



JSP 2.3 & Servlet 3.1

EL & JSTL

- 김근형 -

EL(Expression Language)

- ▶ EL(Expression Language)
 - ▶ EL의 개념은 표현 언어를 이해하고 속성 값들을 편리하게 출력하기 위해 제공된 언어
 - ▶ `<%= %>` , `out.println()`과 같은 자바코드를 더 이상 사용하지 않고 좀더 간편하게 출력을 지원하기 위한 도구.
 - ▶ 배열이나 컬렉션에서도 사용되고, JavaBean의 프로퍼티에서도 사용된다

EL(Expression Language)

▶ EL 특징

- ▶ 파라미터 값이 null 이어도 상관없다
- ▶ 파라미터 값의 파싱을 신경 쓰지 않아도 된다.
- ▶ 변수와 연산자를 포함하고 함수를 호출할 수 있다.
- ▶ JSP의 Scope영역(page, request, session, application)에 저장된 모든 속성 및 자바 빈을 표현 언어의 변수로서 사용할 수 있다.
- ▶ 내장 객체를 지원한다.

EL(Expression Language)

▶ EL 작성 방법

- ▶ 표현 언어는 항상 '\${'로 시작해서 '}'로 끝난다.
- ▶ 표현 언어의 표현식 안에 연산식도 쓸 수 있다.
- ▶ 표현 언어 표현식에는 브라켓 연산자([])를 사용할 수 있다.
- ▶ 표현 언어는 동적으로 값을 받도록 JSTL이나 커스텀 태그의 JSP 액션의 속성값을 지정할 때도 사용할 수 있다.
- ▶ 객체에 접근할 경우 “저장된 객체 속성 명[파라미터명]”으로 접근이 가능하다

EL(Expression Language)

▶ EL 예제

```
<meta charset="UTF-8">
<title>Insert title here</title>
<%
    String s = "안녕하세요";
    int num = 1;
    request.setAttribute("num", num);
    request.setAttribute("s", s);

    Article art = new Article();
    art.setNum(5);

    request.setAttribute("article", art);
%>
</head>
<body>
    <p>${s}</p>
    <p>${num+1}</p>
    <p>${article["num"]}</p>
</body>
</html>
```

EL(Expression Language)

▶ EL 연산자

연산자	설명	연산자	설명
.	빈의 프로퍼티 또는 맵 엔트리에 접근	== , =	같다
[]	배열이나 리스트의 엘리먼트 접근	!=	같지 않다
()	괄호, 표현식의 연산 순서를 바꿔서 연산 시 사용	< , lt	보다 작다
a?b:c	삼항 연산 조건	> , gt	보다 크다
+	덧셈	<= , le	작거나 같다
-	뺄셈	>= , ge	크거나 같다
*	곱셈	&& , and	논리 and
/ , div	나누기	, or	논리 or
% , mod	나머지	! , not	논리 not

EL(Expression Language)

▶ EL 연산자

```
<ul>
  <li>
    <p>표현식 = 값
  <li>
    <p>\${2 + 5} = ${2 + 5}
  <li>
    <p>\${4 / 5} = ${4 / 5}
  <li>
    <p>\${7 mod 5} = ${7 mod 5}
  <li>
    <p>\${2 < 3} = ${2 < 3}
  <li>
    <p>\${3.1 le 3.2} = ${3.1 le 3.2}
  <li>
    <p>\${(5 > 3) ? 5 : 3} = ${ (5 > 3) ? 5 : 3}
</ul>
```


EL(Expression Language)

▶ EL 연산자

```
<% request.setCharacterEncoding("utf-8");%>

<form action="elEx2.jsp" method="post">
  <ul>
    <li><label for="name">이름</label>
      <input type="text" id="name" name="name"
              value="${param['name']}">
      <input type="submit" value="확인">
    <li><p>이름은: ${param.name} 입니다.
  </ul>
</form>
```


EL(Expression Language)

- ▶ 동일한 이름 다른 스코프에 있는 값의 접근 예제
 - ▶ 각각의 scope에 접근하고 싶다면 pageScope, requestScope, sessionScope, applicationScope를 붙여 사용한다.

```
<body>
  <%
    request.setAttribute("abc", "request 저장");
    session.setAttribute("abc", "session 저장");
  %>
  <p>${abc}</p>
  <p>${sessionScope.abc}</p>
</body>
```

