

# **JSP Taglib**

## JSP Tag Library

# EL

## Expression Language

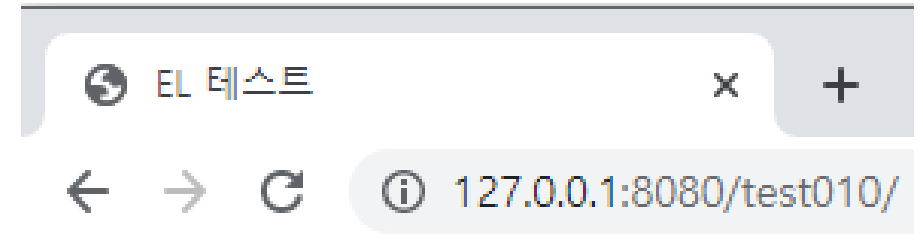
스크립트릿이나  
자바코드를 이용하여  
레이아웃을 구성할 경우  
변수를 입력하는 화면이 복잡해져서  
코드의 가독성이 줄어들 수 있다.

이 경우 EL을 이용하면  
코드를 단순화 하여 생산성을 높일 수 있다.

**`${값}`을 이용하여 사용 가능하다**  
**자바스크립트에서의 ``${var}``와 같은 기능이다.**

# EL 값 확인

```
index.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ page import="java.net.*" %>
4 <!DOCTYPE html>
5 <html>
6   <head>
7     <meta charset="utf-8">
8     <title>EL 테스트</title>
9     <link rel="stylesheet" href="test.css">
10  </head>
11  <body>
12    <!-- EL(Expression Language) -->
13    <!-- jsp를 표현하는 또 다른 방식 -->
14    <%= "TEST" %><br><!-- 표현식을 사용하는 방법 -->
15    <%out.print("TEST"); %><br><!-- 스크립트릿을 사용하는 방법 -->
16    ${"TEST"}<br><!-- EL을 사용하는 방법 -->
17  </body>
18 </html>
```



TEST  
TEST  
TEST

값을 출력하는 방법은 여러가지가 존재한다.

# EL의 종류

```
index.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ page import="java.net.*" %>
4 <!DOCTYPE html>
5 <html>
6   <head>
7     <meta charset="utf-8">
8     <title>EL 테스트</title>
9     <link rel="stylesheet" href="test.css">
10  </head>
11  <body>
12    <!-- EL(Expression Language) -->
13    <!-- jsp를 표현하는 또 다른 방식 -->
14    <%= "TEST" %><br><!-- 표현식을 사용하는 방법 -->
15    <%out.print("TEST"); %><br><!-- 스크립트릿을 사용하는 방법 -->
16    ${"TEST"}<br><!-- EL을 사용하는 방법 -->
17
18    <!-- EL의 종류 -->
19    정수형 : ${10}<br>
20    실수형 : ${5.6}<br>
21    문자열형 : ${"iot융합 프로그래밍"}<br>
22    논리형 : ${true}<br>
23    null : ${null}<br>
24    <!-- null은 값이 표기되지 않는다. -->
25
26    <a href="expr.jsp">계산법 확인하기</a>
27  </body>
28 </html>
```

EL 테스트

127.0.0.1:8080/test010/

TEST  
TEST  
TEST  
정수형 : 10  
실수형 : 5.6  
문자열형 : iot융합 프로그래밍  
논리형 : true  
null :  
[계산법 확인하기](#)

null은 값이 표기되지 않음을 확인

# EL의 산술연산

```
expr.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="utf-8">
7     <title>EL 계산식 확인</title>
8     <link rel="stylesheet" href="test.css">
9   </head>
10  <body>
11    <!-- 앞에 \를 붙이면 부호까지 표기된다. -->
12    <!-- 산술연산 -->
13    \${1+2 } : ${1+2 }<br>
14    \${3-4 } : ${3-4 }<br>
15    \${5*6 } : ${5*6 }<br>
16    \${7/8 } : ${7/8 }<br>
17    \${9%10 } : ${9%10 }<br>
18    \${11 div 12 } : ${11 div 12 }<br><!-- 나눗기와 동일 -->
19    \${13 mod 14 } : ${13 mod 14 }<br><!-- 나머지와 동일 -->
20  </body>
21 </html>
```

EL 계산식 확인

127.0.0.1:8080/test010/expr.jsp

$\{1+2\} : 3$   
 $\{3-4\} : -1$   
 $\{5*6\} : 30$   
 $\{7/8\} : 0.875$   
 $\{9\%10\} : 9$   
 $\{11 \text{ div } 12\} : 0.9166666666666666$   
 $\{13 \text{ mod } 14\} : 13$

null은 값이 표기되지 않음을 확인

# EL의 관계연산

<!-- 관계연산 -->

```
\${1==2 } : ${1==2 }<br>
\${1 eq 2 } : ${1 eq 2 }<br>
\${3!=4 } : ${3!=4 }<br>
\${3 ne 4 } : ${3 ne 4 }<br>
\${5<6 } : ${5<6 }<br>
\${5 lt 6 } : ${5 lt 6 }<br>
\${7>8 } : ${7>8 }<br>
\${7 gt 8 } : ${7 gt 8 }<br>
\${9<=10 } : ${9<=10 }<br>
\${9 le 10 } : ${9 le 10 }<br>
\${11>=12 } : ${11>=12 }<br>
\${11 ge 12 } : ${11 ge 12 }<br>
```

```
${1==2 } : false
${1 eq 2 } : false
${3!=4 } : true
${3 ne 4 } : true
${5<6 } : true
${5 lt 6 } : true
${7>8 } : false
${7 gt 8 } : false
${9<=10 } : true
${9 le 10 } : true
${11>=12 } : false
${11 ge 12 } : false
```

