

JSP Action Tag

▶ JSP Action Tag

XML기술을 이용하여 기존의 JSP문법을 확장하는 매커니즘을 제공하는 태그
 웹 브라우저에서 실행되는 것이 아니라 웹 컨테이너에서 실행되고
 결과만 브라우저로 보냄

	표준 액션 태그	커스텀 액션 태그
사용법	JSP페이지에서 바로 사용 태그 앞에 jsp접두어가 붙음	별도의 라이브러리 설치 필요 라이브러리 선언에 맞는 접두어가 붙음
사용 예시	<code><jsp:include page="../sample.jsp"/></code>	<code><c:set var="count" value="0"/></code>

* 액션 태그 접두어란 태그 이름 앞에 'ooo:' 형식으로 제공하는 태그의 그룹을 지정하는 것을 뜻함

▶ 표준 액션 태그

JSP에서 기본으로 제공하는 태그

태그 명	설명
jsp:include	현재 페이지에 특정 페이지를 포함할 때 사용
jsp:forward	현재 페이지 접근 시 특정 페이지로 이동 (request.forward()와 동일)
jsp:param	<jsp:include>, <jsp:forward>의 하위 요소로 사용되며 해당 페이지에 전달할 값을 기록할 때 사용
jsp:useBean	Java Bean객체를 사용하기 위한 태그로 JSP에서 사용하는 VO, DTO, Entity와 동일
jsp:setProperty	Java객체 사용 시 Setter와 동일한 역할
jsp:getProperty	Java객체 사용 시 Getter와 동일한 역할

▶ 표준 액션 태그

✓ jsp:include

<%@ include file="파일 명" %>과 쓰임이 동일하나
jsp파일이 java파일로 변환될 때 삽입되는 <%@ include %>와는 달리
jsp파일이 java파일로 바뀌고 컴파일이 완료되어 런타임 시 삽입

✓ 사용 문법과 예시

- 문법

```
<jsp:include page="파일 명" flush="true"/>
```

- 예시

```
<jsp:include page="./header.html">
```

```
    <jsp:param name="str" value="안녕하세요">
```

```
</jsp:include>
```

▶ 표준 액션 태그

✓ jsp:forward

하나의 JSP페이지에서 다른 JSP페이지로 요청 처리를 전달할 때 사용
전달하는 페이지에서 request, response객체가 같이 전달되며
URL은 변경되지 않음

✓ 사용 문법과 예시

- 문법

```
<jsp:forward page="파일 명"/>
```

- 예시

```
<% if(str.equals("A")) { %>  
    <jsp:forward page="/A_Class.jsp">  
<% } else { %>  
    <jsp:forward page="/B_Class.jsp">  
<% } %>
```

▶ 표준 액션 태그

✓ jsp:useBean

java class를 참조하여 빈 객체를 생성하고
setProperty와 getProperty를 통해 값 저장 및 조회 가능
이미 같은 이름의 객체가 생성된 경우 기존의 객체 참조

✓ 사용 문법과 예시

- 문법

```
<jsp:useBean id="객체 명" class="패키지 명. 클래스 명" scope="범위 지정자"/>
```

- 예시

```
<jsp:useBean id="m" class="member.model.vo.Member" scope="request">  
    <jsp:setProperty name="m" property="member_name" value="김유신"/>  
    <jsp:setProperty name="m" property="member_age" value="79"/>  
</jsp:useBean>
```

EL/JSTL

▶ EL

Expression Language의 약자로 JSP 2.0버전에서 추가 됨
<%= %>, out.print()와 같이 JSP에 쓰이는 Java코드를 간결하게 사용하는
방법으로 화면에 표현하고자 하는 코드를 \${ value } 형식으로 표현하여 작성