JSTL(JSP Standard Tag Library)

- ▶ JSTL이란 JSP 표준라이브러리(JSP Standard Tag Library) 이다.
- ▶ JSP에서 자주 사용하는 기능(반복과 조건, 데이터 관리 포맷, XML 조작, 데이터베이스 액세스)을 구현하는 커스텀 태그 라이브러리 모음이다.
- ▶ 시간, 날짜, 숫자의 포맷이나 문자열 가공등의 처리에서 비즈니스로직과 프리젠테이션 로직을 분리할 수 있게 해준다.
- ▶ JSTL은 EL(Expression Language)를 사용하여 표현한다.


JSTL(JSP Standard Tag Library)

▶ JSTL 다운로드

- ▶ <http://tomcat.apache.org/taglibs/>

Apache Taglibs

This project is an open source repository for JSP(tm) Tag Libraries.

In particular, Apache Taglibs hosts the [Apache Standard Taglib](#), an implementation of the [JSP Standard Tag Library \(JSTL\) specification](#) . Versions 1.0, 1.

Formerly Jakarta Taglibs

Apache Taglibs was originally called Jakarta Taglibs. When the project changed name, and moved over to coexist with the Tomcat project, many of the orig

News

2015/02/20	[Standard] The Apache Standard Taglib 1.2.3, a minor bugfix version, has been released.
2014/01/02	[Standard] The Apache Standard Taglib 1.2.1, an implementation of JSTL 1.2, has been released.
2009/06/07	[ALL] The Standard and RDC Taglibs have moved over to Tomcat. All other taglibs back at Jakarta have been deprecated.

JSTL(JSP Standard Tag Library)

▶ JSTL 다운로드

▶ <http://tomcat.apache.org/taglibs/>

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

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)

- ▼ WEB-INF
- ▼ lib
 - taglibs-standard-compat-1.2.5.jar
 - taglibs-standard-impl-1.2.5.jar
 - taglibs-standard-jstlel-1.2.5.jar
 - taglibs-standard-spec-1.2.5.jar

JSTL(JSP Standard Tag Library)

▶ JSTL 라이브러리의 uri와 prefix 및 제공기능

라이브러리	기능	URI 식별자	접두어
코어	일반 프로그램 언어에서 제공하는 변수선언, 조건/제어/반복문등의 기능을 제공한다.	http://java.sun.com/jsp/jstl/core	c
포매팅	숫자,날짜,시간을 포매팅 하는 기능과 국제화, 다국어 지원 기능을 제공한다.	http://java.sun.com/jsp/jstl/fmt	fmt
함수	문자열을 처리하는 함수를 제공한다.	http://java.sun.com/jsp/jstl/functions	fn
데이터베이스	데이터베이스의 데이터를 입력/수정/삭제/조회하는 기능을 제공한다.	http://java.sun.com/jsp/jstl/sql	sql
XML처리	XML 문서를 처리할 때 필요한 기능을 제공한다.	http://java.sun.com/jsp/jstl/xml	x

JSTL(JSP Standard Tag Library) - Core

▶ JSTL Core 라이브러리

- ▶ 변수의 선언이나 삭제와 같이 변수와 관련된 작업, if for문 등과 같은 제어문 URL 처리와 그 밖에 예외 처리 및 화면 출력 등에 쓰인다.

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

태그이름	설명
<c:set />	변수의 선언 및 제거
<c:remove />	
<c:out />	변수의 출력
<c:catch />	예외 처리
<c:if />	조건문 (else는 없다)
<c:choose />	Switch문과 비슷
<c:when />	
<c:otherwise />	
<c:forEach />	반복문
<c:forTokens />	구분자로 분할하여 반복문
<c:url />	URL 생성
<c:param />	파라미터 추가
<c:import />	페이지 첨부
<c:redirect />	URL 이동

JSTL(JSP Standard Tag Library) - Core

▶ <c:set> : setAttribute()와 같은 역할

<c:set var="변수명" value="변수에 넣을 값" property="자바빈 객체 or Map 객체 값을 설정할
프로퍼티 명" scope="변수 공유 범위" />

```
<!-- scope 속성은 선택적이며 page로 기본 설정되어 있다. -->
<c:set var="변수명" value="할당된 값"
      scope="변수의 유효 범위 page|request|session|application" />

<!-- 간단 사용 예 -->
<c:set var="country" value="Korea" scope="request" />
<c:set var="intArray" value="<%=new int[] {1,2,3,4,5} %>" />
```