영문자로 된 기호표기가 가능함

# EL의 논리연산

<!-- 논리연산 -->

\\${true && false } : \${true && false }<br>

\\${true || false } : \${true || false }<br>

\\${!true} : \${!true }<br>

<!-- null 확인연산 -->

\\${empty null } : \${empty null }<br>

-\${true && false } : false

-\${true || false } : true

-\${!true} : false

-\${empty null } : true

**영문자로 된 기호표기가 가능함**

# EL의 특징

index.jsp

```
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="utf-8">
7     <title>EL의 패러미터 확장</title>
8   </head>
9   <body>
10    <div id="wrap">
11      <form method="post" action="test.jsp">
12        아이디 : <input type="text" name="user"><br>
13        비밀번호 : <input type="password" name="pw"><br>
14        <hr>
15        좋아하는 계절<br>
16        <input type="checkbox" name="season" value="spring">봄
17        <input type="checkbox" name="season" value="summer">여름
18        <input type="checkbox" name="season" value="fall">가을
19        <input type="checkbox" name="season" value="winter">겨울
20        <input type="submit" value="로그인!">
21      </form>
22
23      <form method="post" action="add.jsp">
24        <input type="text" name="first">+
25        <input type="text" name="second"><br>
26        <input type="submit" value="계산하기!">
27      </form>
28    </div>
29  </body>
30 </html>
```

test.jsp

```
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="utf-8">
7     <title>EL의 패러미터 확인</title>
8   </head>
9   <body>
10    <div id="wrap">
11      <%
12        request.setCharacterEncoding("utf-8");
13        //EL역시도 request 정보를 따라 움직이므로 한글깨짐을 수정하려면
14        //다음의 내용이 표기되어 있어야 한다.
15      %>
16      <h1>${param }</h1>
17      <h1>아이디 : ${param.user }</h1>
18      <h1>비밀번호 : ${param["pw"]} </h1>
19      <!-- 여러개의 값이 오는 경우 처음에 찍힌 값만이 보이는 것을 확인 -->
20      <h1>선택하는 계절 : ${param["season"]} </h1>
21
22      <%
23        String[] arr=request.getParameterValues("season");
24      %>
25      <%
26        for(int i=0; i<arr.length; i++){
27          <%
28            <h1><%=arr[i]%></h1>
29          %>
30        }
31      %>
32      ${paramValues.season} <br>
33      ${paramValues.season[0]}<br>
34      ${paramValues.season[1]} <br>
35
36      기존 방식 : <%=request.getParameter("id")%><br>
37      EL 방식 : ${param.id }<br>
38
39      <!-- 자바식은 리터럴로 인하여 String을 별개의 객체로 본다-->
40      자바식 방식 : <%=request.getParameter("user")==="admin"%><br>
41      <!-- EL은 자바의 연산이 끝난 이후의 값을 기준으로 하므로 같은 리터럴에 해당한다. -->
42      EL 방식 : ${param.user=="admin"}<br>
43    </div>
44  </body>
45 </html>
```

EL을 통하여 request값을 받을 수 있다.



# EL의 계산

```
add.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="utf-8">
7     <title>EL의 패러미터 특징</title>
8   </head>
9   <body>
10    <div id="wrap">
11      <%
12        request.setCharacterEncoding("utf-8");
13        String first=request.getParameter("first");
14        String second=request.getParameter("second");
15
16        String result=first+second;
17      %>
18      <h1>더한 값 : <%=result %></h1>
19
20      <%
21        int fNum=Integer.parseInt(first);
22        int sNum=Integer.parseInt(second);
23        int rNum=fNum+sNum;
24      %>
25      <h1>parseInt : <%=rNum %></h1>
26
27      <h1>EL의 방식 : ${param.first+param.second}</h1>
28      <h1>기존대리의 처리 : ${param.first}${param.second}</h1>
29    </div>
30  </body>
31 </html>
```

12	+	34
<input type="button" value="계산하기!"/>		

더한 값 : 1234

parseInt : 46

EL의 방식 : 46

기존대리의 처리 : 1234

**int와 String의 계산을 임의로 처리함에 주의**

# EL의 scope

```
index.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="utf-8">
7     <title>EL의 스코프범위</title>
8   </head>
9   <body>
10    <%
11      //스코프범위
12      //page : 현재 페이지에서의 스코프
13      //request : 리퀘스트 단위에서의 스코프
14      //session : 세션 단위에서의 스코프
15      //application : 어플리케이션 단위에서의 스코프
16
17      pageContext.setAttribute("name", "page scope");
18      request.setAttribute("name", "request scope");
19      session.setAttribute("name", "session scope");
20      application.setAttribute("name", "application scope");
21    %>
22    <!-- 아무것도 지정하지 않았다면 page scope를 기준으로 데이터를 가져온다. -->
23    <h1>기본형 : ${name }</h1>
24    <!-- 명시적으로 scope의 범위를 지정하여 사용이 가능하다. -->
25    <h1>\${pageScope.name } : ${pageScope.name }</h1>
26    <h1>\${requestScope.name } : ${requestScope.name }</h1>
27    <h1>\${sessionScope.name } : ${sessionScope.name }</h1>
28    <h1>\${applicationScope.name } : ${applicationScope.name }</h1>
29  </body>
30 </html>
```

