

표현 언어와 JSTL

- JSTL

JSTL

❖ JSTL : JSP Standard Tag Library

- JSP 코드의 간결화
- JSP 가독성 향상

JSTL 라이브러리

❖ JSTL 라이브러리

- JSP의 기본 기능이 아님
- 별도의 라이브러리 설치 필요
 - `jstl.jar`, `standar.jar`
- 주요 기능
 - 간단한 로직 구현
 - 다른 JSP 페이지 호출
 - 날짜, 시간, 숫자의 포맷
 - 데이터베이스로의 입력, 수정, 삭제, 조회
 - XML 문서의 처리
 - 문자열 처리 함수 호출

JSTL 라이브러리

❖ JSTL 라이브러리

- 커스텀 태그
 - core, format, xml, sql, functions

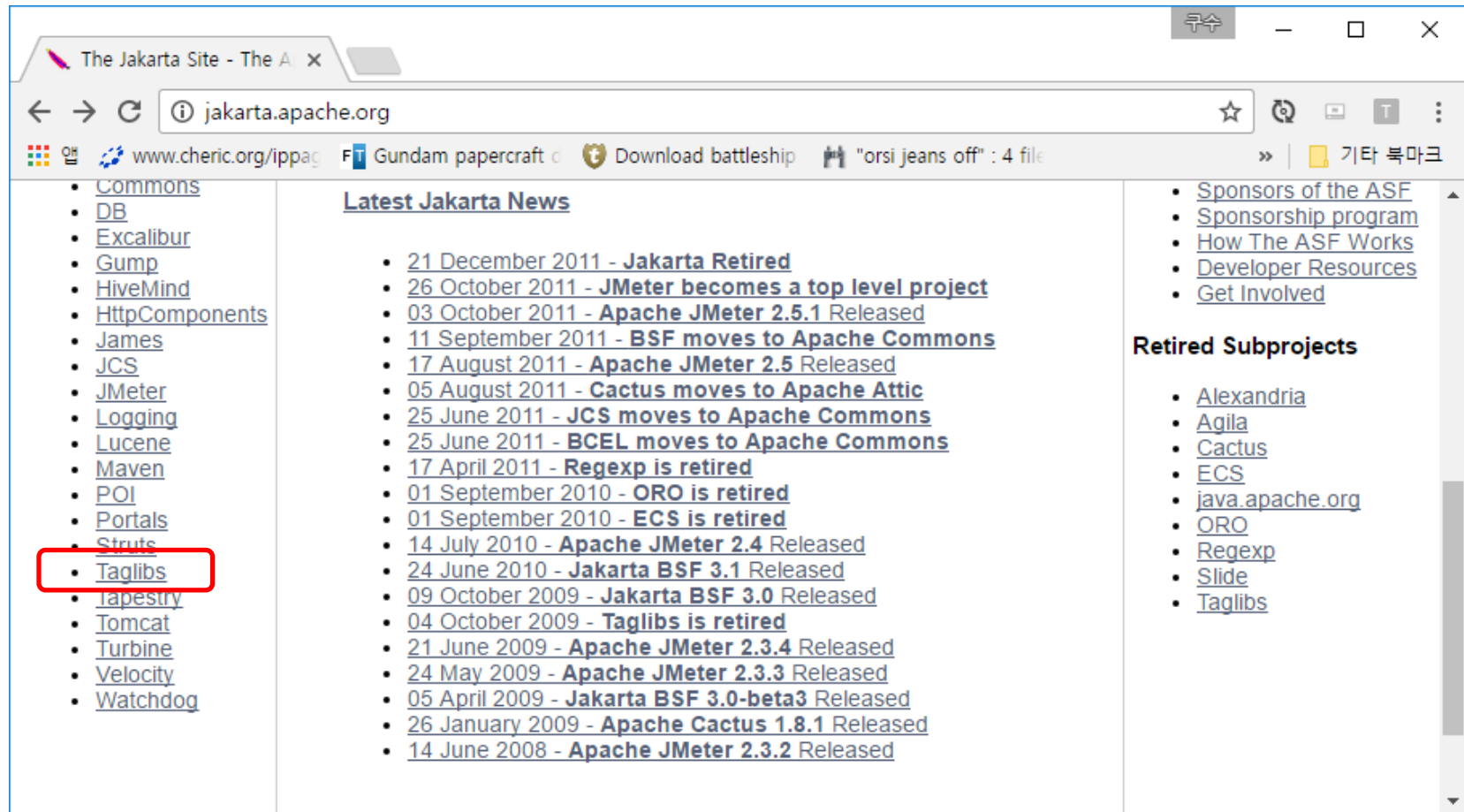
커스텀 태그	설명
기본 기능(core)	일반 프로그램이 언어에서 제공하는 것과 유사한 변수 선언, 실행 흐름의 제어 기능을 제공하고, 다른 JSP 페이지로 제어를 이동하는 기능도 제공한다.
형식화(format)	숫자, 날짜, 시간을 포매팅하는 기능과 국제화, 다국어 지원 기능을 제공한다.
데이터베이스(sql)	데이터베이스의 데이터를 입력/수정/삭제/조회하는 기능을 제공한다.
XML 처리(xml)	XML 문서를 처리할 때 필요한 기능을 제공한다.
함수 처리(functions)	문자열을 처리하는 함수를 제공한다.

