

Chapter

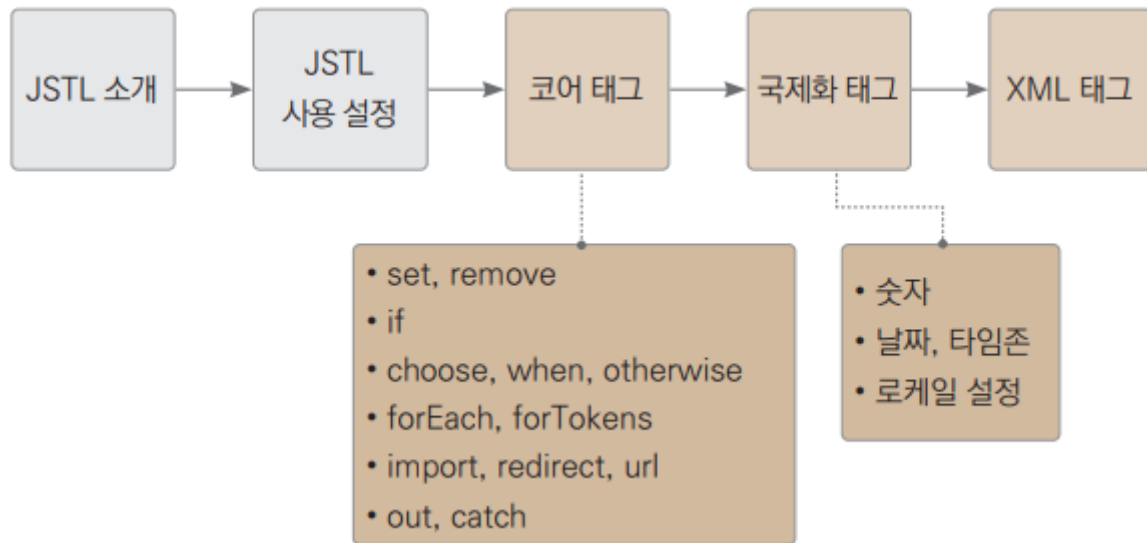
# 11

## JSP 표준 태그 라이브러리 (JSTL)

## ■ 학습 목표

- JSP의 표준 태그 라이브러리인 JSTL을 사용하면 스크립트릿을 사용하지 않고도 제어문, 반복문 등을 사용할 수 있습니다. JSTL이 제공하는 다양한 태그를 학습합니다.

## ■ 학습 순서



## ■ 활용 사례

- JSTL은 모델2 방식의 웹 애플리케이션을 개발할 때 EL과 함께 주로 사용됩니다. EL과 마찬가지로 4가지 영역에 저장된 속성값을 사용할 수 있습니다.
- 서블릿까지 학습을 마친후 14장에서 모델2 방식의 게시판을 제작해보면 활용법을 명확히 알 수 있습니다.



# 11.1 JSTL이란?

## ■ JSTL이란..??

- JSTL(JSP Standard Tag Library)은 JSP에서 빈번하게 사용되는 조건문, 반복문 등을 처리해주는 태그를 모아 표준으로 만들어 놓은 라이브러리
- JSTL을 사용하면 스크립틀릿 없이 태그만으로 작성할 수 있기 때문에 코드가 간결해짐

## ■ JSP 와 JSTL 코드 비교

### JSP로 구현한 구구단

```
<table border="1">
  <%for (int dan = 2; dan <= 9; dan++) {%>
    <tr>
      <%for (int su = 1; su <= 9; su++) {%>
        <td>
          <%=dan%> * <%=su%> = <%=dan * su%>
        </td>
      <%}%>
    </tr>
  <%}%>
</table>
```

### JSTL로 구현한 구구단

```
<table border="1">
  <c:forEach begin="2" end="9" var="dan">
    <tr>
      <c:forEach begin="1" end="9" var="su">
        <td>
          ${dan} * ${su} = ${dan * su}
        </td>
      </c:forEach>
    </tr>
  </c:forEach>
</table>
```



# 11.1 JSTL이란?

## ▣ JSTL에서 제공하는 태그 종류

종류	기능	접두어	URI
Core 태그	변수 선언, 조건문/반복문, URL 처리	c	jakarta.tags.core
Formatting 태그	숫자, 날짜, 시간 포맷 지정	fmt	jakarta.tags.fmt
XML 태그	XML 파싱	x	jakarta.tags.xml
Function 태그	컬렉션, 문자열 처리	fn	jakarta.tags.functions
SQL 태그	데이터베이스 연결 및 쿼리 실행	sql	jakarta.tags.sql

- 이중 Function, SQL 태그는 거의 사용되지 않음



# 11.1 JSTL이란?

## ▣ JSTL 사용을 위한 taglib 지시어

- JSTL 태그 중 core 태그를 사용하기 위한 지시어

JSTL 사용 시 태그 앞에 붙일 접두어입니다.

<c:태그명 /> 형태로 사용합니다.

```
<%@ taglib prefix="c" uri="jakarta.tags.core"%>
```

태그 라이브러리 URI 식별자입니다.



## 11.2 JSTL 사용 설정

### ▣ 라이브러리 다운로드

- JSTL은 JSP의 기본태그가 아니므로 라이브러리 설치 필요
- 메이븐 저장소에서 다운로드
- URL : <https://mvnrepository.com/>
- 접속 후 jakarta.servlet.jsp.jstl 로 검색

The screenshot shows the Maven Repository search results for the query 'jakarta.servlet.jsp.jstl'. The left sidebar shows the 'Repository' list with 'Central' selected, and the 'Group' list with 'jakarta.servlet' selected. The main area displays 'Found 3 results' and lists two items:

Rank	Artifact ID	Group ID	Version	Usage Count	Licenses
1.	Jakarta Standard Tag Library API	jakarta.servlet.jsp.jstl	jakarta.servlet.jsp.jstl-api	513 usages	GPL, EPL
2.	Jakarta Standard Tag Library Implementation	org.glassfish.web	jakarta.servlet.jsp.jstl	38 usages	GPL, EPL



## 11.2 JSTL 사용 설정


### ▣ 라이브러리 다운로드(계속)

- 첫 번째 링크를 클릭한 후 버전을 3.0.0 선택

Central (7) Redhat EA (1)

	Version	Vulnerabilities	Repository	Usages	Date
3.0.x	3.0.0		Central	446	May 18, 2022
2.0.x	2.0.0		Central	42	Nov 21, 2020
	2.0.0-RC1		Central	16	Mar 04, 2020