JSTL(JSP Standard Tag Library) - Core

- ▶ `<c:remove>` : `removeAttribute()`와 같은 역할

```
<c:remove var="변수명" scope="변수 공유 범위" />
```

```
<c:remove var="country" scope="request" />
```

- ▶ `<c:out>` : '`<%=...>`'와 같다. JSP의 표현식을 대체

```
<c:out var="변수명" default="기본값" escapeXML="true | false" />
```

```
<!-- Syntax -->
<c:out var="변수명" default="기본값" escapeXML="true|false" />

<!-- 간단 사용 예 -->
<p><c:out value="${country}" default="Korea" escapeXml="true"/></p>
<p>${country}</p><p>${intArray[0]}</p>
```

JSTL(JSP Standard Tag Library) - Core

- ▶ `<c:catch>` : Body에서 실행되는 코드의 예외 처리

```
<c:catch var="에러메시지가 포함될 변수명" />
```

```
<c:catch var="ex">  
    <%= 1/0 %>  
</c:catch> <br/>  
Error Msg: ${ex} <br/>
```

JSTL(JSP Standard Tag Library) - Core

▶ <c:if> : 조건문

```
<c:if test="조건 판별식" var="변수명" scope="변수 공유 범위" />
```

```
<!-- Syntax -->
<c:if test="expression" var="name" scope="scope">
body content
</c:if>

<!-- 간단 사용 예 -->
<c:set var="login" value="true" />

<c:if test="${!login}"> <p><a href="/login.ok">로그인</a></p></c:if>
<c:if test="${login}"> <p><a href="/logout.ok">로그아웃</a></p></c:if>

<!-- 아래 예제와 같이 null 비교를 하지 않고 empty 비교를 하면 null과 ""를 동시에 체크할 수 있다. -->
<c:if test="${!empty country}"><p><b>${country}</b></p></c:if>
```

JSTL(JSP Standard Tag Library) - Core

- ▶ `<c:choose />`, `<c:when />`, `<c:otherwise />` : Switch문과 동일, 여러개의 `when` 태그와 하나의 `otherwise` 태그를 가진다
- ▶ `<c:if />` 태그에 `else`가 없으므로 대체식으로도 많이 사용

```
<c:choose>
  <c:when test="조건 판별식"> .... </c:when>
  <c:when test="조건 판별식"> .... </c:when>
  <c:when test="조건 판별식"> .... </c:when>
  <c:otherwise> ... </c:otherwise>
</c:choose>
```

JSTL(JSP Standard Tag Library) - Core

- ▶ <c:choose />, <c:when />, <c:otherwise />

```
<!-- Syntax -->
<c:choose>
  <c:when test="expression">
    body content
  </c:when>
  ...
  <c:otherwise>
    body content
  </c:otherwise>
</c:choose>

<!-- 간단 사용 예 -->
<c:choose>
  <c:when test="${login}">
    <p><a href="/logout.ok">로그아웃</a></p>
  </c:when>
  <c:otherwise>
    <p><a href="/login.ok">로그인</a></p>
  </c:otherwise>
</c:choose>
```

JSTL(JSP Standard Tag Library) - Core

- ▶ `<c:forEach />` : 객체 전체에 걸쳐 반복 실행에 사용

```
<c:forEach var="현재 아이템의 변수명" items="반복 데이터가 있는 아이템 Collection 명"
begin="시작 값, 기본값은 0" end="종료 값" step="증가 값" varStatus="반복 상태 값을 지닌 변수" />
```

- ▶ ※ `varStatus`는 `forEach`의 상태를 알 수 있는 값이 들어 있다.

