

Servlet & JSP

EL & JSTL

EL & JSTL

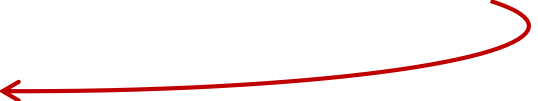
EL(Expression Language)

JSP 2.0 버전에서 추가된 것으로 `<%= %>`, `out.print()` 와 같이 JSP에 쓰이는 Java 코드를 간결하게 사용하는 방법으로, 화면에 표현하고자 하는 코드를 `${ value }` 의 형식으로 표현하여 작성하는 것을 말한다.

사용 문법과 예시

문법 : `${ value }`

예시 :

`<%= request.getParameter("name") %>`
`${ param.name }` 

EL 연산자 기호

() : ~의 원말

	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

EL & JSTL

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 & JSTL

EL 내장 객체

객체 명	설명
pageScope	page 영역의 객체에 접근
requestScope	request 영역의 객체에 접근
sessionScope	session 영역의 객체에 접근
applicationScope	application 영역의 객체에 접근
param	전달된 파라미터값을 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	<code>\${cookie.key명}</code> 으로 쿠키값을 조회한다
initParam	초기 파라미터를 조회한다
pageContext	pageContext 경로를 조회한다

EL & JSTL

JSTL(JSP Standard Tag Library)

JSP에서 사용하는 커스텀 태그로, 공통으로 사용하는 코드의 집합을 사용하기 쉽게 태그화하여 표준으로 제공한 것을 말한다.

선언 방식과 사용 예

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" %>
```

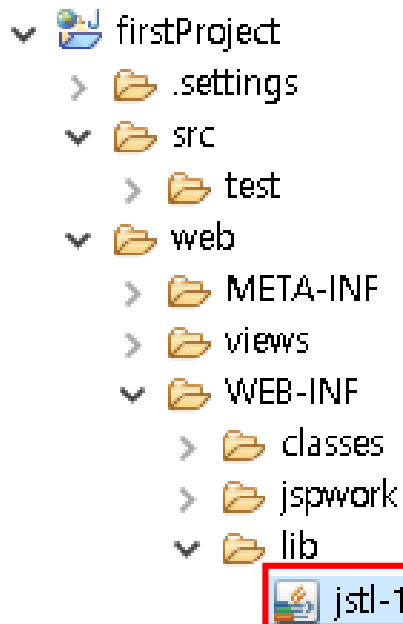
```
<c:out value="${'Welcome to javaTpoint'}"/>
```

EL & JSTL

JSTL 라이브러리 등록

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

상위 링크를 통해 jstl-1.2.jar 파일 설치 후 이클립스 프로젝트 내
web/WEB-INF/lib 내에 등록하고 사용하고자 하는 jsp 파일에서 선언한다