JSTL 라이브러리

❖ JSTL 라이브러리

○ JSTL 라이브러리 설치

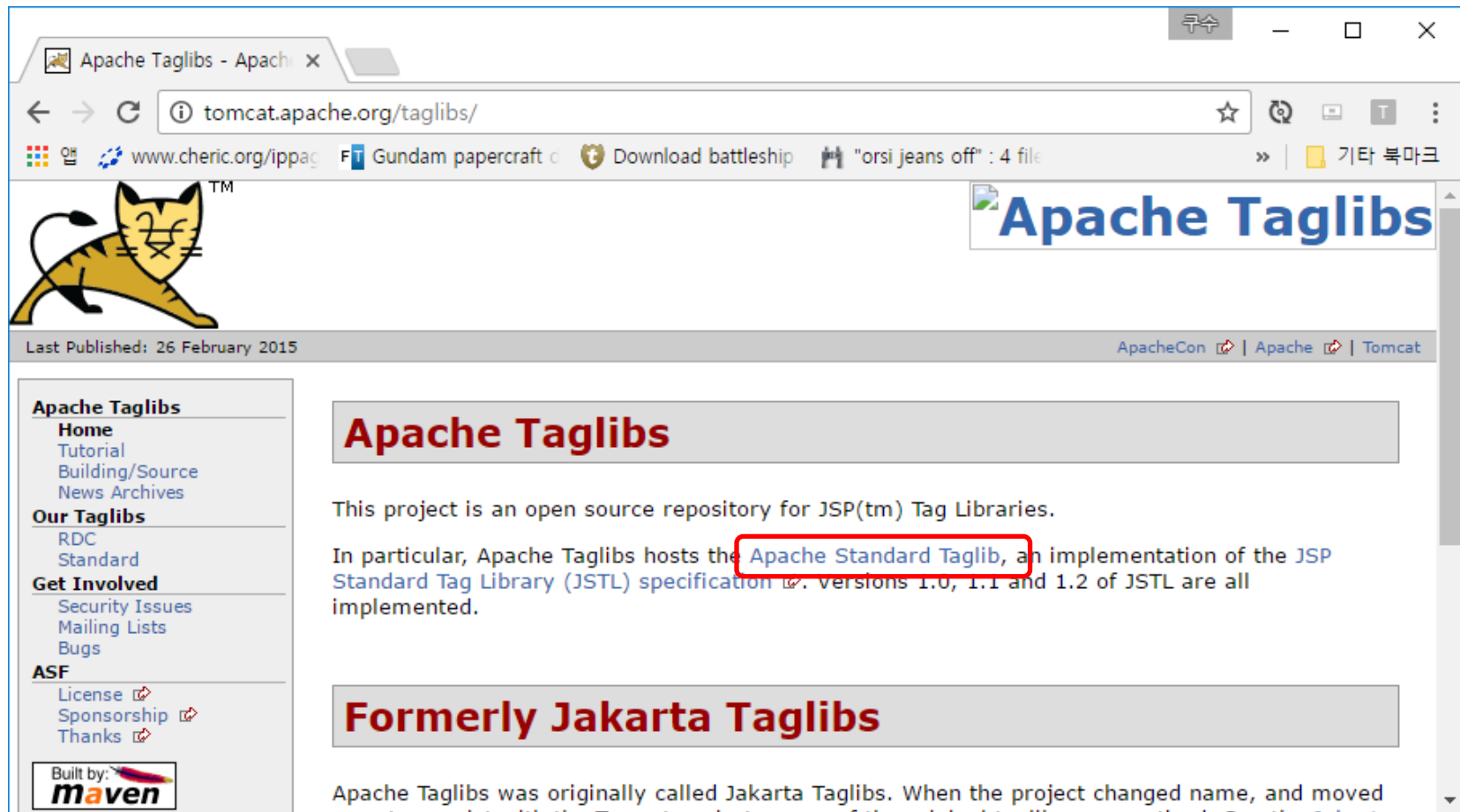
- <http://jakarta.apache.org/> → Taglibs → Apache Standard Taglib