기본형 : page scope

`${pageScope.name }` : page scope

`${requestScope.name }` : request scope

`${sessionScope.name }` : session scope

`${applicationScope.name }` : application scope

## 개발 시 객체와 변수의 scope단위에 주의

# scope 처리

```
index.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="utf-8">
7     <title>scope</title>
8   </head>
9   <body>
10    <%
11      pageContext.setAttribute("page", "page단위의 데이터");
12      session.setAttribute("ses", "session단위의 데이터");
13      application.setAttribute("app", "application단위의 데이터");
14    %>
15    <h1>\${page } : \${page }</h1>
16    <h1>\${req } : \${req }</h1>
17    <h1>\${ses } : \${ses }</h1>
18    <h1>\${app } : \${app }</h1>
19
20    <a href="index.jsp">페이지 단위의 scope</a><br>
21    <a href="#" onclick="sub()">리퀘스트 단위의 scope</a><br>
22    <a href="session.jsp">세션 단위의 scope</a><br>
23    <a href="application.jsp">어플리케이션 단위의 scope</a><br>
24
25    <form id="req" method="get" action="request.jsp">
26      <input type="hidden" name="req" value="request단위의 데이터">
27    </form>
28    <script>
29      let req=document.getElementById("req");
30      function sub(){
31        req.submit();
32      }
33    </script>
34  </body>
35 </html>
```

**`\${page }` : page단위의 데이터**

**`\${req }` :**

**`\${ses }` : session단위의 데이터**

**`\${app }` : application단위의 데이터**

페이지 단위의 scope

리퀘스트 단위의 scope

세션 단위의 scope

어플리케이션 단위의 scope

**request로 받은 데이터가 없으므로 null**

# scope 처리

```
request.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="utf-8">
7     <title>리퀘스트 단위 테스트</title>
8   </head>
9   <body>
10    <h1>request로 전송되어온 데이터의 존속 확인</h1>
11    <h1>\${page } : ${page }</h1>
12    <h1>\${param.req } : ${param.req }</h1>
13    <h1>\${ses } : ${ses }</h1>
14    <h1>\${app } : ${app }</h1>
15
16    <a href="index.jsp">페이지 단위의 scope</a><br>
17    <a href="request.jsp">리퀘스트 단위의 scope</a><br>
18    <a href="session.jsp">세션 단위의 scope</a><br>
19    <a href="application.jsp">어플리케이션 단위의 scope</a><br>
20  </body>
21 </html>
```

request로 전송되어온 데이터의 존속 확인

`\${page }` :

`\${param.req }` : request단위의 데이터

`\${ses }` : session단위의 데이터

`\${app }` : application단위의 데이터

[페이지 단위의 scope](#)

[리퀘스트 단위의 scope](#)

[세션 단위의 scope](#)

[어플리케이션 단위의 scope](#)

## page단위는 연결되지 않음을 확인

# scope 처리

```
session.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="utf-8">
7     <title>세션 단위 테스트</title>
8   </head>
9   <body>
10    <h1>세션정보 존속 중</h1>
11    <h1>\${page } : ${page }</h1>
12    <h1>\${param.req } : ${param.req }</h1>
13    <h1>\${ses } : ${ses }</h1>
14    <h1>\${app } : ${app }</h1>
15    <%
16      session.invalidate();
17    %>
18    <h1>세션정보 만료 후</h1>
19    <h1>\${ses } : ${ses }</h1>
20
21    <a href="index.jsp">페이지 단위의 scope</a><br>
22    <a href="request.jsp">리퀘스트 단위의 scope</a><br>
23    <a href="session.jsp">세션 단위의 scope</a><br>
24    <a href="application.jsp">어플리케이션 단위의 scope</a><br>
25  </body>
26 </html>
```

세션정보 존속 중

`\${page }` :

`\${param.req }` :

`\${ses }` : session단위의 데이터

`\${app }` : application단위의 데이터

세션정보 만료 후

`\${ses }` :

페이지 단위의 scope

리퀘스트 단위의 scope

세션 단위의 scope

어플리케이션 단위의 scope

## 세션 만료시까지 데이터가 존속함을 확인

# scope 처리

```
application.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="utf-8">
7     <title>어플리케이션 단위 테스트</title>
8   </head>
9   <body>
10    <h1>application 단위에서는 항상 존속 확인</h1>
11    <h1>\${page } : ${page }</h1>
12    <h1>\${param.req } : ${param.req }</h1>
13    <h1>\${ses } : ${ses }</h1>
14    <h1>\${app } : ${app }</h1>
15    <%
16      application.removeAttribute("app");
17    %>
18    <h1>application 단위 데이터 삭제</h1>
19    <h1>\${app } : ${app }</h1>
20
21    <a href="page.jsp">페이지 단위의 scope</a><br>
22    <a href="request.jsp">리퀘스트 단위의 scope</a><br>
23    <a href="session.jsp">세션 단위의 scope</a><br>
24    <a href="application.jsp">어플리케이션 단위의 scope</a><br>
25  </body>
26 </html>
```

application 단위에서는 항상 존속 확인

`\${page }` :

`\${param.req }` :

`\${ses }` :

`\${app }` : application단위의 데이터

application 단위 데이터 삭제

`\${app }` :

[페이지 단위의 scope](#)

[리퀘스트 단위의 scope](#)

[세션 단위의 scope](#)

[어플리케이션 단위의 scope](#)

application은 모든상황, 모든 사용자에게 연동되는 데이터이므로  
사용에 주의하여야 한다.