- ▶ `$(변수.current)` : 현재의 인덱스
- ▶ `$(변수.index)` : 0부터의 인덱스
- ▶ `$(변수.count)` : 1부터의 인덱스
- ▶ `$(변수.first)` : 현재 루프가 처음인지 확인
- ▶ `$(변수.last)` : 현재 루프가 마지막인지 확인
- ▶ `$(변수.begin)` : `forEach`문의 시작 값
- ▶ `$(변수.end)` : `forEach`문의 끝 값
- ▶ `$(변수.step)` : `forEach`문의 증가 값

JSTL(JSP Standard Tag Library) - Core

▶ <c:forEach />

```
<!-- 정수 범위내의 반복 Syntax -->
<c:forEach var="name" varStatus="name"
           begin="expression" end="expression" step="expression">
    body content
</c:forEach>

<!-- 컬렉션 범위내의 반복 Syntax -->
<c:forEach var="name" items="expression" varStatus="name"
           begin="expression" end="expression" step="expression">
    body content
</c:forEach>

<!-- forEach 정수 범위내의 반복 -->
<c:forEach var="i" begin="0" end="10" step="2" varStatus="x">
    <p> i = ${i}, i*i = ${i * i} <c:if test="${x.last}">, last = ${i}</c:if> </p>
</c:forEach>

<!-- forEach 컬렉션 범위내의 반복 -->
<%
    java.util.List list = new java.util.ArrayList();

    java.util.Map map = new java.util.HashMap();
    map.put("color","red");
    list.add(map);

    map = new java.util.HashMap();
    map.put("color","blue");
    list.add(map);

    map = new java.util.HashMap();
    map.put("color","green");
    list.add(map);

    request.setAttribute("list", list);
%>

<c:forEach var="map" items="${list}" varStatus="x">
    <p> map(${x.index}) = ${map.color} </p>
</c:forEach>
```


JSTL(JSP Standard Tag Library) - Core

- ▶ `<c:forTokens />` : 문자열을 구분자(delimiter)로 분할

`<c:forTokens var="현재 아이템의 변수 명" items="반복 데이터가 있는 아이템 Collection 명" delims="구분자, 여러개 지정 가능" begin="시작 값, 기본 값은 0" end="종료 값" step="증가 값" varStatus="반복 상태 값을 지닌 변수" />`

```
<!-- Syntax -->
<c:forTokens var="name" items="expression"
             delims="expression" varStatus="name"
             begin="expression" end="expression" step="expression">
    body content
</c:forTokens>

<!-- 간단 사용 예 -->
<b>
<c:forTokens var="color" items="빨|주|노|초|파|남|보" delims="|" varStatus="i" >
    <c:if test="${i.first}">color : </c:if>
    ${color} <a href="<c:url value='/TEST.jsp' />">
        View Test
    </a>
    <c:if test="${!i.last}">,</c:if>
</c:forTokens>
</b>
```

JSTL(JSP Standard Tag Library) - Core

▶ <c:url /> : URL의 생성

```
<c:url var="생성한 URL이 저장될 변수 명" value="생성할 URL" scope="변수 공유 범위" />
```

```
<a href="<c:url value='/TEST.jsp' />">
    View Test
</a>
```

▶ <c:param /> : 파라미터 추가

```
<c:param name="파라미터 명" value="값" />
```

```
<c:url value="TEST.jsp" var="paramTest">
    <c:param name="data1" value="1000" />
    <c:param name="data2" value="2000" />
</c:url>
<a href="${paramTest}">
    View TEST with data1(1000), data2(2000)
</a>
```

JSTL(JSP Standard Tag Library) - Core

- ▶ `<c:import />` : 페이지 첨부
- ▶ import 태그 내에 param 태그도 사용할 수 있다.

```
<c:import url="첨부할 URL" />
```

```
<c:import url="/TEST1.jsp" />
<c:import url="/TEST2.jsp">
    <c:param name="data1" value="1000" />
    <c:param name="data2" value="2000" />
</c:import>
```

JSTL(JSP Standard Tag Library) - Core

- ▶ `<c:redirect />` : `sendRedirect()`와 동일

```
<c:redirect url="이동할 URL" />
```

```
<c:redirect url="http://www.daum.net" />
```

JSTL(JSP Standard Tag Library) – XML

▶ JSTL XML

- ▶ XML을 처리하기 위한 것으로 XML 출력, 흐름제어, XML 변환 등의 작업에 사용된다.