- jar로 표시된 링크를 클릭해서 라이브러리 다운로드

 **Jakarta Standard Tag Library API » 3.0.0**  
Jakarta Standard Tag Library API

License	EPL 2.0   GPL
Categories	Java Specifications
Tags	jakarta   standard   servlet   jsp   api   specs
HomePage	<a href="https://projects.eclipse.org/projects/ee4j.jstl">https://projects.eclipse.org/projects/ee4j.jstl</a>
Date	May 18, 2022
Files	<a href="#">pom (12 KB)</a>   <a href="#">jar (44 KB)</a>   <a href="#">View All</a>
Repositories	Central   Hortonworks

파일명 : jakarta.servlet.jsp.jstl-api-3.0.0.jar




## ▣ 라이브러리 다운로드(계속)

- 두 번째 링크를 클릭한 후 버전을 3.0.1 선택

Central (5) Redhat EA (1)				
	Version	Vulnerabilities	Repository	Usages
3.0.x	3.0.1		Central	19
	3.0.0		Central	10

- jar로 표시된 링크를 클릭해서 라이브러리 다운로드



### Jakarta Standard Tag Library Implementation » 3.0.1

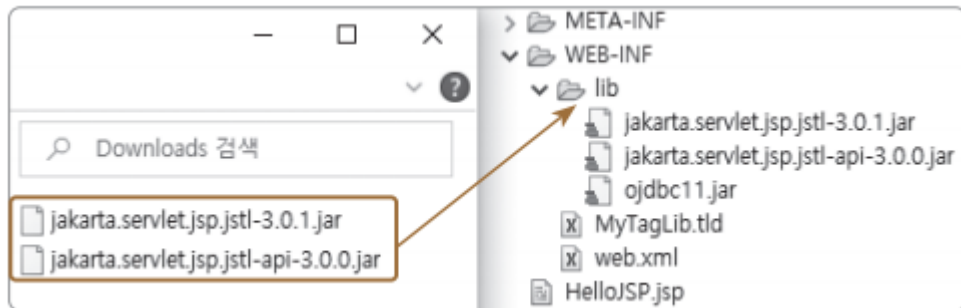
Jakarta Standard Tag Library Implementation

License	EPL 2.0 GPL
Tags	jakarta glassfish servlet web jsp
HomePage	https://projects.eclipse.org/projects/ee4j.jstl
Date	Nov 20, 2022
Files	pom (15 KB) jar (3.5 MB) View All
Repositories	Central ImageJ Public

파일명 : jakarta.servlet.jsp.jstl-3.0.1.jar

## ▣ 라이브러리 다운로드(계속)

- 다운로드 한 jar 파일은 webapp/WEB-INF/lib 폴더로 복사



## ■ 코어(Core) 태그의 종류

태그명	기능
set	EL에서 사용할 변수를 설정합니다. <code>setAttribute()</code> 메서드와 동일한 기능입니다.
remove	설정된 변수를 제거합니다. <code>removeAttribute()</code> 메서드와 동일한 기능입니다.
if	단일 조건문을 주로 처리합니다. <code>else</code> 문이 없다는 단점이 있습니다.
choose	다중 조건을 처리할 때 사용합니다. 하위에 <code>when~otherwise</code> 태그가 있습니다.
forEach	반복문을 처리할 때 사용합니다. 일반 <code>for</code> 문과 향상된 <code>for</code> 문 두 가지 형태로 사용할 수 있습니다.
forTokens	구분자로 분리된 각각의 토큰을 처리할 때 사용합니다. <code>StringTokenizer</code> 클래스와 동일한 기능입니다.
import	외부 페이지를 삽입할 때 사용합니다.
redirect	지정한 경로로 이동합니다. <code>sendRedirect()</code> 메서드와 동일한 기능입니다.
url	경로를 설정할 때 사용합니다.
out	내용을 출력할 때 사용합니다.
catch	예외 처리에 사용합니다.

## ■ <c:set> 태그

- <c:set> 태그는 EL에서 사용할 변수나 자바빈즈를 생성할 때 사용
- 영역에 속성을 저장할 때 사용하는 setAttribute( ) 메서드와 같은 역할
- 아래 2가지 방식으로 사용

```
<c:set var="변수명" value="값" scope="영역" />
```

```
<c:set var="변수명" scope="영역">  
    value 속성에 들어갈 값  
</c:set>
```

## ■ 속성

속성명	기능
var	변수명을 설정합니다.
value	변수에 할당할 값입니다.
scope	변수를 생성할 영역을 지정합니다. page가 기본값입니다.
target	자바빈즈를 설정합니다.
property	자바빈즈의 속성, 즉 멤버 변수의 값을 지정합니다.

## ■ <c:set> 태그

### 예제 11-1] webapp/11JSTL/core/Set1.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="jakarta.tags.core" %> ❶
<html>
<head><title>JSTL - set 1</title></head>
<body>
    <!-- 변수 선언 --> ❷
    <c:set var="directVar" value="100" /> ❸
    <c:set var="elVar" value="${ directVar mod 5}" /> ❹
    <c:set var="expVar" value="<%= new Date() %>" /> ❺
    <c:set var="betweenVar">변수값 이렇게 설정</c:set> ❻
```

- ❶ JSTL 사용을 위한 taglib 지시어
- ❷ <c:set> 태그를 통한 변수 선언
  - 일반 값 입력
  - EL사용
  - 표현식 사용
  - 태그 사이에 값 지정

**[Note]** scope 속성을 통한 영역지정이 없으므로 가장 좁은 page영역에 저장됨



## 11.3 코어(Core) 태그

### ■ <c:set> 태그

#### 예제 11-1] webapp/11JSTL/core/Set1.jsp(계속)

```
<h4>자바빈즈 생성 1 - 생성자 사용</h4>
<c:set var="personVar1" value='<%= new Person("박문수", 50) %>'
      scope="request" /> ⑧
<ul>
  <li>이름 : ${ requestScope.personVar1.name }</li> ⑨
  <li>나이 : ${ personVar1.age}</li>
</ul>

<h4>자바빈즈 생성 2 - target, property 사용</h4>
<c:set var="personVar2" value="<%= new Person() %>" scope="request" /> ⑩
<c:set target="${personVar2 }" property="name" value="정약용" />
<c:set target="${personVar2 }" property="age" value="60" /> ⑪
<ul>
  <li>이름 : ${ personVar2.name }</li>
  <li>나이 : ${ requestScope.personVar2.age }</li>
</ul>
```

⑧ 자바빈즈 생성시 생성자를 통해 초깃값을 설정 후 request 영역에 저장

⑩ target 과 property 속성을 통해 자바빈즈의 값 설정.  
이때는 setter를 통해 값이 설정됨

## ▣ <c:set> 태그

### 예제 11-2] webapp/11JSTL/core/Set2.jsp (컬렉션을 변수로 설정)