# JSTL

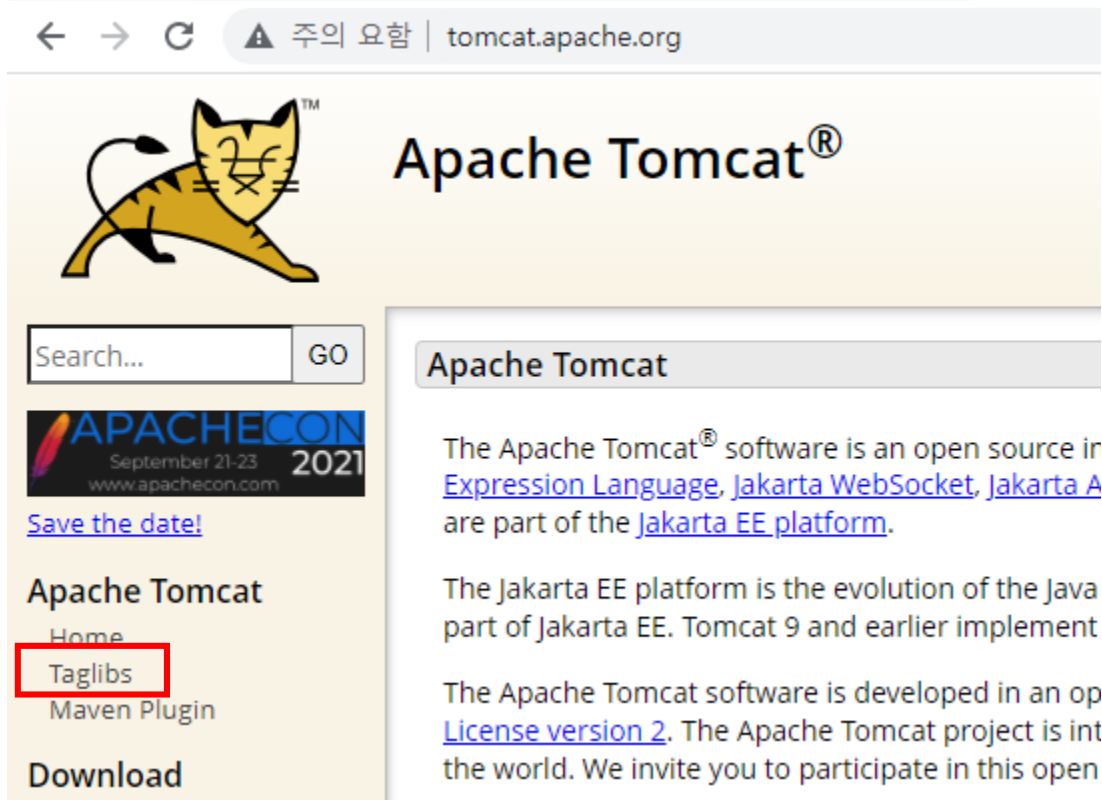
## Jsp Standard Tag Library

jsp에서 지원하는 기본 액션tag  
만으로는  
원하는 기능을 구현하기 쉽지 않다.

JSTL을 이용하면  
코드의 가독성을 높이고  
협업을 편리하게 할 수 있다.

**Tag의 형태로 사용 가능한  
라이브러리이다.**

# jstl 다운로드



## Apache Standard Taglib

The Apache Standard Taglib implements JSTL 1.2 and supports request-time expressions that are evaluated by the JSP container.

In addition, compatibility for applications using 1.0 expression language tags can be enabled in one of two ways:

- Using the **-jstlel** jar supports JSTL 1.0 EL expressions by using the EL implementation originally defined by JSTL itself.
- Using the **-compat** jar supports JSTL 1.0 EL expressions by using the container's implementation of EL to take advantage of newer functionality and potential performance improvements in more modern versions.

[Download](#) | [Changes](#)

Please see the [README](#) file for more detailed information on using the library.

For performance reasons the XML tags use Apache Xalan directly for evaluating XPath expressions. The Xalan 2.7.1 implementation jars xalan.jar and serializer.jar must be added to the classpath.

The Standard Taglib jars may be packaged with a web-application in its /WEB-INF/lib directory, or may be made available to all applications in a container by adding them to the container's classpath.

## 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)

다운로드 받아 필요한 프로젝트에 넣어 준다.