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

Xml 태그	설명
x:out	XPath에 지정한 패턴에 따라 xml 내용을 출력한다
x:parse	Xml 문서를 읽어 파싱한다
x:set	XPath에 따라 선택된 내용을 변수에 저장한다.
x:choose	<c:choose> 태그와 기능이 같다.
x:when	조건식이 true 일 때 수행한다
x:otherwise	조건이 false 일 때 수행한다.
x:if	조건문에 사용하는 태그
x:transform	Xml과 xslt 파일을 결합해서 새로운 형식의 문서를 생성한다.
x:param	파라미터 사용 시 사용

JSTL(JSP Standard Tag Library) – XML

▶ xalan jar 설치

- ▶ 기존 XPath를 사용하기 위해 라이브러리를 따로 다운로드 받아 설치해야 한다.
- ▶ <http://mirror.apache-kr.org/xalan/xalan-j/binaries/>

	Parent Directory	-
	xalan-j_2.7.2-bin-2jars.tar.gz	2014-10-31 02:57 12M
	xalan-j_2.7.2-bin-2jars.zip	2014-10-31 02:57 16M
	xalan-j_2.7.2-bin.tar.gz	2014-10-31 02:57 13M
	xalan-j_2.7.2-bin.zip	2014-10-31 02:57 17M

JSTL(JSP Standard Tag Library) – XML

- ▶ <x:out> : xml 내용 출력

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

```
<vegetables>
  <vegetable>
    <name>onion</name>
    <price>40/kg</price>
  </vegetable>
  <vegetable>
    <name>Potato</name>
    <price>30/kg</price>
  </vegetable>
  <vegetable>
    <name>Tomato</name>
    <price>90/kg</price>
  </vegetable>
</vegetables>
```

```
<x:parse xml="${vegetable}" var="output"/>
<b>Name of the vegetable is</b>:
<x:out select="$output/vegetables/vegetable[1]/name" /><br>
<b>Price of the Potato is</b>:
<x:out select="$output/vegetables/vegetable[2]/price" />
```


JSTL(JSP Standard Tag Library) – XML

- ▶ `<x:parse>` : xml 문서를 읽어 파싱한다

```
<x:parse xml="XML문서" {var= " 변수 명 " [scope= " scope명 " ] | varDom= " 변수  
명 " [scopeDom= " scope명 " ]} [systemId="systemId"] [filter="filter"]/>
```

```
<h2>Books Info:</h2>  
<c:import var="bookInfo" url="novels.xml"/>  
  
<x:parse xml="${bookInfo}" var="output"/>  
<p>First Book title: <x:out select="$output/books/book[1]/name" /></p>  
<p>First Book price: <x:out select="$output/books/book[1]/price" /></p>  
<p>Second Book title: <x:out select="$output/books/book[2]/name" /></p>  
<p>Second Book price: <x:out select="$output/books/book[2]/price" /></p>
```

JSTL(JSP Standard Tag Library) – XML

- ▶ `<x:set>` : XPath에 따라 선택된 내용을 변수에 저장한다.

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

```
<x:parse xml="${book}" var="output"/>  
<x:set var="fragment" select="$output/books/book[2]/price"/>  
<b>The price of the Tomorrow land book</b>:  
<x:out select="$fragment" />
```

JSTL(JSP Standard Tag Library) – XML

- ▶ `<x:set>` : XPath에 따라 선택된 내용을 변수에 저장한다.

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

```
<x:parse xml="${book}" var="output"/>
<x:set var="fragment" select="$output/books/book[2]/price"/>
<b>The price of the Tomorrow land book</b>:
<x:out select="$fragment" />
```

JSTL(JSP Standard Tag Library) – XML

- ▶ `<x:choose>`, `<x:when>`, `<x:otherwise>` : `<c:choose>`와 동일하다

```
<x:choose>
  <x:when select="조건 판별식"> .... </c:when>
  <x:when select ="조건 판별식"> .... </c:when>
  <x:when select ="조건 판별식"> .... </c:when>
  <x:otherwise> ... </c:otherwise>
</x:choose>
```

```
<x:parse xml="${xmltext}" var="output"/>
<x:choose>
  <x:when select="$output//book[1]/author = 'Chetan Bhagat'">
    Book is written by Chetan bhagat
  </x:when>
  <x:when select="$output//book[1]/author = 'Brad Bird'">
    Book is written by Brad Bird
  </x:when>
  <x:otherwise>
    The author is unknown...
  </x:otherwise>
</x:choose>
```

```
<c:set var="xmltext">
<books>
<book>
  <name>Three mistakes of my life</name>
  <author>Chetan Bhagat</author>
  <price>200</price>
</book>
<book>
  <name>Tomorrow land</name>
  <author>Brad Bird</author>
  <price>2000</price>
</book>
</books>
</c:set>
```

JSTL(JSP Standard Tag Library) – XML

- ▶ <x:if> : <x:choose>와 동일하다

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

```
<x:parse xml="${vegetables}" var="output"/>
<x:if select="$output/vegetables/vegetable/price < 100">
    Vegetables prices are very low.
</x:if>
```

- ▶ <x:param> : 파라미터 사용 시에 사용된다.