✓ 사용 문법과 예시

- 문법

`${ value }`

- 예시

`<%= request.getParameter("name") %>`

`${ param.name }`

▶ EL 연산자 기호

	일반 연산자	EL 기호 연산자
덧셈, 뺄셈	+, -	+, -
곱셈, 나눗셈	*, /	*, div
나머지 연산	%	mod
and 연산	&&	and
or 연산		or
! 연산	!	not
~보다 작다	<	lt (less than)
~보다 크다	>	gt (greater than)
작거나 같다	<=	le (less or equal)
크거나 같다	>=	ge (greater or equal)
~와 같다	==	eq (equal)
~와 다르다	!=	ne (not equal)
null 값 처리	value == null	empty

* (): ~의 원말

▶ EL 연산자 우선 순위

순위	기호
1순위	[], .
2순위	()
3순위	not, !, empty
4순위	*, /, div, %, mod
5순위	+, -
6순위	<, <=, >, >=, lt, le, gt, ge
7순위	==, !=, eq, ne
8순위	&&, and
9순위	, or
10순위	? : (삼항 연산자)

▶ EL 내장 객체

객체 명	설명
pageScope	page영역 객체에 접근
requestScope	request영역 객체에 접근
sessionScope	session영역 객체에 접근
applicationScope	application영역 객체에 접근
param	전달된 파라미터 값을 받아올 때 사용
paramValues	전달된 파라미터들을 배열로 받아올 때 사용
header	사용자의 특정 헤더 정보를 받아올 때 사용
headerValues	사용자의 헤더 정보를 배열로 받아올 때 사용
cookie	\${cookie.key명}으로 쿠키 값 조회
initParam	초기 파라미터 조회
pageContext	pageContext경로 조회

▶ JSTL

JSP Standard Tag Library의 약자로 JSP에서 사용하는 커스텀 태그
공통으로 사용하는 코드의 집합을 사용하기 쉽게 태그화 하여
표준으로 제공한 것을 말함

✓ 선언 방식과 예시

- 선언 방식

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

- 예시

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  
<c:out value="${'Welcome to javaTpoint'}"/>
```

▶ JSTL 라이브러리 등록

<https://www.javapoint.com/jsppages/src/jstl-1.2.jar>

상위 링크를 통해 jstl-1.2.jar파일 설치 후 이클립스 프로젝트 내
WebContent/WEB-INF/lib 내에 등록하고 사용하고자 하는 jsp파일에서 선언

▼ firstProject

> .settings

▼ src

> test

▼ web

> META-INF


> views

▼ WEB-INF

> classes

> jspwork

▼ lib

 jstl-1.2.jar

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jstl/fmt"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSTL Core 테스트</title>
</head>
<body>

</body>
</html>
```

▶ JSTL 태그 종류

태그 명	설명
Core Tags	변수와 url, 조건문, 반복문 등의 로직과 관련된 JSTL문법 제공 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
Formatting Tags	메시지 형식이나 숫자, 날짜 형식과 관련된 포맷 방식 제공 <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
Function	trim, substring과 같은 여러 문자열 처리 함수 제공 <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
XML Tags	데이터의 XML파싱 처리 등 XML문서를 화면으로 읽어오는데 필요한 라이브러리 <%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
SQL Tags	페이지 내에서 Database를 연동하고 필요한 쿼리를 실행할 수 있는 라이브러리 <%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>

▶ JSTL Core Tags

✓ <c:set>

변수를 선언하고 나서 그 변수에 초기 값 대입
자바에서 변수를 선언하는 방식과 비슷

Java 변수 선언 방식 : `int num = 100;`

<c:set> 변수 선언 방식 : `<c:set var="num" value="100" />`

✓ 사용법

<c:set>에서 변수 타입은 별도로 선언하지 않지만 초기 값은 반드시 기술해야 하며 <c:set>으로 선언한 변수는 EL 식 안에서 사용 가능하지만 JSP <% %>같은 스크립틀릿 요소에서는 사용 불가능

