
[백두산] 단어만 추출하기 : "백두산"
[백두산] 단어 이후의 문자열만 추출 : ">이 마르고 닳도록... "
[백두산] 단어 이전의 문자열만 추출 : "동해물과 <"
공백 문자열은 잘라 내기 : "동해물과 <백두산>이 마르고 닳도록..."
스페이스를 -으로 치환하기 : "동해물과-<백두산>이-마르고-닳도록...-----"
단어 [백두산]의 발견 위치 : "6"
[동해]라는 단어로 시작하는가 : "true"
[닳도록]이라는 단어로 끝나는가 : "false"
[백두산] 단어의 포함 여부 : "true"
[백두산] 단어의 포함 여부(대소문자 구분 알고) : "true"

mytoken 문자열을 /로 잘라내기 예시
총합 : 55

join(array, "-") = "1-2-3-4-5-6-7-8-9-10"
동해물과 <백두산>이 마르고 닳도록...
escapeXml(str1) = "동해물과 <백두산>이 마르고 닳도록... "
```

function 라이브러리 사용하기2

- 자바의 String 클래스와 유사한 기능을 수행하는 태그이다.

출력 결과

문자열 : To be or not to be, that is the question.
대문자: TO BE OR NOT TO BE, THAT IS THE QUESTION.
소문자 : to be or not to be, that is the question.
is의 위치는? 25
be를 were로 바꾸면? To were or not to were, that is the question.
문자열의 길이는? 41

```
<!-- 변수 message에 문자열을 설정한다. -->
<c:set var="message" value="To be or not to be, that is the question." />

<body>
    문자열 : ${message} <br>
    대문자: ${fn:toUpperCase(message)} <br>
    소문자 : ${fn:toLowerCase(message)} <br>
    is의 위치는? ${fn:indexOf(message, "is")} <br>
    be를 were로 바꾸면? ${fn:replace(message, "be", "were")} <br>
    문자열의 길이는? ${fn:length(message)} <br>
</body>
```

파일 이름	설명
fnFunctionExam2.jsp	함수 기능들에 대하여 살펴 보는 페이지이다.