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

JSTL(JSP Standard Tag Library) – i18n

▶ JSTL I18N(International, Formatting)

▶ 국제화 지역화 태그

▶ 다국어 문서 처리 시 유용하며 날짜와 숫자 형식을 다룰 때 유용하다.

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

Tag 이름	설명
requestEncoding	Request.setCharacterEncoding()과 같은 역할을 한다
setLocale	다국어를 지원하는 페이지를 만들 경우에 사용한다.
timeZone	타임 존(Time Zone)을 적용할 때 사용된다.
setTimeZone	특정 scope의 타임 존을 설정할 때 사용된다.
bundle	Properties 확장자를 사용하는 자원 파일을 읽어오는 역할을 한다.
setBundle	페이지 전체에서 사용할 수 있는 번들(bundle)을 지정하는데 사용된다.
message	번들 태그에서 정한 값을 가져온다.

JSTL(JSP Standard Tag Library) – i18n

▶ JSTL I18N(International, Formatting)

Tag 이름	설명
Param	<fmt:message> 태그의 서브 태그로 설정하지 않은 값을 채워준다
formatNumber	숫자 형식을 표현할 때 사용된다.
parseNumber	문자열로부터 수치를 파싱해낸다. 즉, 문자열을 숫자로 변환할 때 사용된다.
formatDate	날짜 형식을 표현할 때 사용된다.
parseDate	문자열에서 날짜를 파싱해낸다. 즉, 문자열을 날짜로 변환할 때 사용한다.

JSTL(JSP Standard Tag Library) – 18n

- ▶ `<fmt:parseNumber>` : 숫자 형식을 표현할 때 사용한다.

```
<fmt:parseNumber value="Number로 파싱할 수 있는 수치 데이터"
    type="숫자, 통화, 퍼센트 중 하나{number | currency | percent}"
    pattern="사용자 지정 패턴"
    parseLocale="로케일 지정"
    integerOnly="integer로만 파싱할지 지정"
    var="변수"
    scope="범위" />
```

```
<h3>The fmt:parseNumber tag Example is:</h3>
<c:set var="Amount" value="786.970" />
<fmt:parseNumber var="j" type="number" value="${Amount}" />
<p><i>Amount is:</i> <c:out value="${j}" /></p>
<fmt:parseNumber var="j" integerOnly="true" type="number" value="${Amount}" />
<p><i>Amount is:</i> <c:out value="${j}" /></p>
```

JSTL(JSP Standard Tag Library) – 18n

▶ <fmt:timeZone> : 타임존 지정

```
<fmt:timeZone value="">
```

```
<c:set var="str" value="<%=new java.util.Date()%>" />
<table border="2" width="100%">
  <tr>
    <td width="100%" colspan="2" bgcolor="#FF7F50">
      <p align="center">
        <b>
          <font color="#000000" size="6">Formatting:
          <fmt:formatDate value="${str}" type="both"
            timeStyle="long" dateStyle="long" />
        </font>
      </b>
    </p>
  </td>
</tr>
```

```
<c:forEach var="zone"
  items="<%=java.util.TimeZone.getAvailableIDs()%>">
  <tr>
    <td width="50%" bgcolor="#C0C0C0">
      <c:out value="${zone}" />
    </td>
    <td width="50%" bgcolor="#FFEB3D">
      <fmt:timeZone value="${zone}">
        <fmt:formatDate value="${str}" timeZone="${zn}"
          type="both"/>
      </fmt:timeZone>
    </td>
  </tr>
</c:forEach>
```