`<c:set var="num" value="100" />`

`${num}` * <c:set>에서 선언한 변수는 EL 식 안에서 사용 가능

▶ JSTL Core Tags

✓ <c:set> 사용법

<c:set>에서 변수 타입은 별도로 선언하지 않지만 초기 값은 반드시 기술해야 하며 <c:set>으로 선언한 변수는 EL 식 안에서 사용 가능하나 <% %>같은 스크립틀릿 요소에서는 사용 불가능 (반대로는 가능)

```
<c:set var="num" value="100" />
```

`${num}` * <c:set>에서 선언한 변수는 EL 식 안에서 사용 가능

```
<% int num1 = 10, num2 = 20; %>
```

```
<c:set var="sum" value="<%= num1 + num2 %>" />
```

`${sum}`

▶ JSTL Core Tags

✓ <c:set> scope 속성

scope속성을 이용하면 page영역 뿐만 아니라 request, session, application 영역에 속성 저장 가능 (미 설정 시 기본 값은 page)

```
<c:set var="num" value="100" scope="request" />
```

✓ <c:set> 태그 배열 설정

<c:set>의 body부분에 ", "를 이용해서 배열이나 Collection처럼 여러 개의 값 지정 가능 * body에 있는 값이 array변수에 할당된 변수 값이 됨

```
<c:set var="array" scope="request">  
    yellow, blue, pink, red, green  
</c:set>
```

▶ JSTL Core Tags

✓ <c:remove> 태그 배열 속성

<c:set>으로 선언한 변수는 page, request, session, application영역에 속성으로 저장되기 때문에 삭제해야 할 필요가 있을 때 사용

```
<c:remove var="num1" scope="request">
```

* request영역에 num1이라는 변수 제거, 만일 scope속성을 정의하지 않으면 page, request, session, application영역에 저장되어 있는 num1이라는 이름의 속성을 모두 찾아 제거

▶ JSTL Core Tags

✓ <c:out>

데이터 출력할 때 사용하는 태그로 <, >, & 특수 문자를 자동으로 이스케이프 시퀀스(escape sequence)로 바꿔줌
출력할 데이터에 포함된 특수 문자를 태그의 일부로 인식시키고 싶을 땐 escapeXml이라는 속성을 추가하고 false값 지정

```
<c:out value="<title>은 <head>의 하위태그이다."/>
```

* <title>과 <head>는 웹 브라우저가 해석하지 않고 기술한 대로 화면에 나타남

```
<c:out value="<h2>데이터 출력</h2>" escapeXml="false"/>
```

▶ JSTL Core Tags

✓ <c:if>

자바 프로그램의 if문과 비슷한 역할을 하는 태그로 조건식은 test라는 속성 값으로 지정 (이 때 조건식은 반드시 EL형식으로 기술)

```
<c:if test="${num1 > num2}">
```

num1이 더 큼니다.

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

▶ JSTL Core Tags

✓ <c:choose>

자바 프로그램의 switch문과 비슷한 역할을 하는 태그로
<c:when>, <c:otherwise> 태그와 함께 사용되는데,
각각 switch문의 case, default절과 비슷한 역할

```
<c:choose>
```

```
    <c:when test="{num == 0}"> * 조건식
```

```
        처음 뵙겠습니다.<br>
```

```
    </c:when>
```

```
    <c:when test="{num == 1}">
```

```
        다시 뵙게 되어 반갑습니다.<br>
```

```
    </c:when>
```

```
    <c:otherwise> * 아무 조건도 만족하지 못 할 경우 실행
```

```
        안녕하세요.<br>
```

```
    </c:otherwise>
```

```
</c:choose>
```

▶ JSTL Core Tags

✓ <c:forEach>

자바의 for, for-in문에 해당하는 기능 제공

```
<c:forEach begin="1" end="10" items="${list}" var="value">
```

반복문


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

✓ 속성

items	반복할 객체 명(Collection 객체)
begin	반복이 시작할 요소 번호(0...n)
end	반복이 끝나는 요소 번호
step	반복할 횟수 번호
var	현재 반복 횟수에 해당하는 변수 이름
varStatus	현재 반복에 해당하는 객체 요소