# jstl의 사용

WebContent

META-INF

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

jstl출력

EL출력

scriptlet출력

java출력

java로 저장한 값

jstl로 저장한 값

30

```
index.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!-- DOCTYPE html -->
5 <html>
6   <head>
7     <meta charset="utf-8">
8     <title>jstl의 활용</title>
9   </head>
10  <body>
11    <!-- jsp standard tag library -->
12    <!-- jstl을 사용하기 위해서는 tag라이브러리를 불러와야 한다. -->
13    <!-- taglib은 특정 태그를 사용할 수 있도록 해 주는 기능을 의미한다. -->
14    <!-- uri : uniform resource identifier, 고유식별자 인터넷에 존재하는 각 자료들의 id 고유값 -->
15    <!-- url : uniform resource locator, 해당 자료의 위치를 표기하는 방법으로 사이트의 도메인처럼 위치를 의미 -->
16    <!-- urn : uniform resource name, 해당 자료의 이름을 표기하는 방법 -->
17    <!-- 기본기능 :: c :: http://java.sun.com/jsp/jstl/core -->
18    <!-- 형식변경 :: fmt :: http://java.sun.com/jstl/fmt -->
19    <!-- DB조작 :: sql :: http://java.sun.com/jstl/sql -->
20    <!-- XML조작 :: x :: http://java.sun.com/jstl/xml -->
21    <!-- 함수처리 :: fn :: http://java.sun.com/jsp/jstl/fn -->
22
23    <!-- c:out 값을 출력한다. -->
24    <c:out value="jstl출력"></c:out><br>
25    ${"EL출력"}<br>
26    <%= "scriptlet출력" %><br>
27    <%out.print("java출력"); %><br>
28
29    <!-- c:set 속성에 값을 저장한다 -->
30    <%pageContext.setAttribute("test", "java로 저장한 값"); %>
31    <c:set var="test2" value="jstl로 저장한 값" scope="page"></c:set>
32    ${test}<br>
33    ${test2}<br>
34    <c:set var="number">
35      30
36    </c:set>
37    ${number}
38  </body>
39 </html>
```

개발 시 객체와 변수의 scope단위에 주의

# el과 jstl

```
test.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="utf-8">
8   <title>자바빈과 jstl</title>
9 </head>
10 <body>
11 <%
12   String var="자바단위";
13   pageContext.setAttribute("page", "페이지 단위");
14   request.setAttribute("request", "리퀘스트 단위");
15   session.setAttribute("session", "세션 단위");
16   application.setAttribute("app", "어플리케이션 단위");
17   //스코프 범위로 넘어가지 않으면 출력되지 않음을 확인
18 %>
19 <h1>EL의 출력</h1>
20 \${var } : \${var }<br>
21 \${page } : \${page }<br>
22 \${request } : \${request }<br>
23 \${session } : \${session }<br>
24 \${app } : \${app }<br>
25
26 <c:set var="page" value="페이지 단위" scope="page"></c:set>
27 <c:set var="request" value="리퀘스트 단위" scope="request"></c:set>
28 <c:set var="session" value="세션 단위" scope="session">
29 </c:set><c:set var="app" value="어플리케이션 단위" scope="application"></c:set>
30
31 <h1>jstl의 출력</h1>
32 <c:out value="\${page }"></c:out><br>
33 <c:out value="\${request }"></c:out><br>
34 <c:out value="\${session }"></c:out><br>
35 <c:out value="\${app }"></c:out><br>
36 </body>
37 </html>
```

## EL의 출력

`\${var }` :

`\${page }` : 페이지 단위

`\${request }` : 리퀘스트 단위

`\${session }` : 세션 단위

`\${app }` : 어플리케이션 단위

## jstl의 출력

페이지 단위

리퀘스트 단위

세션 단위

어플리케이션 단위

# 표현기법이 조금 상이함에 주의할 것

# 자바빈과 jstl

index.jsp

```
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="utf-8">
8   <title>자바빈과 jstl</title>
9 </head>
10 <body>
11   <h1>자바방식</h1>
12   <%
13     beans.Student s1=new beans.Student();
14     s1.setName("이진선");
15     s1.setAge("19");
16     s1.setAddress("서울시 강북구");
17   %>
18   <%=s1 %>
19   <h1>jsp액션태그방식</h1>
20   <jsp:useBean id="s2" class="beans.Student"></jsp:useBean>
21   <jsp:setProperty name="s2" property="name" value="고희선"></jsp:setProperty>
22   <jsp:setProperty name="s2" property="age" value="25"></jsp:setProperty>
23   <jsp:setProperty name="s2" property="address" value="서울시 중구"></jsp:setProperty>
24   <%=s2 %>
25   <h1>jstl방식</h1>
26   <c:set var="s3" value="<%=new beans.Student() %>"></c:set>
27   <c:set target="${s3}" property="name" value="이영준"></c:set>
28   <c:set target="${s3}" property="age" value="31"></c:set>
29   <c:set target="${s3}" property="address" value="서울시 성북구"></c:set>
30   ${s3 }
31 </body>
32 </html>
```

## 자바방식

<<정보출력>> 이름:이진선,나이:19,주소:서울시 강북구

## jsp액션태그방식

<<정보출력>> 이름:고희선,나이:25,주소:서울시 중구

## jstl방식

<<정보출력>> 이름:이영준,나이:31,주소:서울시 성북구

자바빈을 jstl과 결합하면 다소 복잡해질 수 있다.

# jstl:delete

```
index.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="utf-8">
8   <title>jstl의 활용</title>
9 </head>
10 <body>
11   <c:set var="add" value="${3+5 }"></c:set>
12   value="add" : <c:out value="add"></c:out><br>
13   value="\${add}" : <c:out value="\${add}"></c:out><br>
14   <c:remove var="add"></c:remove>
15   삭제후 : <c:out value="\${add}"></c:out><br>
16   <!-- c:set 은 setAttribute의 기능을 수행-->
17   <!-- c:remove 은 removeAttribute의 기능을 수행-->
18 </body>
19 </html>
```

value="add" : add  
value="\\${add}" : 8  
삭제 후 :