JSTL(JSP Standard Tag Library) – 18n

- ▶ `<fmt:formatNumber>` : 수치 데이터를 숫자, 통화, 퍼센트로 변환

```
<fmt:formatNumber value="Number로 형식화할 수치 데이터"
  type="숫자, 통화, 퍼센트 중 하나{number | currency | percent}"
  pattern="사용자 지정 패턴"
  currencyCode="통화코드지정"
  currencySymbol="통화기호"
  groupingUsed="{true | false}출력시 그룹 분리 기호(.) 포함여부"
  maxIntegerDigits="출력시 integer 최대 자릿수 지정"
  minIntegerDigits="출력시 integer 최소 자릿수 지정"
  maxFractionDigits="출력시 소수점 이하 최대 자릿수 지정"
  minFractionDigits="출력시 소수점 이하 최소 자릿수 지정"
  var="변수"
  scope="범위" />
```

JSTL(JSP Standard Tag Library) – i18n

- ▶ `<fmt:formatNumber>` : 수치 데이터를 숫자, 통화, 퍼센트로 변환

```
<h3>Formatting of Number:</h3>
<c:set var="Amount" value="9850.14115" />
<p> Formatted Number-1:
<fmt:formatNumber value="${Amount}" type="currency" /></p>
<p>Formatted Number-2:
<fmt:formatNumber type="number" groupingUsed="true" value="${Amount}" /></p>
<p>Formatted Number-3:
<fmt:formatNumber type="number" maxIntegerDigits="3" value="${Amount}" /></p>
<p>Formatted Number-4:
<fmt:formatNumber type="number" maxFractionDigits="6" value="${Amount}" /></p>
<p>Formatted Number-5:
<fmt:formatNumber type="percent" maxIntegerDigits="4" value="${Amount}" /></p>
<p>Formatted Number-6:
<fmt:formatNumber type="number" pattern="###.###$" value="${Amount}" /></p>
```

JSTL(JSP Standard Tag Library) – 18n

- ▶ <fmt:parseDate> : 문자열에서 날짜로 파싱

```
<fmt:parseDate value="파싱할 날짜 데이터"  
    type="{time | date | both}"  
    dateStyle="{short | full}"  
    timeStyle="{short | full}"  
    pattern="사용자 지정 패턴"  
    timeZone="타임존 지정"  
    parseLocale="로케일 지정"  
    var="변수"  
    scope="범위" />
```

```
<h3>Parsed Date:</h3>
```

```
<c:set var="date" value="12-08-2016" />
```

```
<fmt:parseDate value="${date}" var="parsedDate" pattern="dd-MM-yyyy" />
```

```
<p><c:out value="${parsedDate}" /></p>
```

JSTL(JSP Standard Tag Library) – 18n

▶ <fmt:formatDate> : 문자열에서 날짜로 파싱

```
<fmt:formatDate value="형식화할 날짜와 시간 데이터"
                type="{time | date | both}"
                dateStyle="{short | full}"
                timeStyle="{short | full}"
                pattern="사용자 지정 패턴"
                timeZone="타임존 지정"
                var="변수"
                scope="범위" />
```

```
<h2>Different Formats of the Date</h2>
<c:set var="Date" value="<%=new java.util.Date()%>" />
<p>Formatted Time : <fmt:formatDate type="time" value="${Date}" /></p>
<p>Formatted Date : <fmt:formatDate type="date" value="${Date}" /></p>
<p>Formatted Date and Time : <fmt:formatDate type="both" value="${Date}" /></p>
<p>Formatted Date and Time in short style : <fmt:formatDate type="both"
dateStyle="short" timeStyle="short" value="${Date}" /></p>
<p>Formatted Date and Time in medium style : <fmt:formatDate type="both"
dateStyle="medium" timeStyle="medium" value="${Date}" /></p>
<p>Formatted Date and Time in long style : <fmt:formatDate type="both"
dateStyle="long" timeStyle="long" value="${Date}" /></p>
```


JSTL(JSP Standard Tag Library) – SQL

▶ JSTL SQL 라이브러리

- ▶ DataSource를 이용한 SQL 작업 처리에 사용된다.