```
<h4>List 컬렉션 이용하기</h4>
<%
ArrayList<Person> pList = new ArrayList<Person>(); ①
pList.add(new Person("성삼문", 55));
pList.add(new Person("박팽년", 60)); ②
%>
<c:set var="personList" value="<%= pList %>" scope="request" /> ③
<ul>
    <li>이름 : ${ requestScope.personList[0].name }</li> ④
    <li>나이 : ${ personList[0].age }</li> ⑤
</ul>
```

- ① List 컬렉션 생성 후 Person 객체 추가
- ③ List를 request영역에 저장
- ④ List 는 인덱스를 통해 접근



## 11.3 코어(Core) 태그

### ▣ <c:set> 태그

#### 예제 11-2] webapp/11JSTL/core/Set2.jsp (계속)

```
<h4>Map 컬렉션 이용하기</h4>
<%
Map<String, Person> pMap = new HashMap<String, Person>(); ⑥
pMap.put("personArgs1", new Person("하위지", 65));
pMap.put("personArgs2", new Person("이개", 67));
%>
<c:set var="personMap" value="%= pMap %" scope="request" /> ⑦
<ul>
  <li>이름 : ${ requestScope.personMap.personArgs2.name }</li>
  <li>나이 : ${ personMap.personArgs2.age }</li>
</ul>
```

⑥ Map 컬렉션 생성 후 Person 객체 추가

⑦ request 영역에 Map 저장

⑧ Map은 key를 통해 접근

#### List 컬렉션 이용하기

- 이름 : 성삼문
- 나이 : 55

#### Map 컬렉션 이용하기

- 이름 : 이개
- 나이 : 67



# 11.3 코어(Core) 태그

## ▣ <c:remove> 태그

- <c:set> 태그로 설정한 변수를 제거할 때 사용
- JSP에서 영역의 속성을 제거할 때 사용하는 removeAttribute() 메서드와 같은 역할
- 속성

속성명	기능
var	삭제할 변수명을 설정합니다.
scope	삭제할 변수의 영역을 지정합니다. 지정하지 않으면 모든 영역의 변수가 삭제됩니다.

## 예제 11-3] webapp/11JSTL/core/Remove.jsp

<!-- 변수 선언 -->

```

<c:set var="scopeVar" value="Page Value" /> ②
<c:set var="scopeVar" value="Request Value" scope="request" />
<c:set var="scopeVar" value="Session Value" scope="session" />
<c:set var="scopeVar" value="Application Value" scope="application" />

```

① 4가지 영역에 같은 속성명으로 서로 다른 값을 저장

## ■ <c:remove> 태그

### 예제 11-3] webapp/11JSTL/core/Remove.jsp(계속)

```

<h4>session 영역에서 삭제하기</h4>
<c:remove var="scopeVar" scope="session" /> ⑤
<ul>
  <li>sessionScope.scopeVar : ${ sessionScope.scopeVar }</li> ⑥
</ul>

<h4>scope 지정 없이 삭제하기</h4>
<c:remove var="scopeVar" /> ⑦
<ul>
  <li>scopeVar : ${ scopeVar }</li>
  <li>requestScope.scopeVar : ${ requestScope.scopeVar }</li>
  <li>applicationScope.scopeVar : ${ applicationScope.scopeVar }</li> ⑧
</ul>

```

- ⑤ scope로 session영역을 지정했으므로 지정된 속성만 삭제
- ⑦ 영역 지정이 없으면 모든 영역의 속성을 삭제
- ⑧ 아무값도 출력되지 않음.

#### session 영역에서 삭제하기

- sessionScope.scopeVar :

#### scope 지정 없이 삭제하기

- scopeVar :
- requestScope.scopeVar :
- applicationScope.scopeVar :

## ■ <c:if> 태그

- 자바의 if와 동일하게 제어 구문을 작성할 때 사용
- else가 별도로 없기 때문에 여러 조건을 나열하는 형태로 작성하기에는 어려움

```
<c:if test="조건" var="변수명" scope="영역">  
    조건이 true일 때 출력할 문장  
</c:if>
```

- 속성

속성명	기능
test	if문에서 사용할 조건을 지정합니다.
var	조건의 결과를 저장할 변수명을 지정합니다.
scope	변수가 저장될 영역을 지정합니다.



## 11.3 코어(Core) 태그

### ■ <c:if> 태그

#### 예제 11-4] webapp/11JSTL/core/If.jsp

```
<!-- 변수 선언 -->
<c:set var="number" value="100" />
<c:set var="string" value="JSP" />

<h4>JSTL의 if 태그로 짝수/홀수 판단하기</h4>
<c:if test="${ number mod 2 eq 0 }" var="result"> ❶
    ${ number }는 짝수입니다. <br />
</c:if>
result : ${ result } <br /> ❷

<h4>문자열 비교와 else 구문 흉내내기</h4>
<c:if test="${ string eq 'Java' }" var="result2"> ❸
    문자열은 Java입니다. <br />
</c:if>
<c:if test="${ not result2 }"> ❹
    'Java'가 아닙니다. <br />
</c:if>
```

- ❶ 2개의 변수를 선언
- ❷ number를 2로 나눈 나머지가 0과 같은지 판단한 결과를 변수 result에 저장
- ❸ number가 100이므로 true
- ❹ 문자열끼리 비교한 결과를 result2에 저장
- ❺ not을 이용해 else와 같은 구문으로 작성

#### JSTL의 if 태그로 짝수/홀수 판단하기

는 짝수입니다.  
result : true

#### 문자열 비교와 else 구문 흉내내기

'Java'가 아닙니다.

## ▣ <c:if> 태그

### 예제 11-4] webapp/11JSTL/core/If.jsp(계속)

```

<h4>조건식 주의사항</h4>
<c:if test="100" var="result3"> ⑥
    EL이 아닌 정수를 지정하면 false
</c:if>
result3 : ${ result3 } <br />
<c:if test="tRuE" var="result4"> ⑦
    대소문자 구분 없이 "tRuE"인 경우 true <br />
</c:if>
result4 : ${ result4 } <br />
<c:if test=" ${ true } " var="result5"> ⑧
    EL 양쪽에 빈 공백이 있는 경우 false<br />
</c:if>
result5 : ${ result5 } <br />

```

⑥ 주의사항1 : EL이 아닌 일반값이 오면 false

⑦ 주의사항2 : 문자열 'true'는 대소문자 관계없이 true. 가령 tRuE라고 써도 동일한 결과.

⑧ 주의사항3 : test 속성에 EL을 통해 조건식을 쓸 때 공백이 삽입되면 무조건 false를 반환

형식 1 : <c:if test="\${\_true\_}" var="result5"> : 정상 출력

형식 2 : <c:if test="\_\${true}\_ " var="result5"> : 에러 발생

#### 조건식 주의사항

result3 : false

대소문자 구분 없이 "tRuE"인 경우 true

result4 : true

result5 : false

## 11.3 코어(Core) 태그

### ■ <c:choose>, <c:when>, <c:otherwise> 태그

- 다중 조건을 판단해야 할 때 사용
- 하위 태그로 <c:when>, <c:otherwise> 태그를 함께 사용