▶ JSTL Core Tags

✓ <c:forEach>의 varStatus

current	현재 반복 회수	상태 값 명.current
index	반복 라운드의 제로기반 인덱스	상태 값 명.index
count	반복 라운드의 1 기반 인덱스	상태 값 명.count
first	현재 라운드가 반복을 통한 첫 번째임을 의미	상태 값 명.first
last	현재 라운드가 반복을 통한 마지막 번째임을 의미	상태 값 명.last

✓ 사용 예

```

<c:forEach items="${booklist}" var="book" varStatus="status">
    * Collection 객체 명                                * 상태 값 명
    <tr>
        <td> <c:out value="${status.count}"/> </td>
        <td> <c:out value="${book.name}"/> </td>
    </tr>
</c:forEach>

```

▶ JSTL Core Tags

✓ <c:forTokens>

문자열에 포함된 구분자를 통해 토큰을 분리해 반복 처리
items속성에는 토큰을 포함하는 문자열을 지정하고
delims속성에는 토큰을 분리하는데 사용할 구획 문자 기술

```
<c:forTokens var="color" items="yellow blue pink red green" delims=" ">  
    ${color} <br>  
</c:forTokens>
```


▶ JSTL Core Tags

✓ <c:url>

url경로를 생성하고 해당 url의 param속성을 선언하여
쿼리 스트링을 정의할 수 있는 태그로 해당 태그를 통해 url경로와
관련 쿼리 스트링 값을 미리 설정하여 제어 가능

```
<c:url var="url" value="jstl.jsp">
```

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

```
</c:url>
```

```
<a href="${url}">jstl.jsp</a>
```

▶ JSTL Formatting Tags

✓ <fmt:formatNumber>

표현하고자 하는 숫자의 포맷을 통화 기호, ',' 표시, % 표시 등 원하는 쓰임에 맞게 지정 가능

```
<c:set var="number" value="12300.12"/>
```

```
<p> 포맷 방식 세 자리 구분 : <fmt:formatNumber value="{number}"  
    type="number" groupingUsed="true"/> * 12,300.12
```

```
</p>
```

```
<p>포맷 방식 통화 기호 : <fmt:formatNumber value="{number}"  
    type="currency"/> * ₩12,300
```

```
</p>
```

```
<p>포맷 방식 백분율 : <fmt:formatNumber value="{number}"  
    type="percent" /> * 012%
```

```
</p>
```

▶ JSTL Formatting Tags

✓ <fmt:formatNumber>

maxIntegerDigits 및 minIntegerDigits의 속성으로 표시하고자 하는
수의 단위 표현 가능

숫자가 지정한 최대 값을 초과할 경우 해당 자릿수만큼 표시

```
<fmt:formatNumber type="number" maxIntegerDigits="4"  
value="${number}"/>
```

* 2300.12, 숫자 범위가 지정한 부분을 넘어 앞자리 1은 표시되지 않음

▶ JSTL Formatting Tags

✓ <fmt:formatNumber>

maxFractionalDigits 및 minFractionalDigits의 속성은 소수 자릿수를 지정할 수 있으며 숫자가 최소 자릿수를 초과할 시 자동 반올림 패턴 속성을 사용하여 숫자 포맷 방법 지정 가능

```
<fmt:formatNumber type="number" pattern="000.00"
```

```
maxFractionDigits="2" value="12300.125"/> * 12300.12
```

```
<fmt:formatNumber type="number" pattern="###.###"
```

```
minFractionDigits="3" value="12300.1"/> * 12300.100
```

▶ JSTL Formatting Tags

✓ <fmt:formatDate>

날짜나 시간의 포맷 방식을 지정하여 화면에 보여줄 때 사용
value 속성으로는 java.util.Date()객체를 사용해야 함

```
<c:set var="date" value="<%= new java.util.Date() %>"/>
```

```
<fmt:formatDate type="time" value="${date}"/> * 날짜 포맷 태그 사용
```