**자바빈을 jstl과 결합하면 다소 복잡해질 수 있다.**

# jstl:if

```
index.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!DOCTYPE html>
5 <html>
6   <head>
7     <meta charset="utf-8">
8     <title>jstl if</title>
9   </head>
10  <body>
11    <div id="wrap">
12      <form method="post" action="select.jsp">
13        색상을 선택하세요
14        <select name="color">
15          <option value="1">빨강</option>
16          <option value="2">주황</option>
17          <option value="3">노랑</option>
18          <option value="4">초록</option>
19          <option value="5">파랑</option>
20          <option value="6">남색</option>
21          <option value="7">보라</option>
22        </select>
23        <input type="submit" value="전송">
24      </form>
25    </div>
26  </body>
27 </html>
```

```
select.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!DOCTYPE html>
5 <html>
6   <head>
7     <meta charset="utf-8">
8     <title>색상 출력</title>
9   </head>
10  <body>
11    <h1>일반 자바방식</h1>
12    <%
13      request.setCharacterEncoding("utf-8");
14      String color=request.getParameter("color");
15      int num=Integer.parseInt(color);
16
17      if(num==1){
18        <%
19          <span style="color:red;">빨강</span>
20        <%
21          }
22        <%
23      }
24
25      <h1>jstl을 사용한 경우</h1>
26      <c:if test="${param.color==1}">
27        <span style="color:red;">빨강</span>
28      </c:if>
29    </body>
30 </html>
```

색상을 선택하세요

빨강  
주황  
노랑  
초록  
파랑  
남색  
보라

전송

일반 자바방식

빨강

jstl을 사용한 경우

빨강

조건문이나 함수를 수행하는데 이용 가능하다.

# jstl:if else

```
select.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!DOCTYPE html>
5 <html>
6   <head>
7     <meta charset="utf-8">
8     <title>색상 출력</title>
9   </head>
10  <body>
11    <h1>일반 자바방식</h1>
12    <%
13      request.setCharacterEncoding("utf-8");
14      String color=request.getParameter("color");
15      int num=Integer.parseInt(color);
16
17      if(num==1){
18
19        <span style="color:red;">빨강</span>
20      }else{
21
22        <span style="color:orange;">주황</span>
23      }
24    %>
25
26    <h1>jstl을 사용한 경우</h1>
27    <c:if test="${param.color==1}">
28      <span style="color:red;">빨강</span>
29    </c:if>
30
31    <h1>c:choose 방식</h1>
32    <c:choose>
33      <c:when test="${param.color==1}">
34        <span style="color:red;">빨강</span>
35      </c:when>
36      <c:otherwise>
37        <span style="color:orange;">주황</span>
38      </c:otherwise>
39    </c:choose>
40  </body>
41 </html>
```

색상을 선택하세요

노랑 ▼

전송

빨강  
주황  
노랑  
초록  
파랑  
남색  
보라

## 일반 자바방식

주황

## jstl을 사용한 경우

## c:choose 방식

주황

# HTML과 유사한 형태로 표현이 가능하다.

# jstl:if else if

```
index.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="utf-8">
7     <title>JSTL if-else if</title>
8   </head>
9   <body>
10    <form method="post" action="select.jsp">
11      과일을 선택하세요
12      <select name="fruit">
13        <option value="1">사과</option>
14        <option value="2">메론</option>
15        <option value="3">바나나</option>
16      </select>
17      <input type="submit" value="전송!">
18    </form>
19  </body>
20 </html>
```

```
select.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
4 <!DOCTYPE html>
5 <html>
6   <head>
7     <meta charset="utf-8">
8     <title>다중 조건문</title>
9   </head>
10  <body>
11    <%
12      String str=request.getParameter("fruit");
13      System.out.println(str);
14      if(str.equals("1")){
15        out.print("<h1 style='color:red;'>사과</h1>");
16      }else if(str.equals("2")){
17        out.print("<h1 style='color:green;'>메론</h1>");
18      }else if(str.equals("3")){
19        out.print("<h1 style='color:yellow;'>바나나</h1>");
20      }
21    %>
22    <c:choose>
23      <c:when test="${param.fruit==1 }">
24        <h1 style="color:red;">사과</h1>
25      </c:when>
26      <c:when test="${param.fruit==2 }">
27        <h1 style="color:green;">메론</h1>
28      </c:when>
29      <c:when test="${param.fruit==3 }">
30        <h1 style="color:yellow;">바나나</h1>
31      </c:when>
32    </c:choose>
33  </body>
34 </html>
```

과일을 선택하세요

메론

메론

기존의 방식과 비교해보면 큰 차이가 없음을 알 수 있다.

# jstl:forEach

```
index.jsp
1  <%@ page language="java" contentType="text/html; charset=utf-8"
2    pageEncoding="utf-8"%>
3  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
4  <!DOCTYPE html>
5  <html>
6  <head>
7    <meta charset="utf-8">
8    <title>jstl 반복문</title>
9  </head>
10 <body>
11 <%
12     String[] season={"봄", "여름", "가을", "겨울"};
13
14     //for문방식
15     for(int i=0; i<season.length; i++){
16         out.print("<h3 style='color:red;'>" + season[i] + "</h3>");
17     }
18
19     //향상된 for문방식
20     for(String str:season){
21         out.print("<h3 style='color:blue;'>" + str + "</h3>");
22     }
23     pageContext.setAttribute("array", season);
24 %>
25
26 <!-- jstl forEach방식 -->
27 <c:forEach var="str" items="${array }">
28     <h3 style='color:green;'>${str }</h3>
29 </c:forEach>
30 </body>
31 </html>
```