```
<%@ 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>
```

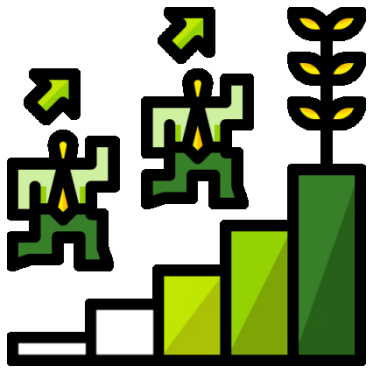
EL & JSTL

JSTL 태그 종류

태그 명	설명
Core Tags	<ul style="list-style-type: none">- 변수와 url, 조건문, 반복문 등의 로직과 관련된 JSTL 문법을 제공- 선언 : <code><%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %></code>
Formatting Tags	<ul style="list-style-type: none">- 메시지 형식이나 숫자, 날짜 형식과 관련된 포맷 방식을 제공- 선언 : <code><%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %></code>
Function	<ul style="list-style-type: none">- trim, substring 과 같은 여러 문자열 처리 함수를 제공- 선언 : <code><%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %></code>
XML Tags	<ul style="list-style-type: none">- 데이터의 XML 파싱 처리 등 XML 문서를 화면으로 읽어오는 데 필요한 라이브러리- 선언 : <code><%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %></code>
SQL Tags	<ul style="list-style-type: none">- 페이지 내에서 Database를 연동하고, 필요한 쿼리를 실행할 수 있는 라이브러리- 선언 : <code><%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %></code>

※ 최근에는 XML보다 JSON으로 인한 데이터 처리가 늘고 있어 XML Tags 사용을 지양한다.

※ SQL Tags는 Database를 직접 연동하는 것에는 용이하나, Database 트랜잭션을 별도로 묶어 관리하는 MVC Model 2 방식에는 맞지 않기 때문에 사용을 지양한다.



Servlet&JSP

JSTL Core Tags

JSTL Core Tags

〈c:set〉 태그

〈c:set〉은 변수를 선언하고 나서 그 변수에 초기값을 대입하는 기능의 태그로, 자바에서 변수를 선언하는 방법과 비슷하다.

Java 변수 선언 방식

int num = 100;

변수 타입 변수 명 초기값



〈c:set〉 변수 선언 방식

〈c:set var="num" value="100" /〉

변수 명 초기값



JSTL Core Tags

〈c:set〉 사용법

- 〈c:set〉에서의 변수 타입은 별도로 선언하지 않는다.
- 초기값은 반드시 기술해야 한다.
- 〈c:set〉으로 선언한 변수는 EL 식 안에서 사용할 수 있다. 하지만 JSP 〈% %〉 같은 스크립트릿 요소에서는 사용할 수 없다.

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

...

`${num}`



〈c:set〉에서 선언한 변수는
EL식 안에서 사용 가능함

JSTL Core Tags

〈c:set〉 사용법

- 〈c:set〉에서 선언한 변수를 JSP 스크립팅 요소에서 쓰는 것은 불가능하지만, 반대로 스크립팅 요소 안에서 선언한 변수를 〈c:set〉의 value 속성에 사용하는 것은 가능하다.

```
〈% int num1=10, num2=20; %〉
```

```
〈c:set var="sum" value="〈%= num1+num2 %〉" /〉
```

```
${sum}
```

JSP 스크립트 요소의 변수를
value 속성에서 사용할 수 있음

JSTL Core Tags

〈c:set〉 실습

```
<%@ page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSTL Core 실습</title>
</head>
<body>
<h2>JSTL Core Tags 실습</h2>
<hr>
<c:set var="num1" value="10"/>
<c:set var="num2" value="20"/>
<ol>
<li>&lt;c:set> 테스트 : ${num1} + ${num2} = ${num1 + num2}</li>
</ol>
</body>
</html>
```

JSTL Core Tags

〈c:set〉 태그 scope 속성

- 〈c:set〉의 scope 속성을 이용하면 page 영역 뿐 아니라 request, session, application 영역에 속성을 저장하는 것이 가능하다.
- 설정하지 않을 시 기본값은 page 이다.

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

JSTL Core Tags

〈c:set〉 태그 배열 설정

- 〈c:set〉의 body 부분에 "." 를 이용해서 배열이나 Collection 과 같이 여러 개의 값을 지정할 수 있다.

〈c:set var="array" scope="request" 〉

yellow, blue, pink, red, green

〈/c:set〉

〈c:set〉 태그의 body에 있는 값이
array 변수에 할당된 변수 값이 된다

JSTL Core Tags

〈c:remove〉 태그 배열 설정

- 〈c:set〉을 이용해서 선언한 변수는 page, request, session, application 영역에 속성으로 저장되기 때문에 삭제해야 할 필요가 있다.
- 이 때 사용하는 태그가 〈c:remove〉 태그이다.

〈c:remove var="num1" scope="request"〉

↑
변수 명

↑
request 영역에 있는 변수 제거

※ scope 속성을 정의하지 않으면 page, request, session, application 영역에 저장되어있는 num 이라는 이름의 속성을 모두 찾아서 제거한다.

JSTL Core Tags

〈c:out〉 태그

- 〈c:out〉 태그는 데이터를 출력할 때 사용하는 태그이다.
- 〈, 〉, & 특수 문자를 자동으로 이스케이프 시퀀스(escape sequence)로 바꾼다.

특수 문자	이스케이프 시퀀스(escape sequence)
<	<
>	>
&	&

사용 예

〈c:out value="〈title〉은 〈head〉의 하위태그이다." />

〈title〉와 〈head〉는 웹 브라우저가 해석하지
않고 기술한 대로 화면에 나타난다

JSTL Core Tags

〈c:out〉 태그

- 출력할 데이터에 포함된 특수 문자를 태그의 일부로 인식시키고 싶을 경우 `escapeXml` 이라는 속성을 추가하고, `false` 값을 지정한다.

사용 예

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

〈h2〉 태그는 웹 브라우저에 의해
html 태그로 인식되어 화면에 나타난다

JSTL Core Tags

<c:out> 실습

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri=" http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSTL Core 실습</title>
</head>
<body>
<h2>JSTL Core Tags 실습</h2>
<hr>
  <li>&lt;c:out&gt; 테스트 :
    <c:out value="<h3>c:out 테스트 입니다.</h3>"/></li>
  <li>&lt;c:out&gt; 테스트 escapeXml:
    <c:out value="<h3>c:out 테스트 입니다.</h3>" escapeXml="false"/></li>
</body>
</html>
```