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

태그	설명
transaction	트랜잭션을 구현할 때 사용한다.
query	Sql 태그의 속성 또는 body에 정의된 SQL 쿼리 문장을 실행한다. executeQuery() 와 같은 기능을 한다
update	Sql 태그의 속성 또는 body에 정의된 SQL 쿼리 문장을 실행한다. executeUpdate() 와 같은 기능을 한다
param	Java.sql.PreparedStatement.setString()의 역할을 한다.
dateParam	Java.sql.PreparedStatement.setTimestamp() 역할을 한다.
setDataSource	DataSource를 지정한다

JSTL(JSP Standard Tag Library) – function

▶ JSTL function

- ▶ JSTL functions 라이브러리는 JSTL에서 제공하는 각종 함수를 사용해서 문자열이나 컬렉션들을 처리한다.
- ▶ 이 라이브러리는 함수 실행 결과값을 반환하기 때문에 태그 자체로 쓰이기보다는 특정 태그 내의 값으로 많이 사용된다.

함수	설명
length(obj)	obj가 Collection인 경우 저장된 항목의 개수를, 문자인 경우 문자열의 길이를 반환
toUpperCase(str)	str을 대문자로 변환
toLowerCase(str)	str을 소문자로 변환
substring(str, idx1, idx2)	str.substring(idx1, idx2)의 결과를 반환, idx2가 -1 일 경우 str.substring(idx1)과 동일
substringAfter(str1, str2)	str1에서 str1에 포함되어 있는 str2 이후의 문자열을 구함
substringBefore(str1, str2)	str1에서 str1에 포함되어 있는 str2 이전의 문자열을 구함 trim(str)
trim(str)	str 좌우의 공백 문자를 제거

JSTL(JSP Standard Tag Library) – function

▶ JSTL function

함수	설명
trim(str)	str 좌우의 공백 문자를 제거
replace(str, src, dest)	str에 있는 src를 dest로 변환
indexOf(str1, str2)	str1에서 str2가 위치한 인덱스를 구함
startsWith(str1, str2)	str1이 str2로 시작할 경우 true, 그렇지 않을 경우 false를 반환
endsWith(str1, str2)	str1이 str2로 끝나는 경우 true, 그렇지 않을 경우 false를 반환
contains(str1, str2)	str1이 str2를 포함하고 있을 경우 true를 반환
containsIgnoreCase(str1, str2)	대소문자 구분없이 str1이 str2를 포함하고 있을 경우 true를 반환
split(str1, str2)	str2로 명시한 글자를 기준으로 str1을 분리해서 배열로 반환
join(array, str2)	array에 저장된 문자열을 합침, 각 문자열의 사이에는 str2가 붙음
escapeXml(str)	XML의 객체 참조에 해당하는 특수문자를 처리함

JSTL(JSP Standard Tag Library) – function

▶ JSTL function

```
<%  
    HashMap<String,String> map = new HashMap<String,String>();  
    map.put("1", "1");  
    map.put("2", "2");  
    map.put("3", "3");  
    map.put("4", "4");  
    map.put("5", "5");  
%>  
<c:set var="map" value="<%= map %>" />  
<c:set var="str1" value="김근형(graykim) 깃헙, https://github.com/graykim" />  
<c:set var="str2" value="깃헙" />  
<c:set var="tokens" value="1,2,3,4,5,6,7,8,9,10" />
```

JSTL(JSP Standard Tag Library) – function

▶ JSTL function

`<h1>Function 태그</h1>`

`length(map) : ${ fn:length(map) }
`

`length(str1) : ${ fn:length(str1) }
`

`toUpperCase(str1) : ${ fn:toUpperCase(str1) }
`

`toLowerCase(str1) : ${ fn:toLowerCase(str1) }
`

`substring(str1, 19, 40) : ${ fn:substring(str1, 19, 40) }
`

`substringAfter(str1, str2) : ${ fn:substringAfter(str1, str2) }
`

`substringBefore(str1, str2) : ${ fn:substringBefore(str1, str2) }
`

`replace(str1, src, dest) : ${ fn:replace(str1, " ", "^") }
`

`indexOf(str1, str2) : ${ fn:indexOf(str1, str2) }
`

`startsWith(str1, str2) : ${ fn:startsWith(str1, '깃헙') }
`

`endsWith(str1, str2) : ${ fn:endsWith(str1, '/') }
`

`contains(str1, str2) : ${ fn:contains(str1, str2) }
`

`containsIgnoreCase(str1, str2) : ${ fn:containsIgnoreCase(str1, str2) }
`

`<c:set var="array" value="${ fn:split(tokens, ',') }" />`

`join(array, "-") : ${ fn:join(array, "-") }
`

`escapeXml(str1) : ${ fn:escapeXml(str1) }`