▶ JSTL Formatting Tags

✓ <fmt:formatDate>

type지정 방식에 따라 날짜, 시간 모두를 표시할 수 있고
dateStyle, timeStyle속성으로 보여줄 포맷 표기 방식 설정 가능

```
<fmt:formatDate type="time" value="${date}"/> * 오전 4:40:52
```

```
<fmt:formatDate type="date" value="${date}"/> * 2018. 1. 10
```

```
<fmt:formatDate type="both" value="${date}"/> * 2018. 1. 10 오전 4:40:52
```

```
<fmt:formatDate type="both" dateStyle="short" timeStyle="short"  
value="${date}"/> * 18. 1. 10 오전 4:40
```

```
<fmt:formatDate type="both" dateStyle="medium" timeStyle="medium"  
value="${date}"/> * 2018. 1. 10 오전 4:40:52
```

```
<fmt:formatDate type="both" dateStyle="long" timeStyle="long"  
value="${date}"/> * 2018년 1월 10일 (수) 오전 4시 40분 52초
```

▶ JSTL Formatting Tags

✓ <fmt:setLocale>

지역 설정을 통해 통화 기호나 시간 대역을 다르게 설정 가능

* 국가-지역 설정 참고 주소 : <http://www.lingoes.net/en/translator/langcode.html>

```
<h1>대한민국</h1>
```

```
금액 : <fmt:formatNumber value="1000000" type="currency"/> <br>
```

```
일시 : <fmt:formatDate value="${date}" type="both" dateStyle="full" timeStyle="full"/>
```

```
<h1>미국</h1>
```

```
<fmt:setLocale value="en_us"/>
```

```
금액 : <fmt:formatNumber value="1000000" type="currency"/> <br>
```

```
일시 : <fmt:formatDate value="${date}" type="both" dateStyle="full" timeStyle="full"/>
```

```
<h1>일본</h1>
```

```
<fmt:setLocale value="ja_jp"/>
```

```
금액 : <fmt:formatNumber value="1000000" type="currency"/> <br>
```

```
일시 : <fmt:formatDate value="${date}" type="both" dateStyle="full" timeStyle="full"/>
```

▶ JSTL Function

문자열 처리에 관한 메소드들을 EL형식에서 사용할 수 있게 하는 라이브러리
다른 JSTL태그들과는 다르게 `${fn:메소드 명(문자열)}`의 형식으로 EL태그에
직접 대입하여 사용

✓ 선언 방식과 예시

- 선언 방식

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

- 예시

```
<c:set var="theString" value="I am s test String"/>
```

```
<c:if test="${fn:contains(theString, 'test')}">
```

```
    <p>Found test String</p>
```

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


▶ JSTL Function

함수 명	설명
fn:contains(str, 'text')	str에 두 번째 인자 값의 내용이 포함되어 있는지 확인
fn:containsIgnoreCase(str, 'text')	str에 대소문자 구분 없이 'text'값이 포함되어 있는지 확인
fn:startsWith(str, 'text')	str이 'text'로 시작하는지 확인
fn:endsWith(str, 'text')	str이 'text'로 끝나는지 확인
fn:escapeXml(str)	str에 xml태그가 포함됐다면 해당 태그까지 화면에 출력
fn:indexOf(str, 'text')	str내에 'text'의 첫 글자 시작 위치 반환(0부터 시작)
fn:length(str)	str 길이 반환
fn:replace(str, 'text1', 'text2')	str내의 text1을 찾아 text2로 변경
fn:substring(str, index1, index2)	str에서 index1부터 index2까지의 문자열 반환
fn:split(str, ' ')	str을 ' '구분자를 기준으로 나눠 배열로 만들어 반환
fn:join(str, '-')	배열요소로 나뉘어진 str을 '-'구분자를 붙여 합친 뒤 반환
fn:trim(str)	str 값의 좌우 공백 제거