JSTL Core Tags

〈c:if〉 태그

- 자바 프로그램의 if 문과 비슷한 역할을 하는 태그.
- 〈c:if〉 태그에서 조건식은 test라는 속성의 값으로 지정해야 한다.
이 때 조건식은 반드시 EL 형식으로 기술한다.

사용 예

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

num1 이 더 큼니다. . .

```
〈 /c:if 〉
```

JSTL Core Tags

〈c:choose〉 태그

- 자바 프로그램의 switch 문과 비슷한 역할을 하는 태그.
- 〈c:when〉, 〈c:otherwise〉 태그와 함께 사용되는데,
각각 switch 문의 case, default 절과 비슷한 역할을 한다.

사용 예

〈c:choose〉

〈c:when test="\${num == 0}"〉

처음 뵙겠습니다. 〈br/〉

〈/c:when〉

〈c:when test="\${num == 1}"〉

다시 뵙게 되어 반갑습니다. 〈br/〉

〈/c:when〉

〈c:otherwise〉

안녕하세요. 〈br/〉

〈/c:otherwise〉

〈/c:choose〉

조건식

아무 조건도 만족하지 못할 경우
실행되는 코드

JSTL Core Tags

〈c:forEach〉 태그

- 자바의 for, for-in문에 해당하는 기능을 제공한다.

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 변수의 이름
varStatus	현재 반복에 해당하는 객체의 요소

사용 예

〈c:forEach begin="1" end="10" items="\${list}" var="value"〉

반복문 〈br/〉

〈/c:forEach〉

JSTL Core Tags

<c:forEach> 태그

- varStatus는 다음과 같은 속성을 가지고 있다.

속성	상태 값 명	사용법
current	현재 반복 횟수를 나타낸다	상태값 명.current
index	반복 라운드의 제로 기반(zero-based) 인덱스 (0 ... n)	상태값 명.index
count	반복 라운드의 1 기반(one-based) 인덱스 (1 ... n)	상태값 명.count
first	현재 라운드가 반복을 통한 첫 번째임을 의미	상태값 명.first
last	현재 라운드가 반복을 통한 마지막 번째임을 의미	상태값 명.last

사용 예

```
<c:forEach items="${bookList}" var="book" varStatus="status">
```

```
<tr>
```

```
<td> <c:out value="${status.count}" /></td>
```

```
<td> <c:out value="${book.name}" /></td>
```

```
</tr>
```

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

Collection 객체명

상태값 명

JSTL Core Tags

〈c:forTokens〉 태그

- 문자열에 포함된 구분자를 통해 토큰을 분리해서 반복 처리를 한다.
- items 속성에는 토큰을 포함하는 문자열을 지정하고, delims 속성에는 토큰을 분리하는데 사용할 구획 문자를 기술한다.