```
<c:choose>                                     // 대응하는 자바 코드
  <c:when test="조건1">조건1을 만족하는 경우</c:when>  ——— if (조건1) { ...
  <c:when test="조건2">조건2을 만족하는 경우</c:when>  ——— } else if (조건2) {...
  <c:otherwise>아무 조건도 만족하지 않는 경우</c:otherwise> — } else { ... }
</c:choose>
```

- <c:if>와 동일하게 조건은 test 속성을 사용

### 예제 11-5] webapp/11JSTL/core/Choose.jsp

```
<!-- 변수 선언 -->
<c:set var="number" value="100" /> ①

<h4>choose 태그로 홀짝 판단하기</h4>
<c:choose> ②
  <c:when test="${ number mod 2 eq 0 }">
    ${ number }는 짝수입니다.
  </c:when>
```

- ① 변수를 생성
- ② number를 2로 나눈 나머지가 0인지 판단하는 조건



## 11.3 코어(Core) 태그

### 예제 11-5] webapp/11JSTL/core/Choose.jsp(계속)

```
<c:otherwise>
    ${ number }는 홀수입니다.
</c:otherwise>
</c:choose>

<c:if test="${ not (empty param.kor or empty param.eng or empty param.math) }">
    <!-- 평균 계산 --> ⑤
    <c:set var="avg" value="${ (param.kor + param.eng + param.math) / 3}" />
    평균 점수는 ${avg }으로 ⑥
    <!-- 학점 출력 --> ⑦
    <c:choose>
        <c:when test="${ avg >= 90 }">A 학점</c:when>
        <c:when test="${ avg >= 80 }">B 학점</c:when> ⑧
        <c:when test="${ avg ge 70 }">C 학점</c:when> ⑨
        <c:when test="${ avg ge 60 }">D 학점</c:when>
        <c:otherwise>F 학점</c:otherwise>
    </c:choose>
    입니다.
</c:if>
```

② 0이 아닌 경우 홀수가 출력.  
else문과 동일한 기능.

⑤ get방식으로 전송된 국,영,수 점  
수를 EL로 받은 후 평균값 계산

⑥ 평균값 출력

⑦ 학점 판단 후 출력

localhost:8081/MustHaveJSP/11JSTL/core/Choose.jsp?kor=72&eng=88&math=92

① choose 태그로 출력 판단하기

100는 짝수입니다.

국,영,수 점수를 입력하면 평균을 내어 학점 출력

국어 :

영어 :

수학 :

학점 구하기

③ 평균 점수는 84.0으로 B 학점 입니다.

## ■ <c:forEach> 태그

- 반복을 위해 사용되는 태그로 일반 for문, 향상된 for문 2가지로 사용 가능
- 일반 for문 형태

```
<c:forEach var="변수명" begin="시작값" end="마지막값" step="증가값" />  
for ( int i = 0 ; i < 100 ; i += 2 ) { ... }
```

- 향상된 for문 형태

```
<c:forEach var="변수명" items="컬렉션 혹은 배열" />  
for ( int number : numbers ) { ... }
```



## ▣ <c:forEach> 태그

- 속성

속성명	기능
var	변수명을 지정합니다.
items	반복을 위한 객체를 지정합니다. 배열, 컬렉션 등을 지정할 수 있습니다.
begin	시작값을 지정합니다.
end	종료값을 지정합니다.
step	증가할 값을 지정합니다.
varStatus	루프의 현재 상태를 알려주는 변수의 이름을 지정합니다.

## ■ <c:forEach> 태그

- varStatus 속성
  - var속성과 별개로 루프의 자세한 상태 정보를 반환
  - for문의 형태에 따라 확인할 수 있는 값이 조금 다름

속성명	일반 for문	향상된 for문
current	var에 지정한 현재 루프의 변수값 반환	현재 루프의 실제 요소를 반환
index	var에 지정한 현재 루프의 변수값 반환	현재 루프의 인덱스를 표시(0~마지막)
count	실제 반복 횟수(1~마지막)	일반 for문과 동일함
first	루프의 처음일 때 true 반환	일반 for문과 동일함
last	루프의 마지막일 때 true 반환	일반 for문과 동일함



## 11.3 코어(Core) 태그

### ■ <c:forEach> 태그 - 일반 for문 형태

#### 예제 11-6] webapp/11JSTL/core/ForEachNormal.jsp

```
<h4>일반 for문 형태의 forEach 태그</h4>
<c:forEach begin="1" end="3" step="1" var="i"> ❶
    <p>반복 ${ i }입니다</p> ❷
</c:forEach>

<h4>varStatus 속성 살펴보기</h4>
<table border="1">
<c:forEach begin="3" end="5" var="i" varStatus="loop"> ❸
    <tr>
        <td>count : ${ loop.count }</td>
        <td>index : ${ loop.index }</td>
        <td>current : ${ loop.current }</td> ❹
        <td>first : ${ loop.first }</td>
        <td>last : ${ loop.last }</td> ❺
    </tr>
</c:forEach>
</table>
```

- ❶ 1~3까지 1씩 증가하며 반복. 증가하는 값은 변수 i에 저장
- ❸ 3~5까지 증가하는 반복문
- ❹ varStatus를 통한 상태값
  - count : 실제반복횟수(1부터)
  - index : 변수 i의 현재값
  - current : 변수 i의 현재값

#### 일반 for문 형태의 forEach 태그

반복 1입니다

반복 2입니다

반복 3입니다

#### varStatus 속성 살펴보기

count : 1	index : 3	current : 3	first : true	last : false
count : 2	index : 4	current : 4	first : false	last : false
count : 3	index : 5	current : 5	first : false	last : true



## 11.3 코어(Core) 태그

### ■ <c:forEach> 태그 - 향상된 for문 형태1

#### 예제 11-7] webapp/11JSTL/core/ForEachExtend1.jsp