JSTL 라이브러리

❖ JSTL 라이브러리

- JSTL 라이브러리 설치
 - <http://jakarta.apache.org/> → Taglibs → Apache Standard Taglib



JSTL 라이브러리

❖ JSTL 라이브러리

- JSTL 라이브러리 설치



The screenshot shows a web browser window with the URL `tomcat.apache.org/taglibs/standard/`. The page features the Apache Taglibs logo (a cat) and the text "Apache Taglibs". Below the header, there is a sidebar with navigation links and a main content area titled "Standard Taglib". The main content area includes a description of the JSP(tm) Standard Tag Library implementations and a table listing various versions and their requirements. The "download" link for the first row is highlighted with a red box.

Apache Taglibs - Apache

tomcat.apache.org/taglibs/standard/

www.cheric.org/ippag | Gundam papercraft | Download battleship | "orsi jeans off" : 4 file | 기타 북마크

Apache Taglibs

Last Published: 26 February 2015

ApacheCon | Apache | Tomcat

Apache Taglibs

- Home
- Tutorial
- Building/Source
- News Archives

Our Taglibs

- RDC
- Standard

Get Involved

- Security Issues
- Mailing Lists
- Bugs

ASF

- License
- Sponsorship
- Thanks

Built by: **maven**

Standard Taglib

JSP(tm) Standard Tag Library implementations

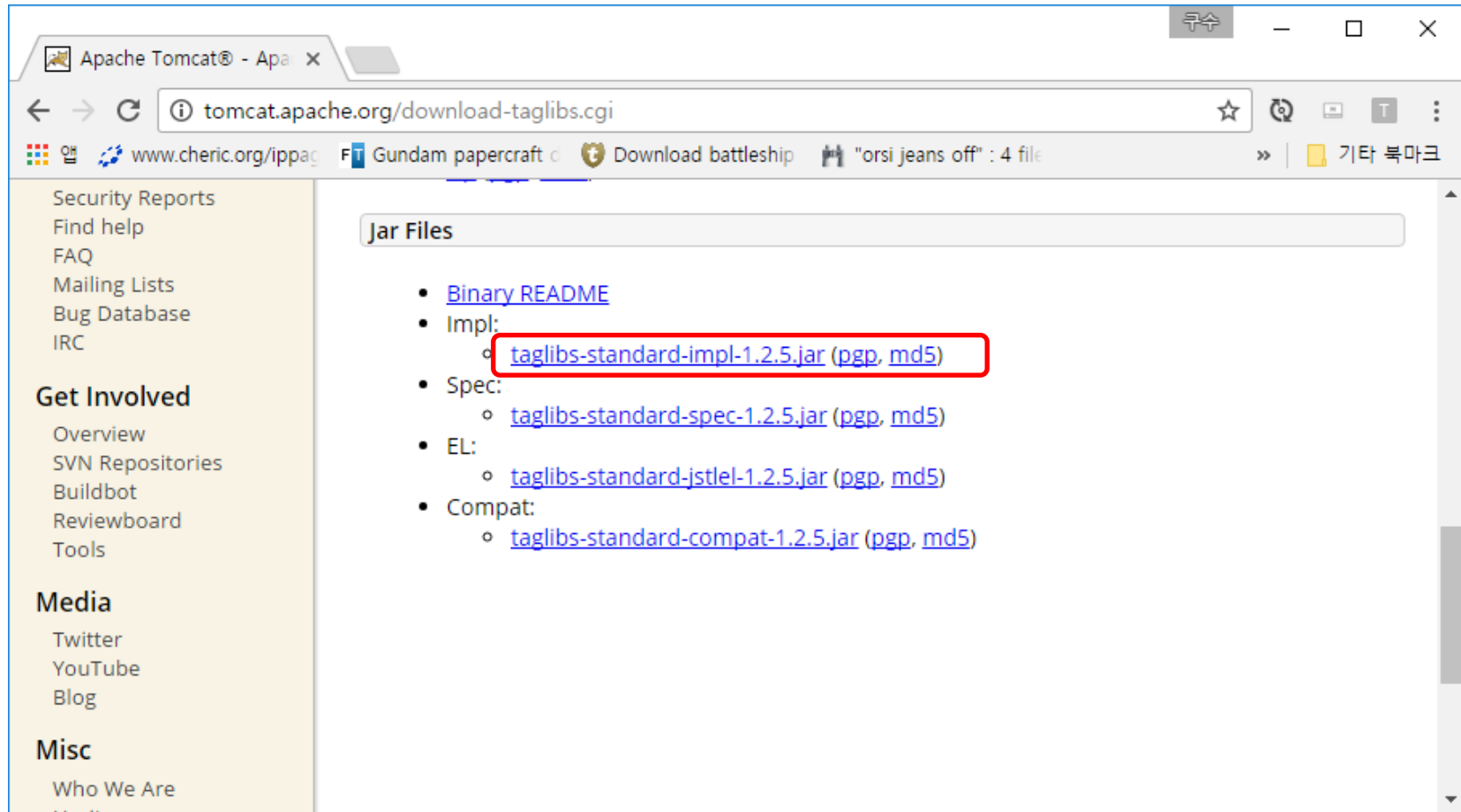
Apache hosts the Apache Standard Taglib, an implementation of the JSP Standard Tag Library (JSTL) specification. Various versions are available.

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

JSTL 라이브러리

❖ JSTL 라이브러리

- JSTL 라이브러리 설치



JSTL 라이브러리

❖ JSTL 라이브러리

- JSTL 라이브러리 설치

- 다운로드받은 jar 파일
 - WEB-INF/lib 폴더에 배치

- JSTL 라이브러리 사용

- 사용할 태그 taglib의 URI 식별자와 사용할 접두사 지정

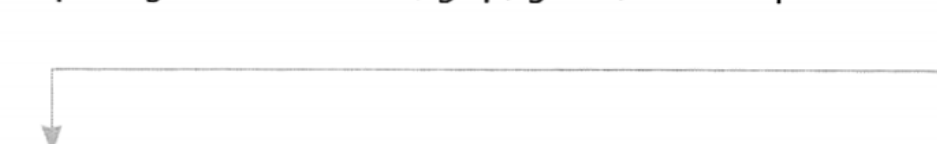
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

사용할 태그 라이브러리 식별자

태그에서 사용할 접두사

- JSTL 태그 사용

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

A line connects the 'c' in 'prefix="c"' of the taglib declaration to the 'c' in 'c:out' of the tag below.

```
<c:out value="Hello World!"/>
```

JSTL 라이브러리

❖ JSTL URI 식별자와 접두사

기능	prefix	기본 URI
기본 기능	c	http://java.sun.com/jsp/jstl/core
형식화	fmt	http://java.sun.com/jstl/fmt
데이터베이스 작업	sql	http://java.sun.com/jstl/sql
XML 처리	x	http://java.sun.com/jstl/xml
함수 처리	fn	http://java.sun.com/jsp/jstl/fn