사용 예

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


JSTL Core Tags

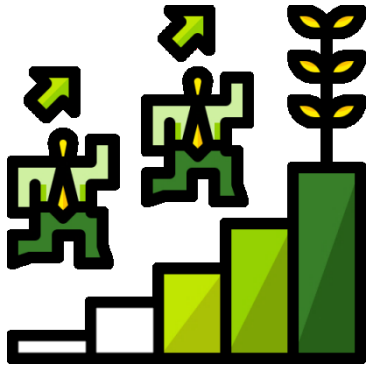
〈c:url〉 태그

- url 경로를 생성하고, 해당 url의 param 속성을 선언하여 쿼리스트링을 정의할 수 있는 태그.
- 해당 태그를 통해 url 경로와 관련 쿼리스트링의 값을 미리 설정하여 이를 제어할 수 있다.

사용 예

```
〈c:url var="url" value="jstl.jsp" 〉  
    〈c:param name="name" value="abc" /〉  
〈/c:url〉
```

```
〈a href="${url}"〉 jstl.jsp 〈/a〉
```



Servlet&JSP

JSTL Formatting Tags

JSTL Formatting Tags

<fmt: ??? >

태그 명	내 용
requestEncoding	value 속성을 통해 지정한 문자 셋으로 변경
setLocale	통화 기호나 시간 대역을 설정한 지역에 맞게 표시
timeZone	특정 영역의 시간대(GMT / GMT-9 등)를 설정
setTimeZone	특정 영역의 시간대 설정 정보를 변수에 저장하는 태그
bundle	basename 속성에 지정된 properties 파일을 읽어오는 태그
setBundle	properties 파일을 읽어와 다양한 영역에서 참조할 수 있게 설정
message	bundle 태그를 통해 저장된 key로 value를 가져오는 태그
formatNumber	숫자를 특정 양식에 맞추어 출력하는 태그
parseNumber	문자열을 숫자 형식으로 변환하는 태그
formatDate	날짜 정보를 가진 객체(Date)를 특정 형식으로 변환하여 출력하는 태그
parseDate	문자열을 날짜 형식으로 변환하여 출력하는 태그

JSTL Formatting Tags

〈fmt:requestEncoding〉

- 요청 파라미터의 문자셋을 value속성으로 설정하는 태그.

사용 예

〈fmt:requestEncoding value="변경하고자 하는 문자셋"/〉

〈% request.setCharacterEncoding("변경하고자 하는 문자셋"); %〉 과 동일하다.

〈fmt:requestEncoding value="UTF-8"/〉

JSTL Formatting Tags

〈fmt:setLocale〉

- 지역 설정을 통해 통화 기호나 시간 대역을 다르게 설정할 수 있다.
- 국가-지역 설정은 다음 주소를 참고하자.

<http://www.lingoes.net/en/translator/langcode.html>

JSTL Formatting Tags

〈fmt:setLocale〉 사용 예

```
〈c:set var="Date" value="〈%=new java.util.Date()%〉" /〉
```

```
〈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 Formatting Tags

〈fmt:timeZone〉

- 특정 지역을 설정하여 통화 기호나 시간 대역을 다르게 표현한다.
- 각 지역의 timeZone 정보는 아래의 링크를 참고하자.

https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

JSTL Formatting Tags

〈fmt:timeZone〉 사용 예

```
〈c:set var="now" value="〈%= new java.util.Date() %〉" /〉
```

```
〈fmt:timeZone value="America/New_York"〉
```

```
뉴욕 : 〈fmt:formatDate value="${ now }" type="both" dateStyle="full" timeStyle="full" /〉
```

```
〈/fmt:timeZone〉
```

```
〈br〉
```

```
〈fmt:timeZone value="Europe/London"〉
```

```
런던 : 〈fmt:formatDate value="${ now }" type="both" dateStyle="full" timeStyle="full" /〉
```

```
〈/fmt:timeZone〉
```


JSTL Formatting Tags

〈fmt:setTimeZone〉

- fmt:timeZone과 동일하나, 시작과 끝 태그에 영향을 미치는 것이 아닌, 선언한 이후의 모든 태그에 영향을 미친다.

사용 예