```
<%  
String[] rgba = {"Red", "Green", "Blue", "Black"}; ❶  
%>  
<c:forEach items="<%= rgba %>" var="c" ❷  
    <span style="color:${ c };">${ c }</span>  
</c:forEach>  
  
<h4>varStatus 속성 살펴보기</h4>  
<table border="1">  
<c:forEach items="<%= rgba %>" var="c" varStatus="loop" ❸  
    <tr>  
        <td>count : ${ loop.count }</td>  
        <td>index : ${ loop.index }</td>  
        <td>current : ${ loop.current }</td> ❹  
        <td>first : ${ loop.first }</td>  
        <td>last : ${ loop.last }</td>  
    </tr>  
</c:forEach>  
</table>
```

- 1 문자열 배열 생성
- 2 배열의 크기만큼 반복하며 배열의 원소를 c에 할당
- 4 varStatus를 통한 상태값
  - count : 실제 반복 횟수(1부터)
  - index : 현재 루프의 인덱스(0부터)
  - current : 변수 c의 현재값

#### 향상된 for문 형태의 forEach 태그

Red Green Blue Black

#### varStatus 속성 살펴보기

count : 1	index : 0	current : Red	first : true	last : false
count : 2	index : 1	current : Green	first : false	last : false
count : 3	index : 2	current : Blue	first : false	last : false
count : 4	index : 3	current : Black	first : false	last : true

## ▣ <c:forEach> 태그 - 향상된 for문 형태2

예제 11-8] webapp/11JSTL/core/ForEachExtend2.jsp

```
<h4>List 컬렉션 사용하기</h4>
<%
LinkedList<Person> lists = new LinkedList<Person>(); ❶
lists.add(new Person("맹사성", 34));
lists.add(new Person("장영실", 44));
lists.add(new Person("신숙주", 54));
%>
<c:set var="lists" value="<%= lists %>"/> ❷
<c:forEach items="${ lists }" var="list"> ❸
<li>
    이름 : ${ list.name }, 나이 : ${ list.age } ❹
</li>
</c:forEach>
```

- ❶ List 컬렉션 생성 및 Person객체 추가
- ❷ 변수로 선언
- ❸ List에 저장된 갯수만큼 반복하여 var속성에 지정한 변수로 할당
- ❹ 멤버 변수명을 통해 getter를 호출하여 값 출력

### List 컬렉션 사용하기

- 이름 : 맹사성, 나이 : 34
- 이름 : 장영실, 나이 : 44
- 이름 : 신숙주, 나이 : 54

## ▣ <c:forEach> 태그 - 향상된 for문 형태2(계속)

### 예제 11-8] webapp/11JSTL/core/ForEachExtend2.jsp

```
<h4>Map 컬렉션 사용하기</h4>
<%
Map<String,Person> maps = new HashMap<String,Person>(); ⑤
maps.put("1st", new Person("맹사성", 34));
maps.put("2nd", new Person("장영실", 44));
maps.put("3rd", new Person("신숙주", 54));
%>
<c:set var="maps" value="<%= maps %>" /> ⑥
<c:forEach items="${ maps }" var="map"> ⑦
    <li>Key => ${ map.key } <br /> ⑧
        Value => 이름 : ${ map.value.name }, 나이 : ${ map.value.age }</li> ⑨
</c:forEach>
```

- ⑤ Map 컬렉션 생성 및 Person객체 추가
- ⑥ 변수 생성
- ⑦ Map의 갯수만큼 반복
- ⑧ 컬렉션을 순회하면서 Key와 Value를 출력

#### Map 컬렉션 사용하기

- Key => 1st  
Value => 이름 : 맹사성, 나이 : 34
- Key => 3rd  
Value => 이름 : 신숙주, 나이 : 54
- Key => 2nd  
Value => 이름 : 장영실, 나이 : 44

## 11.3 코어(Core) 태그

### ■ <c:forTokens> 태그

- 구분자를 기준으로 문자열을 나눠 토큰의 개수만큼 반복

```
<c:forTokens items="문자열" delims="문자열 구분자" var="변수명" />
```

#### 토큰이란?

일반적으로 토큰은 ‘문법적으로 의미있는 최소 단위’를 말함.

전화번호를 예로 든다면 “010-1234-5678”을 구분자 -(하이픈)으로 분리하면 “010”, “1234”, “5678”이 각각 토큰이 됨

### 예제 11-9] webapp/11JSTL/core/ForTokens.jsp

```
<%  
String rgba = "Red,Green,Blue,Black"; ①  
%>  
<h4>JSTL의 forTokens 태그 사용</h4>  
<c:forTokens items="<%= rgba %>" delims="," var="color"> ②  
    <span style="color:${ color };">${ color }</span> <br /> ③  
</c:forTokens>
```

- ① 콤마로 구분된 문자열 정의
- ② delims 속성에 구분자인 콤마를 입력
- ③ 구분자를 기준으로 분리된 토큰은 var속성의 변수에 저장됨

## ■ <c:import> 태그

- <jsp:include> 액션 태그와 같이 외부 파일을 현재 위치에 삽입할 때 사용
- 웹 애플리케이션에 속하지 않은 외부의 페이지도 삽입할 수 있음

```
<c:import url="페이지 경로 혹은 URL" scope="영역" />
```

- 다음과 같이 var속성을 사용하면 선언과 삽입을 분리할 수 있음

```
<c:import url="페이지 경로 혹은 URL" var="변수명" scope="영역" />
${ 변수명 }
```

- 매개변수로 전달할 값이 있다면 <c:param>태그를 사용하면 됨

```
<c:import url="페이지 경로 혹은 URL?매개변수1=값1" >
    <c:param name="매개변수2" value="값2" />
</c:import>
```





## 11.3 코어(Core) 태그

### ■ <c:import> 태그

#### 예제 11-10] webapp/11JSTL/inc/OtherPage.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<h4>OtherPage.jsp</h4>
<ul>
    <li>저장된 값 : ${ requestVar }</li> ❶
    <li>매개변수 1 : ${ param.user_param1 }</li>
    <li>매개변수 2 : ${ param.user_param2 }</li>
</ul>
```

❶ 포함시킬 2개의 파일을  
먼저 준비

#### 예제 11-11] webapp/11JSTL/inc/GoldPage.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<c:import url="https://goldenrabbit.co.kr/" /> ❶
```

## ▣ <c:import> 태그

예제 11-12] webapp/11JSTL/core/Import.jsp

```
<body>
  <c:set var="requestVar" value="MustHave" scope="request" /> ❶
  <c:import url="/11JSTL/inc/OtherPage.jsp" var="contents"> ❷
    <c:param name="user_param1" value="JSP" /> ❸
    <c:param name="user_param2" value="기본서" />
  </c:import>