JSTL 라이브러리

❖ <c:out> 태그

- 간단한 문자열 출력

❖ 10_jstl.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <c:out value="Hello World!" />
</body>
</html>
```

JSTL core 태그

❖ JSTL core 태그

- 속성 제어 및 간단한 로직 처리

태그	설명
<c:set>	변수에 값을 설정한다.
<c:remove>	변수에 설정된 값을 제거한다.
<c:if>	조건에 따라 처리를 달리 할 때 사용한다.
<c:choose>	여러 조건에 따라 처리를 달리 할 때 사용한다.
<c:forEach>	반복 처리를 위해서 사용한다.
<c:forTokens>	구분자로 분리된 각각의 토큰을 처리할 때 사용한다.
<c:import>	외부의 자원을 url을 지정하여 가져다 사용한다.
<c:redirect>	지정한 경로로 이동한다.
<c:url>	url을 재 작성한다.
<c:out>	데이터를 출력할 때 사용하는 태그로 표현식인 <%= %>를 대체할 수 있다.
<c:catch>	예외 처리에 사용한다.

JSTL core 태그

❖ JSTL core 태그

○ <c:set>

- 영역 속성 설정

```
<c:set var="변수 이름" value="저장할 값" [scope="{page|request|session|application}"]>
```

```
pageContext.setAttribute("msg", "Hello");
```

변수 이름 저장할 값



```
<c:set var="msg" value="Hello" scope="page"/>
```

변수 이름

저장할 값

page 영역에 변수 생성

JSTL core 태그

❖ JSTL core 태그

○ <c:set>

- value 속성대신 태그 안에 값 지정 가능

```
<c:set var="변수 이름" [scope="{page|request|session|application}"]>  
    저장할 값  
</c:set>
```

변수 이름

```
<c:set var="age">  
    30 ————— 몸체에 기술한 값이 변수에 저장됨  
</c:set>
```

- scope 생략시 디폴트 값은 page

JSTL core 태그

❖ <jsp:setProperty> 보다 나은 <c:set>

`<c:set target="자바 빈 객체" property="프로퍼티 이름" value="저장할 값">`

`<jsp:setProperty name="member" property="name" value="전수빈"/>`

자바 빈즈 객체

프로퍼티 이름

저장할 값



`<c:set target="${member}" property="name" value="전수빈"/>`

자바 빈즈 객체

프로퍼티 이름

저장할 값

JSTL core 태그

❖ <jsp:setProperty> 보다 나은 <c:set>

- 자바빈의 생성

자바 코드에서 자바 빈 객체 생성

```
com.saeyan.javabeans.MemberBean member = new com.saeyan.javabeans.MemberBean();
```

||

```
<jsp:useBean id="member" class="com.saeyan.javabeans.MemberBean" />
```

액션 태그에서 자바 빈 객체 생성



```
<c:set var="member" value="<%= new com.saeyan.javabeans.MemberBean()  
%>">
```


JSTL core 태그

❖ <jsp:setProperty> 보다 나은 <c:set>

- 자바빈의 생성

```
<c:set target="${member}" property="name" value="전수빈" >
```

```
<c:set target="자바 빈 객체" property="프로퍼티 이름">
```

저장할 값

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

```
<c:set target="${member}" property="userid">
```

pinksubin

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

JSTL core 태그

❖ <jsp:setProperty> 보다 나은 <c:set>

- 간단한 연산 수행

```
<c:set var="add" value="${10 + 5}">
```

```
<c:set var="flag" value="${10 > 5}">
```

❖ 11_jstlCore.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
:
<body>
    <c:set var="msg" value="Hello"></c:set>
    \${msg} = ${msg}<br>
    <c:set var="age">30</c:set>
    \${age} = ${age}<hr>

    <c:set var="member" value=
    "<%= new com.saeyan.javabeans.MemberBean() %>"></c:set>
    <c:set target="\${member}" property="name" value="전수빈" ></c:set>
    <c:set target="\${member}" property="userid">pinksubin</c:set>
    \${member} = ${member}<hr>
    <c:set var="add" value="\${10 + 5}"></c:set>
    \${add} = ${add}<br>
    <c:set var="flag" value="\${10 > 5}"></c:set>
    \${flag} = ${flag}<br>
</body>
</html>
```