봄

여름

가을

겨울

봄

여름

가을

겨울

봄

여름

가을

겨울

## java에서의 배열for문을 사용하는 방식



# jstl:forEach

```
<!-- jstl forEach방식 -->  
<!-- varStatus 속성을 이용하면 일반 for문처럼 값을 확인할 수 있다. -->  
<c:forEach var="str" items="${array}" varStatus="iter">  
    <h3 style='color:green;'>${str }:${iter.index }:${iter.count }:${iter.first }:${iter.last }</h3>  
</c:forEach>
```

봄:0:1:true:false

여름:1:2:false:false

가을:2:3:false:false

겨울:3:4:false:true

**status** 패러미터를 사용할 수 있다.

# jstl:forEach if혼합

```
<!-- jstl forEach방식 -->
<!-- varStatus 속성을 이용하면 일반 for문처럼 값을 확인할 수 있다. -->
<c:forEach var="str" items="${array}" varStatus="iter">
  <c:if test="${iter.first}"><div style="border:10px solid red;"></c:if>
  <h3 style='color:green;'>${str }:${iter.index }:${iter.count }:${iter.first }:${iter.last }</h3>
  <c:if test="${iter.last}"></div></c:if>
</c:forEach>
```

봄:0:1:true:false

여름:1:2:false:false

가을:2:3:false:false

겨울:3:4:false:true

**간결한 형태의 제어문을 태그단위에서 사용가능**

# jstl:for

index.jsp

```
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
4 <!DOCTYPE html>
5 <html>
6   <head>
7     <meta charset="utf-8">
8     <title>jstl 반복문</title>
9   </head>
10  <body>
11    <%
12      //for문방식
13      for(int i=0; i<=10; i++){
14        out.print("<h3 style='color:red;'>" + i + "</h3>");
15      }
16    %>
17
18    <!-- jstl for방식 -->
19    <c:forEach var="i" begin="0" end="10" varStatus="status">
20      <h3 style='color:blue;'>${i }</h3>
21    </c:forEach>
22  </body>
23 </html>
```

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

forEach가 아닌 for를 사용할 수도 있다.

# jstl:for 반복조절

```
index.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
4 <!DOCTYPE html>
5 <html>
6   <head>
7     <meta charset="utf-8">
8     <title>jstl 반복문</title>
9   </head>
10  <body>
11    <%
12      //for문방식
13      for(int i=0; i<=10; i+=2){
14        out.print("<h3 style='color:red;'>"+i+"</h3>");
15      }
16    %>
17
18    <!-- jstl for방식 -->
19    <c:forEach var="i" begin="0" end="10" step="2" varStatus="status">
20      <h3 style='color:blue;'>${i }:${status.index }:${status.count }</h3>
21    </c:forEach>
22  </body>
23 </html>
```

0

2

4

6

8

10

0:0:1

2:2:2

4:4:3

6:6:4

8:8:5

10:10:6

**for와 같은 횟수조작이 가능하다.**

# jstl:forTokens

index.jsp

```
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
4 <!DOCTYPE html>
5 <html>
6   <head>
7     <meta charset="utf-8">
8     <title>jstl 토큰처리</title>
9   </head>
10  <body>
11    <%
12      String str="봄,여름,가을,겨울";
13      pageContext.setAttribute("season", str);
14    %>
15
16    <!-- 토큰을 통하여 해당 데이터를 분리해 낼 수 있다. -->
17    <c:forTokens var="data" items="${season }" delims=", ">
18      ${data }<br>
19    </c:forTokens>
20  </body>
21 </html>
```

봄  
여름  
가을  
겨울

특정 문자를 잘라낼 수있다.

# jstl:forTokens

```
index.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
4 <!DOCTYPE html>
5 <html>
6   <head>
7     <meta charset="utf-8">
8     <title>jstl 토큰처리</title>
9   </head>
10  <body>
11    <%
12      String str="봄,여름,가을,겨울";
13      pageContext.setAttribute("season", str);
14
15      String msg="iot,융합.프로그래밍?학과";
16      pageContext.setAttribute("iot", msg);
17    %>
18
19    <!-- 토큰을 통하여 해당 데이터를 분리해 낼 수 있다. -->
20    <c:forTokens var="data" items="${season }" delims=", ">
21      ${data }<br>
22    </c:forTokens>
23
24    <c:forTokens var="data" items="${iot }" delims=".,?">
25      ${data }<br>
26    </c:forTokens>
27  </body>
28 </html>
```

봄  
여름  
가을  
겨울  
iot  
융합  
프로그래밍  
학과

구분해야 하는 문자가 다양한 경우 여러 개 사용가능

# jstl:import

```
<!-- 다른 페이지를 import해오는 방법 -->
<!-- 변수처럼 저장해서 재사용이 가능하다. -->
<c:import url="test.jsp" var="test" scope="application" charEncoding="utf-8"></c:import>
${test }

<a href="app.jsp">다른 페이지로 이동</a>
```

test.jsp

```
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <h1>이 내용은 test.jsp에 적힌 내용입니다.</h1>
```

이 내용은 test.jsp에 적힌 내용입니다.

[다른 페이지로 이동](#)

app.jsp

```
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="utf-8">
7     <title>scope에 저장된 내용을 확인</title>
8   </head>
9   <body>
10    ${applicationScope.test }
11  </body>
12 </html>
```

이 내용은 test.jsp에 적힌 내용입니다.