```
〈c:set var="now" value="〈%= new java.util.Date() %〉" /〉
```

```
〈fmt:setTimeZone value="America/New_York" /〉
```

```
뉴욕 : 〈fmt:formatDate value="${ now }" type="both" /〉
```

```
〈br〉
```

```
〈fmt:setTimeZone value="Europe/London" /〉
```

```
런던 : 〈fmt:formatDate value="${ now }" type="both" /〉
```

JSTL Formatting Tags

<fmt:formatNumber>

- 표현하고자 하는 숫자의 포맷을 통화 기호, ' , ' 표시, %표시 등 원하는 쓰임에 맞게 지정할 수 있다.

사용 예

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

<p> 포맷 방식 세자리 구분 : <fmt:formatNumber value="\${number}" type="number" groupingUsed="true" /></p> <!-- 12,300.12 -->

<p> 포맷 방식 통화 기호 : <fmt:formatNumber value="\${number}" type="currency" /></p> <!-- ₩12,300 -->

<p> 포맷 방식 백분율 : <fmt:formatNumber value="\${number}" type="percent" /></p> <!-- 012% -->

JSTL Formatting Tags

<fmt:formatNumber> 속성 정리

속성 명	설 명	필수 여부
value	원하는 표현 방식으로 사용하기 위한 숫자 데이터	Yes
type	표시할 타입 지정(NUMBER, CURRENCY, PERCENT)	No
pattern	화면에 표현할 데이터 스타일을 지정하며, 패턴은 java.text.DecimalFormat 클래스의 포맷 방식을 따름	No
currencyCode	type 속성이 "currency"일 경우, 인식할 화폐 단위	No
currencySymbol	type 속성이 "currency"일 경우, 표시할 화폐 기호	No
groupingUsed	'.' 와 같은 각 숫자 단위의 구분자 표시 여부	No
maxIntegerDigits	화면에 표시할 숫자의 최대 자릿수	No
minIntegerDigits	화면에 표시할 숫자의 최소 자릿수	No
maxFractionDigits	화면에 표시할 소수점 이하 숫자의 최대 개수	No
minFractionDigits	화면에 표시할 소수점 이하 숫자의 최소 개수	No
var	변환된 숫자 데이터를 담을 변수 생성	No

JSTL Formatting Tags

〈fmt:formatNumber〉

- maxIntegerDigits 및 minIntegerDigits의 속성으로 표시하고자 하는 수의 단위를 표현할 수 있다. 숫자가 지정한 최대값을 초과할 경우 해당 자릿수 만큼만 표시된다.

사용 예

```
〈fmt:formatNumber type = "number" maxIntegerDigits = "4"  
value = "${number}" /> <!-- 12300.12 -->
```

숫자의 범위가 지정한 부분을 넘어
앞 자리 '1'은 표시되지 않는다.

JSTL Formatting Tags

〈fmt:formatNumber〉

- minFractionalDigits 및 maxFractionalDigits의 속성은 소수 자릿수를 지정할 수 있으며, 숫자가 최소 자릿수를 초과할 시 자동 반올림이 된다.
- 패턴 속성을 사용하여 숫자 포맷 방법을 지정할 수 있다.

사용 예

```
〈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:parseNumber〉

- 특정 문자열을 값으로 받아 원하는 방식의 숫자 표현으로 변환 시킨다.

사용 예

```
〈c:set var = "balance" value = "123000.123" /〉
```

```
〈fmt:parseNumber var = "i" type = "number" value = "${balance}" /〉
```

```
〈p〉숫자 변환 테스트 1 : ${i}〈/p〉 <!-- 123000.123 -->
```

```
〈fmt:parseNumber var = "i" integerOnly = "true" type = "number" value = "${balance}" /〉
```

```
〈p〉숫자 변환 테스트 2 : ${i}〈/p〉 <!-- 123000 -->
```