▶ JSTL XML Tags

XML태그를 사용하기 위해 별도의 라이브러리 설치 후 WEB-INF/lib에 추가

✓ 설치

<http://apache.mirror.cdnetworks.com/xalan-j/binaries/>

xalan.jar와 XercesImpl.jar 설치 후

<%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>로 선언

* 설치하지 않으면 java.lang.NoClassDefFoundError 발생

▶ JSTL XML Tags

✓ <x:out>

JSP <%= %>와 유사하나 XPath 표현식 사용
select속성에 사용하는 xml객체 명시

```
<c:set var="xmlbook">
    <book>
        <name>Greate Mistry</name>
        <author>NUHA</author>
        <price>20000</price>
    </book>
</c:set>
<x:parse xml = "${xmltext}" var = "output"/>
<b>책 정보 출력</b>:
<x:out select = "$output/book/name" />
```

* select속성으로 접근 시 해당 객체의 요소에 파일 경로(xpath)처럼 접근

▶ JSTL XML Tags

✓ <x:parse>

객체로 사용할 xml파일을 Xpath로 접근이 가능하도록 파싱 처리
var로 선언한 변수 명을 <x:out>으로 접근 시
EL형식 \${변수 명}이 아닌 \$변수 명/속성 명의 Xpath형식으로 접근

```
<c:import var="bookInfo" url="/book.xml" />
```

* 이전에 선언한 xml을 별도 문서로 만들어 import

```
<x:parse xml = "${bookInfo}" var = "output"/>
```

* 추가한 변수는 xml속성에, Xpath로 접근할 변수 명은 var에 선언

책 정보 출력:

```
<x:out select = "$output/book/name" /> ,
```

```
<x:out select = "$output/book/author" /> ,
```

```
<x:out select = "$output/book/price" />
```

▶ JSTL XML Tags

✓ <x:set>

xml로부터 데이터를 받아와 변수에 저장하는 태그로
값을 담은 변수는 <x:out>태그로 접근 가능

```
<c:import var="bookInfo" url="/book.xml" />
```

```
<x:parse xml = "${bookInfo}" var = "output"/>
```

```
<x:set var="price" select="$output//price"/>
```

* \$output//price로 경로를 지정하면 해당 이름을 가진 태그로 직접 찾아감

책 가격 출력:

```
<x:out select = "$price" />
```

▶ JSTL XML Tags

✓ <x:if>

<c:if> 태그와 유사하나 test속성 대신 select속성으로 조건 지정

```
<c:import var="bookInfo" url="/book.xml"/>
```

```
<x:parse xml = "${bookInfo}" var = "output"/>
```

```
<x:if select="$output//book"> * select속성과 Xpath표현식으로 조건 작성  
    book객체가 최소 한 개 이상입니다.
```

```
</x:if>
```

```
<br>
```

```
<x:if select="$output/book/price > 20000"/>  
    책 가격이 높은 편입니다.
```

```
</x:if>
```

▶ JSTL XML Tags

✓ <x:choose>

<c:choose> 태그와 유사하나 test속성 대신 select속성 사용

```
<c:import var="bookInfo" url="/book.xml"/>
```

```
<x:parse xml = "${bookInfo}" var = "output"/>
```

```
<x:choose>
```

```
    <x:when select="$output/book/author = '남궁성'">  
        저자는 남궁성입니다.
```

```
    </x:when>
```

```
    <x:otherwise>
```

```
        저자는 남궁성이 아닙니다.
```

```
    </x:otherwise>
```

```
</x:choose>
```

▶ JSTL XML Tags

✓ <x:forEach>

<c:forEach> 태그와 유사하나 item속성 대신 select속성으로 조건 지정

```
<c:import var="bookInfo" url="/book.xml"/>
```

```
<x:parse xml = "${bookInfo}" var = "output"/>
```

```
<ul>
```

```
    <x:forEach select="$output/book" var="value">
```

```
        <li>책 이름 : <x:out select="value"/></li>
```

```
    </x:forEach>
```

```
</ul>
```