## scope 단위에 페이지를 저장할 수 있어 편리

# jstl:fmt

index.jsp

```
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
4 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
5 <%@ page import="java.util.*" %>
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <meta charset="utf-8">
10    <title>jstl 포맷사용</title>
11  </head>
12  <body>
13    <!-- 주의!! taglib부분의 jsp가 누락되면 fmt는 jsp를 기반으로 동작하지 않으므로 스크립트릿 및 EL을 사용할 수 없다. -->
14    <c:set var="num" value="1000000" scope="page"></c:set>
15    fmt:number : <fmt:formatNumber value="${num}" type="number"></fmt:formatNumber><br>
16    fmt:number 구분기호 없음 : <fmt:formatNumber value="${num}" type="number" groupingUsed="false"></fmt:formatNumber><br>
17    fmt:number %표기 : <fmt:formatNumber value="0.5" type="percent"></fmt:formatNumber><br>
18    fmt:number 단위표기 : <fmt:formatNumber value="5000" type="currency"></fmt:formatNumber><br>
19    fmt:number 단위변경 : <fmt:formatNumber value="5000" type="currency" currencySymbol="$"></fmt:formatNumber><br>
20    fmt:number 패턴표기 : <fmt:formatNumber value="1234.5678" pattern="000000,000.000000#"></fmt:formatNumber><br>
21    <!-- 0.0# : 0은 자릿수 #은 빈공간인 경우 작성하지 않겠다는 의미 -->
22    <!-- 0,00 : 두자리마다 ,로 구별을 하겠다는 의미, 소수 자릿수를 표기하지 않으면 소수는 나오지 않는다. -->
23    <!-- 0,0.00# : 소수 셋째자리까지 표기하며 매 숫자마다 ,를 찍겠다는 패턴 -->
24    <!-- 0,000.0# : 소수 둘째자리까지 표기하며 천단위 구분기호를 찍겠다는 패턴 -->
25    <!-- 000000,000.000000 : 천단위 구분기호를 사용하여 9자리를 확보하고 빈 공간은 0으로 표기한다. 소수는 7자리까지 표기하며 빈 공간은 0으로 둔다. -->
26
27    <c:set var="date" value="%<new java.util.Date() %>" scope="page"></c:set>
28    \${date } : \${date }<br>
29    fmt:formatDate 기본방식 : <fmt:formatDate value="\${date }"></fmt:formatDate><br>
30    fmt:formatDate time : <fmt:formatDate value="\${date }" type="time"></fmt:formatDate><br>
31    fmt:formatDate both : <fmt:formatDate value="\${date }" type="both"></fmt:formatDate><br>
32    fmt:formatDate 서식지정 : <fmt:formatDate value="\${date }" pattern="yyyy년 MM월 dd일 hh시 mm분 ss초"></fmt:formatDate><br>
33  </body>
34 </html>
```

fmt:number : 1,000,000  
fmt:number 구분기호 없음 : 1000000  
fmt:number %표기 : 50%  
fmt:number 단위표기 : ₩5,000  
fmt:number 단위변경 : \$5,000  
fmt:number 패턴표기 : 000,001,234.567800  
\\${date } : Wed Jul 07 19:21:02 KST 2021  
fmt:formatDate 기본방식 : 2021. 7. 7  
fmt:formatDate time : 오후 7:21:02  
fmt:formatDate both : 2021. 7. 7 오후 7:21:02  
fmt:formatDate 서식지정 : 21년 07월 07일 07시 21분 02초

표현 서식을 지정할 수 있어 매우 편리하다.



# jstl 기본언어설정

```
locale.jsp
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
4 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
5 <%@ page import="java.util.*" %>
6 <!DOCTYPE html>
7 <html>
8 <head>
9   <meta charset="utf-8">
10  <title>jstl Locale 설정</title>
11 </head>
12 <body>
13   <c:set var="date" value="<%=new java.util.Date() %>" scope="page"></c:set>
14   톰캣 서버의 기본 Locale 확인 : <%=response.getLocale() %><br>
15
16   <hr>
17   <fmt:setLocale value="ko_kr"></fmt:setLocale>
18   로케일을 한국어로 설정 한 후 locale확인 : <%=response.getLocale() %><br>
19   통화 : <fmt:formatNumber value="5000" type="currency"></fmt:formatNumber><br>
20   날짜 : <fmt:formatDate value="{date}"></fmt:formatDate><br>
21
22   <hr>
23   <fmt:setLocale value="ja_jp"></fmt:setLocale>
24   로케일을 일본어로 설정 한 후 locale확인 : <%=response.getLocale() %><br>
25   통화 : <fmt:formatNumber value="5000" type="currency"></fmt:formatNumber><br>
26   날짜 : <fmt:formatDate value="{date}"></fmt:formatDate><br>
27
28   <hr>
29   <fmt:setLocale value="en_us"></fmt:setLocale>
30   로케일을 영어로 설정 한 후 locale확인 : <%=response.getLocale() %><br>
31   통화 : <fmt:formatNumber value="5000" type="currency"></fmt:formatNumber><br>
32   날짜 : <fmt:formatDate value="{date}"></fmt:formatDate><br>
33 </body>
34 </html>
```

톰캣 서버의 기본 Locale 확인 : ko\_KR

---

로케일을 한국어로 설정 한 후 locale확인 : ko\_KR  
통화 : ₩5,000  
날짜 : 2021. 7. 7

---

로케일을 일본어로 설정 한 후 locale확인 : ja\_JP  
통화 : ¥ 5,000  
날짜 : 2021/07/07

---

로케일을 영어로 설정 한 후 locale확인 : en\_US  
통화 : \$5,000.00  
날짜 : Jul 7, 2021

**표현 서식을 지정할 수 있어 매우 편리하다.**