  <h4>다른 문서 삽입하기</h4>
  ${ contents } ❹

  <h4>외부 자원 삽입하기</h4>
  <iframe src="../../inc/GoldPage.jsp" style="width:100%;height:600px;"></iframe> ❺
</body>
```

- ❶ request 영역에 속성 저장
- ❷ OtherPage를 임포트 한 후 변수에 저장
- ❸ 임포트 한 페이지로 파라미터 전달
- ❹ 임포트 한 문서를 현재위치에 삽입
- ❺ 골든레빗 웹 사이트를 통째로 삽입



## 11.3 코어(Core) 태그

### ■ <c:redirect> 태그

- response 내장 객체의 sendRedirect()와 동일하게 페이지 이동
- 매개변수를 전달하고 싶다면 <c:import> 태그와 동일하게 <c:param> 태그를 사용

```
<c:redirect url="이동할 경로 및 URL" />
```

### 예제 11-13] webapp/11JSTL/core/Redirect.jsp

```
<body>
  <c:set var="requestVar" value="MustHave" scope="request" /> ①
  <c:redirect url="/11JSTL/inc/OtherPage.jsp"> ②
    <c:param name="user_param1" value="출판사" /> ③
    <c:param name="user_param2" value="골든래빗" />
  </c:redirect>
</body>
```

- ① request 영역에 변수 저장
- ② OtherPage로 리다이렉트
- ③ 매개변수 전달





## 11.3 코어(Core) 태그

### ■ <c:url> 태그

- 지정한 경로와 매개변수를 이용해서 컨텍스트 루트를 포함한 URL을 생성
- 생성된 URL은 <a> 태그의 href 속성이나, <form> 태그의 action 속성에 사용

```
<c:url value="설정된 경로" scope="영역" />
```

### 예제 11-14] webapp/11JSTL/core/Url.jsp

```
<body>
  <h4>url 태그로 링크 걸기</h4>
  <c:url value="/11JSTL/inc/OtherPage.jsp" var="url"> ❶
    <c:param name="user_param1" value="Must" /> ❷
    <c:param name="user_param2">Have</c:param> ❸
  </c:url>
  <a href="${url }">OtherPage.jsp 바로가기</a> ❹
</body>
```

- ❶ URL을 생성. var속성의 변수에 저장됨.
- ❷ 매개변수 전달
- ❸ <a> 태그에 적용

**[Note]** 예제 실행후 소스보기를 해보면 href="/MustHaveJSP/11JSTL/..." 과 같이 컨텍스트 루트 경로가 포함되어 있는것을 볼 수 있다.

## ■ <c:out> 태그

- JSP의 표현식처럼 변수를 출력할 때 사용

```
<c:out value="출력할 변수" default="기본값" escapeXml="특수문자 처리 유무" />
```

- 속성

속성명	기능
value	출력할 변수를 지정합니다.
escapeXml	특수 문자를 변환할지 여부를 결정합니다. 기본값은 true로 특수 기호를 그대로 출력합니다.
default	value 속성에 값을 지정하지 않을 경우 출력할 값을 지정합니다.

## ▣ <c:out> 태그

예제 11-15] webapp/11JSTL/core/Out.jsp

```
<c:set var="iTag"> ❶
```

i 태그는 <i>기울임</i>을 표현합니다.

```
</c:set>
```

```
<h4>기본 사용</h4>
```

```
<c:out value="${ iTag }" /> ❷
```

```
<h4>escapeXml 속성</h4>
```

```
<c:out value="${ iTag }" escapeXml="false" /> ❸
```

```
<h4>default 속성</h4>
```

```
<c:out value="${ param.name }" default="이름 없음" /> ❹
```

```
<c:out value="" default="빈 문자열도 값입니다." /> ❺
```

- ❶ HTML 태그가 포함된 문자열을 변수 생성
- ❷ 기본 상태에서는 <i>태그가 그대로 출력
- ❸ escapeXml 속성을 false로 지정하면 마크업이 적용됨
- ❹ 변수값이 null일때는 default 속성에 적용한 기본값이 출력됨
- ❺ 빈 문자열은 null이 아니므로 기본값은 출력되지 않음

default 속성

이름 없음 ❹

❺



## 11.3 코어(Core) 태그

### ■ <c:catch> 태그

- 예외 처리를 위해 사용됨
- Java코드 혹은 EL에서 발생한 예외를 처리할 때 사용

#### 예제 11-16] webapp/11JSTL/core/Catch.jsp

```
<h4>자바 코드에서의 예외</h4>
<%
int num1 = 100; ❶
%>
<c:catch var="eMessage"> ❷
    <%
        int result = num1 / 0; ❸
    %>
</c:catch>
예외 내용 : ${ eMessage } ❹
```

```
<c:catch var="변수명">
    실행 코드
</c:catch>
```

```
<h4>EL에서의 예외</h4>
<c:set var="num2" value="200" /> ❺

<c:catch var="eMessage"> ❻
    ${ "일" + num2 } ❼
</c:catch>
예외 내용 : ${ eMessage } ❽
```

#### 자바 코드에서의 예외

예외 내용 : java.lang.ArithmeticException: / by zero

#### EL에서의 예외

예외 내용 : java.lang.NumberFormatException: For input string: "일"

## 11.4 국제화(Formatting) 태그

### ■ Formatting 태그란..??

- 국제화 태그로, 국가별로 다양한 언어, 날짜, 시간, 숫자 형식을 설정 할 때 사용
- 접두어로는 fmt를 사용

```
<%@ taglib prefix="fmt" uri="jakarta.tags.fmt" %>
```

- 국제화 태그의 종류

분류	태그명	기능
숫자 포맷	formatNumber	숫자 포맷을 설정합니다.
	parseNumber	문자열을 숫자 포맷으로 변환합니다.
날짜 포맷	formatDate	날짜나 시간의 포맷을 설정합니다.
	parseDate	문자열을 날짜 포맷으로 변환합니다.
타임존 설정	setTimeZone	시간대 설정 정보를 변수에 저장합니다.
	timeZone	시간대를 설정합니다.
로케일 설정	setLocale	통화 기호나 시간대를 설정한 지역에 맞게 표시합니다.
	requestEncoding	요청 매개변수의 문자셋을 설정합니다.



## 11.4 국제화(Formatting) 태그

### ■ 숫자 포매팅 및 파싱 - <fmt:formatNumber>

- 숫자 포맷과 관련한 태그는 <fmt:formatNumber>와 <fmt:parseNumber> 가 있음