JSTL Formatting Tags

〈fmt:parseNumber〉 속성 정리

속성 명	설 명	필수 여부
value	숫자 변환에 사용할 데이터 선언	No
type	표시할 타입 지정(NUMBER, CURRENCY, PERCENT)	No
parseLocale	변환할 숫자에 따른 지역 선언	No
integerOnly	소수점 자리를 무시한 정수 영역만 변환하여 표시	No
pattern	변환한 숫자를 표현할 특정 스타일 지정	No
var	변환한 숫자 데이터를 담을 변수 선언	No
scope	var에 담은 변수를 저장할 영역 설정(기본은 page)	No

JSTL Formatting Tags

〈fmt:formatDate〉

- 날짜나 시간의 포맷방식을 지정하여 화면에 보여줄 때 사용한다.
- value 속성으로는 java.util.Date() 객체를 사용해야 한다.
- type 지정 방식에 따라 날짜, 시간, 둘 모두를 표시할 수 있고, dateStyle, timeStyle 속성으로 보여줄 포맷의 표기 방식을 설정할 수 있다.

사용 문법

```
〈c:set var="Date" value="〈%=new java.util.Date()%〉" /〉
```

날짜 포맷 태그 사용 :

```
〈fmt:formatDate type="time" value="${Date}" /〉
```


JSTL Formatting Tags

〈fmt:formatDate〉 속성 정리

속성 명	설 명	필수 여부
value	원하는 날짜, 시간으로 표시하기 위한 데이터	Yes
type	표시할 타입 지정(TIME, DATE, BOTH)	No
pattern	화면에 표현할 데이터 스타일을 지정하며, 패턴은 java.text.DateFormat 클래스의 방식을 따름	No
dateStyle	날짜에 대해 표현할 스타일을 정의 Default, short, medium, long, full	No
timeStyle	시간에 대해 표현할 스타일을 정의 Default, short, medium, long, full	No
timeZone	지역에 따른 시간을 설정할 때 사용하며 이때 사용되는 값은 〈fmt:setTimeZone〉의 timeZone 객체.	No
var	변환된 날짜 데이터를 담을 변수 생성	No
scope	var에 담은 변수를 저장할 영역 설정(기본은 page)	No

JSTL Formatting Tags

Date Style 별 표현 방식

속성 명	설 명
DEFAULT	2018. 1. 31
SHORT	18. 1. 31
MEDIUM	2018. 1. 31
LONG	2018년 1월 31일 (수)
FULL	2018년 1월 31일 수요일

Time Style 별 표현 방식

속성 명	설 명
DEFAULT	오전 1:10:00
SHORT	1:10
MEDIUM	오전 1:10:00
LONG	오전 1시 10분 00초
FULL	오전 1시 10분 00초 KST

JSTL Formatting Tags

〈fmt:formatDate〉 사용 예

〈fmt:formatDate type="time" value="\${Date}" />
〈br〉 오전 1:01:00

〈fmt:formatDate type="date" value="\${Date}" />
〈br〉 2018. 1. 01

〈fmt:formatDate type="both" value="\${Date}" />
〈br〉 2018. 1. 01 오전 1:01:00

〈fmt:formatDate type="both" dateStyle="short" timeStyle="short" value="\${Date}" />
〈br〉 18. 1. 01 오전 1:01

〈fmt:formatDate type="both" dateStyle="medium" timeStyle="medium" value="\${Date}" />
〈br〉 2018. 1. 01 오전 1:01:00

〈fmt:formatDate type="both" dateStyle="long" timeStyle="long" value="\${Date}" />
〈br〉 2018년 1월 1일 (월) 오전 1시 1분 00초

JSTL Formatting Tags

〈fmt:parseDate〉

- 문자열로 표시된 날짜 및 시간 값을 java.util.Date로 변환하여 화면에 표시할 때 사용한다.