```
<fmt:formatNumber value="출력할 숫자" type="문자열 양식 패턴" var="변수 설정"
groupingUsed="구분 기호 사용 여부" pattern="숫자 패턴" scope="영역" />
```

- 속성

속성명	기능
value	출력할 숫자를 설정합니다.
type	출력 양식을 설정합니다. percent(퍼센트), currency(통화), number(일반 숫자, 기본값) 등을 지원합니다.
var	출력할 숫자를 변수에 저장합니다. 해당 속성 사용 시 즉시 출력되지 않고, 원하는 위치에 출력할 수 있습니다.
groupingUsed	세 자리마다 콤마를 출력할지 여부를 결정합니다. 기본값은 true입니다.
pattern	출력할 숫자의 양식을 패턴으로 지정합니다.
scope	변수를 저장할 영역을 지정합니다.

## 11.4 국제화(Formatting) 태그

### ■ 숫자 포매팅 및 파싱 - <fmt:parseNumber>

```
<fmt:parseNumber value="파싱할 문자열" type="출력 양식" var="변수 설정"
integerOnly="정수만 파싱" pattern="패턴" scope="영역" />
```

#### ● 속성

속성명	기능
value	변환할 문자열을 설정합니다.
type	문자열의 타입을 설정합니다. 기본값은 number(숫자)입니다.
var	출력할 값을 변수에 저장합니다.
pattern	문자열의 양식을 패턴으로 지정합니다.
scope	변수를 저장할 영역을 지정합니다.
integerOnly	정수 부분만 표시할지 여부를 결정합니다. 기본값은 false입니다.

# 11.4 국제화(Formatting) 태그

## ■ 숫자 포매팅 및 파싱

예제 11-17] webapp/11JSTL/etc/Fmt1.jsp

[Note] <%@ taglib prefix="fmt" uri="jakarta.tags.fmt" %>  
지시어가 상단에 포함되야 함

<h4>숫자 포맷 설정</h4>

<c:set var="number1" value="12345" /> ①

coma 0 : <fmt:formatNumber value="\${ number1 }" /><br /> ②

coma X : <fmt:formatNumber value="\${ number1 }" groupingUsed="false" /><br /> ③

<fmt:formatNumber value="\${number1 }" type="currency" var="printNum1" /> ④

통화기호 : \${ printNum1 } <br /> ⑤

<fmt:formatNumber value="0.03" type="percent" var="printNum2" /> ⑥

퍼센트 : \${ printNum2 } ⑦

<h4>문자열을 숫자로 변경</h4>

<c:set var="number2" value="6,789.01" /> ⑧

<fmt:parseNumber value="\${ number2 }" pattern="00,000.00" var="printNum3" /> ⑨

소수점까지 : \${ printNum3 } <br />

<fmt:parseNumber value="\${ number2 }" integerOnly="true" var="printNum4" /> ⑩

정수 부분만 : \${ printNum4 }

① 변수 선언

② 세자리 마다 콤마출력

④ 통화기호 출력

⑥ 백분율(%)

⑧ 콤마와 점이 포함된 문자  
열

⑨ 소수점까지 출력

⑩ 정수부만 출력

## 11.4 국제화(Formatting) 태그

### ■ 날짜 포맷 및 타임존

- <fmt:formatDate> 태그는 날짜와 시간 포맷을 지정하는 태그

```
<fmt:formatDate value="출력할 날짜" type="출력 양식" var="변수 설정"
    dateStyle="날짜 스타일" timeStyle="시간 스타일" pattern="날짜 패턴" scope="영역"
/>
```

- 속성

속성명	기능
value	출력할 값을 설정합니다.
type	출력 시 날짜(date), 시간(time), 날짜 및 시간(both) 세 가지 중 선택할 수 있습니다.
var	출력할 숫자를 변수에 저장합니다.
dateStyle	날짜 스타일을 지정합니다. default, short, medium, long, full 중 선택할 수 있습니다.
timeStyle	시간 스타일을 지정합니다. default, short, medium, long, full 중 선택할 수 있습니다.
pattern	출력할 날짜 및 시간의 양식을 패턴으로 직접 지정합니다.
scope	변수를 저장할 영역을 지정합니다.



## 11.4 국제화(Formatting) 태그

### ■ 날짜 포맷 및 타임존

#### 예제 11-18] webapp/11JSTL/etc/Fmt2.jsp

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

<h4>날짜 포맷</h4>
full : <fmt:formatDate value="${ today }" type="date" dateStyle="full"/> <br />
short : <fmt:formatDate value="${ today }" type="date" dateStyle="short"/> <br />
long : <fmt:formatDate value="${ today }" type="date" dateStyle="long"/> <br />
default : <fmt:formatDate value="${ today }" type="date" dateStyle="default"/>

<h4>시간 포맷</h4>
full : <fmt:formatDate value="${ today }" type="time" timeStyle="full"/> <br />
short : <fmt:formatDate value="${ today }" type="time" timeStyle="short"/> <br />
long : <fmt:formatDate value="${ today }" type="time" timeStyle="long"/> <br />
default : <fmt:formatDate value="${ today }" type="time" timeStyle="default"/>

<h4>날짜/시간 표시</h4>
<fmt:formatDate value="${ today }" type="both" dateStyle="full" timeStyle="full"/> ❷
<br />
<fmt:formatDate value="${ today }" type="both" pattern="yyyy-MM-dd hh:mm:ss"/> ❸
```

❶ Date 객체를 변수로 선언

❷ 날짜 포맷을 출력하기 위해 type을 date로 설정

❸ 시간 포맷을 출력하기 위해 type을 time으로 설정

❹ 날짜와 시간을 동시에 출력

❺ 스타일 대신 pattern 속성을 직접 지정할 수 있음

## 11.4 국제화(Formatting) 태그

### 로케일 설정

예제 11-19] webapp/11JSTL/etc/Fmt3.jsp

```
<h4>로케일 설정</h4>
```

```
<c:set var="today" value="%<%= new java.util.Date() %>" /> ①
```

```
한글로 설정 : <fmt:setLocale value="ko_kr" /> ②
```

```
<fmt:formatNumber value="10000" type="currency" /> /
```

```
<fmt:formatDate value="{ today }" /><br />
```

```
일어로 설정 : <fmt:setLocale value="ja_JP" /> ③
```

```
<fmt:formatNumber value="10000" type="currency" /> /
```

```
<fmt:formatDate value="{ today }" /><br />
```

```
영어로 설정 : <fmt:setLocale value="en_US" /> ④
```

```
<fmt:formatNumber value="10000" type="currency" /> /
```

```
<fmt:formatDate value="{ today }" /><br />
```

① Date 객체를 변수로 선언

② 한글로 설정

③ 일어로 설정

④ 영어로 설정

## ■ XML 태그란..??

- XML 문서를 처리하기 위한 것으로, XML 파싱 및 출력, 흐름 제어 등의 기능을 제공
- 접두어는 “x”를 사용하는 다음의 지시어가 필요

```
<%@ taglib prefix="x" uri="jakarta.tags.xml"%>
```

**[Note]** XML(eXtensible Markup Language)이란 확장이 가능한 마크업 언어

HTML과는 달리 XML은 태그를 개발자가 직접 정의할 수 있고, 데이터를 저장하고 전달할 목적으로 만들어졌음. 그래서 웹 애플리케이션 사이에서 데이터를 전달하는 역할을 함

- xml 태그의 종류

태그명	기능
out	select 속성에 지정한 XPath 표현식의 결과를 출력합니다.
parse	XML을 파싱할 때 사용합니다.



# 11.5 XML 태그

## ■ Xalan.jar 라이브러리 추가

- 메이븐 저장소에 접속
- <https://mvnrepository.com/>
- “Xalan”으로 검색하여 [Xalan Java] 링크를 클릭 ⇒ 2.7.2 를 선택

**MVN REPOSITORY**

Xalan

Search

Repository

Central 26

Redhat GA 10

Spring Lib M 6

Sonatype 5

Spring Plugins 5

JBoss 3rd-party 4

WSO2 Dist 4

Found 56 results

Sort: **relevance** | popular | newest

1. Xalan Java

xalan » xalan

Xalan-Java is an XSLT processor text, or other XML document type, Version 1.0 and XML Path Language, the command line, in an applet.

License

Apache 2.0

Ranking

#345 in MvnRepository (See Top Artifacts)

Used By

1,261 artifacts

Central (11)

Atlassian 3rd-P Old (1)

Redhat GA (9)

Redhat EA (1)

JBoss 3rd-party (2)

JBoss Public (2)

Geomajas (2)

Alfresco (2)

XWiki Externals (1)

ICM (2)

	Version	Vulnerabilities	Repository	Usages	Date
2.7.x	2.7.2	1 vulnerability	Central	337	Jul 24, 2014
	2.7.1	2 vulnerabilities	Central	640	Sep 30, 2008
	2.7.0	2 vulnerabilities	Central	272	Jan 31, 2014






# 11.5 XML 태그

## ■ Xalan.jar 라이브러리 추가

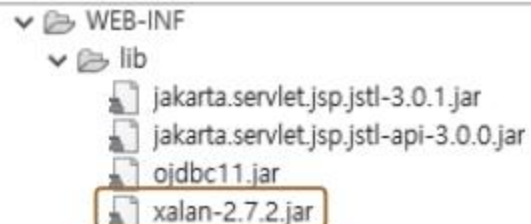
- jar 버튼 클릭하여 다운로드
- webapp/WEB-INF/lib 폴더에 복사

Home » xalan » xalan » 2.7.2

 **Xalan Java » 2.7.2**

Xalan-Java is an XSLT processor for transforming XML documents into HTML, text, or other XML document types. It implements XSL Transformations (XSLT) Version 1.0 and XML Path Language (XPath) Version 1.0 and can be used from the command line, in an applet or a servlet, or as a module in other program.

License	Apache 2.0
HomePage	<a href="http://xml.apache.org/xalan-j/">http://xml.apache.org/xalan-j/</a>
Date	Jul 24, 2014
Files	<a href="#">pom (1 KB)</a> <a href="#">jar (3.0 MB)</a> <a href="#">View All</a>
Repositories	<a href="#">Central</a> <a href="#">Inepex</a> <a href="#">Kyligence Public</a> <a href="#">Redhat GA</a> <a href="#">Sonatype</a>





## 11.5 XML 태그

### 예제 11-20] webapp/11JSTL/inc/BookList.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<booklist>
  <book>
    <name>사피엔스</name>
    <author>유발 하라리</author>
    <price>19800</price>
  </book>
  <book>
    <name>총, 군, 쇠</name>
    <author>제러드 다이아몬드</author>
    <price>25200</price>
  </book>
</booklist>
```

파싱할 XML 문서 작성



## 11.5 XML 태그

### 예제 11-20] webapp/11JSTL/etc/Xml.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="jakarta.tags.core"%>
<%@ taglib prefix="x" uri="jakarta.tags.xml"%>
```

```
<c:set var="booklist"> ❶
    <c:import url="/11JSTL/inc/BookList.xml" charEncoding="UTF-8" /> ❷
</c:set>
<x:parse xml="${booklist}" var="blist" /> ❸

<h4>파싱 1</h4>
제목 : <x:out select="$blist/booklist/book[1]/name" /> <br />
저자 : <x:out select="$blist/booklist/book[1]/author" /> <br />
가격 : <x:out select="$blist/booklist/book[1]/price" /> <br />
```

상단에 taglib 지시어 추가

❶ import를 이용해서 xml 문서의  
내용을 변수로 선언

❷ 변수의 내용을 파싱해 줄  
<x:parse> 태그를 선언

❸ select 속성에 XPath를 이용해  
파싱

## 예제 11-20] webapp/11JSTL/etc/Xml.jsp(계속)

```
<x:choose> ⑦
  <x:when select="$item/price >= 20000">
    2만 원 이상 <br />
  </x:when>
  <x:otherwise>
    2만 원 미만 <br />
  </x:otherwise>
</x:choose>
```

```
<x:forEach select="$blist/booklist/book" var="item"> ⑧
<tr>
  <td><x:out select="$item/name" /></td>
  <td><x:out select="$item/author" /></td>
  <td><x:out select="$item/price" /></td>
  <td><x:if select="$item/name = '총,균,쇠'">구매함</x:if></td> ⑨
</tr>
</x:forEach>
```

⑦ <x:choose> 태그를 이용해  
가격이 20,000원 이상인지 판단

⑧ <x:forEach> 태그를 사용하여  
반복되는 노드가 있을때 반복

⑨ <x:if> 태그를 사용하여 문자  
열을 비교



# 학습 마무리

## ■ 핵심요약

- Core 태그 : 프로그래밍 언어에서 가장 기본이 되는 변수 선언, 조건문, 반복문 등을 대체하는 태그를 제공
- Formatting 태그 : 국가별로 다양한 언어, 날짜와 시간, 숫자 형식을 설정할 때 사용
- XML 태그 : XML 문서를 처리하기 위한 태그들로 XML 파싱, 출력, 흐름 제어 등의 기능을 제공