사용 예

```
〈fmt:parseDate value = "2021-01-01 01:00:00"  
  pattern = "yyyy-MM-dd HH:mm:ss" var = "date"/>〉
```

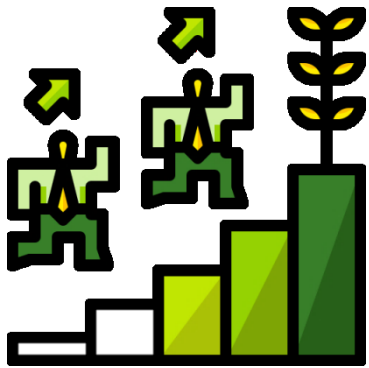
```
〈p〉 ${date} 〈/p〉
```

```
〈!-- 2021-01-01 01:00:00 --〉
```

JSTL Formatting Tags

<fmt:parseDate> 속성 정리

속성 명	설 명	필수 여부
value	원하는 날짜, 시간으로 표시하기 위한 데이터	Yes
type	표시할 타입 지정(TIME, DATE, BOTH)	No
pattern	화면에 표현할 데이터 스타일을 지정하며, 패턴은 java.text.DateFormat 클래스의 방식을 따름	No
dateStyle	날짜에 대해 표현할 스타일을 정의 Default, short, medium, long, full	No
timeStyle	시간에 대해 표현할 스타일을 정의 Default, short, medium, long, full	No
parseLocale	변환할 날짜에 따른 지역 선언	No
timeZone	지역에 따른 시간을 설정할 때 사용하며 이때 사용되는 값은 <fmt:setTimeZone> 의 timeZone 객체.	No
var	변환된 날짜 데이터를 담을 변수 생성	No
scope	var에 담은 변수를 저장할 영역 설정(기본은 page)	No



Servlet&JSP

JSTL Function

JSTL Function

JSTL Function 종류

함수 명	설 명
<code>fn:contains(str, 'text')</code>	<code>str</code> 에 두번째 인자 값의 내용이 포함되어 있는지 확인
<code>fn:containsIgnoreCase(str, 'text')</code>	<code>str</code> 에 대소문자 구분 없이 'text'값이 포함되어 있는지 확인
<code>fn:startsWith(str, 'text')</code>	문자열 <code>str</code> 이 'text'로 시작하는지 확인
<code>fn:endsWith(str, 'text')</code>	문자열 <code>str</code> 이 'text'로 끝나는지 확인
<code>fn:escapeXml(str)</code>	문자열 <code>str</code> 에 xml 태그가 포함되었다면, 해당 태그까지 화면에 출력
<code>fn:indexOf(str, " text")</code>	<code>str</code> 내에 'text'의 첫글자 시작 위치를 반환 (0번 부터 시작한다)
<code>fn:length(str)</code>	문자열 <code>str</code> 의 길이를 반환한다.
<code>fn:replace(str, 'text1', 'text2')</code>	<code>str</code> 내의 <code>text1</code> 을 찾아 <code>text2</code> 로 변경한다.
<code>fn:substring(str, index1, index2)</code>	<code>str</code> 에서 <code>index1</code> 부터 <code>index2</code> 까지의 문자열을 반환한다.
<code>fn:split(str, ' ')</code>	<code>str</code> 을 ' '으로 지정한 구분자를 기준으로 나눠 배열로 만들어 반환
<code>fn:join(str, '-')</code>	배열요소로 나뉘어진 <code>str</code> 을 '-' 구분자를 붙여 합친 뒤 반환한다
<code>fn:trim(str)</code>	<code>str</code> 값의 좌우 공백을 제거한다.

JSTL Function

JSTL Function

문자열 처리에 관한 메소드들을 EL 형식에서 사용할 수 있게 제공하는 라이브러리로, 다른 JSTL 태그들과는 다르게 `${fn:메소드명(문자열)}` 의 형식으로 EL 태그에 직접 대입하여 사용한다.

선언 방식 :

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

예시 :

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

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

```
    <p>Found test string</p>
```

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


학습점검

- ✓ EL&JSTL이 무엇인지 이해할 수 있다.
- ✓ EL내장 객체가 무엇인지 이해하고 사용할 수 있다.
- ✓ JSTL Core태그를 사용할 수 있다.
- ✓ JSTL Formatting 태그를 사용할 수 있다.
- ✓ JSTL Function 태그를 사용할 수 있다.