JSTL core 태그

❖ <c:remove>

- 영역에서 속성 삭제
- JSP의 `removeAttribute()`와 같은 역할

```
<c:remove var="변수 이름" [scope="{page|request|session|application}"]>
```

```
<c:remove var="age">
```

JSTL core 태그

❖ 흐름을 제어하는 태그

```
<%
if(request.getParameter("color").
equals("1")){
%>
    <span style="color: red;">빨강</span>
<%
}else if(request.getParameter("color").
equals("2")){
%>
    <span style="color: green;">초록</span>
<%
}else if(request.getParameter("color").
equals("3")){
%>
    <span style="color: blue;">파랑</span>
<%
}
%>
```

```
<c:if test="${param.color == 1}">
    <span style="color: red;">빨강</span>
</c:if>
<c:if test="${param.color == 2}">
    <span style="color: green;">초록</span>
</c:if>
<c:if test="${param.color == 3}">
    <span style="color: blue;">파랑</span>
</c:if>
```

JSTL core 태그

❖ <c:if>

- 자바의 if문과 비슷한 기능 제공
 - else 태그 없음
- 형식

```
<c:if test="조건식">
```

조건이 참일 경우 실행할 문장

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

파라미터 color가 1이라는 조건 제시

```
<c:if test="$ {param.color == 1 }">
```

```
  <span style="color: red;">빨강</span>
```

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

조건에 만족할 경우에만 실행됨

JSTL core 태그

❖ <c:if>

■ JSTL을 사용하지 않았을 경우

```
<%  
String str=request.  
getParameter("color");  
int color=Integer.parseInt(str);  
if(color==1){  
%>  
    <span style="color: red;">빨강</span>  
%>  
}
```

■ JSTL을 사용할 경우

```
<c:if test="${param.color == 1}">  
    <span style="color: red">빨강</span>  
</c:if>
```

❖ 12_colorSelectForm.jsp

```
<form action="12_colorSelect.jsp">
  <label for="color">색상을 선택하세요.</label><br>
  <select id="color" name="color" >
    <option value="1">빨강</option>
    <option value="2">초록</option>
    <option value="3">파랑</option>
  </select>
  <input type="submit" value="전송">
</form>
</body>
</html>
```


❖ 12_colorSelect.jsp

```
<body>
  <c:if test="${param.color == 1}">
    <span style="color: red;">빨강</span>
  </c:if>
  <c:if test="${param.color == 2}">
    <span style="color: green;">초록</span>
  </c:if>
  <c:if test="${param.color == 3}">
    <span style="color: blue;">파랑</span>
  </c:if>
</body>
</html>
```

JSTL core 태그

❖ <c:choose>

- 다중 조건

- <c:if>가 else 태그를 지원하지 않는 점 보완

- 기본 형식

<c:choose>

<c:when test="조건1"> 몸체1 </c:when> <!-- 조건1에 만족할 때 -->

<c:when test="조건2"> 몸체2 </c:when> <!-- 조건2에 만족할 때 -->

<c:otherwise> 몸체3 </c:otherwise> <!-- 조건에 만족하지 않을 때 -->

</c:choose>

- 사용 예

<c:choose>

<c:when test="\$ {param.userType == 'admin'}">

 \${param.id}(관리자)

</c:when>

<c:otherwise>—————<c:otherwise>는 위에 제시한

 \${param.id}(회원) <c:when>에 만족하지 않을 때 실행

</c:otherwise>

</c:choose>

❖ 13_fruitSelectForm.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
:
<body>
    <form action="13_fruitSelect.jsp">
        <label for="fruit">과일을 선택하세요</label><br>
        <select id="fruit" name="fruit">
            <option value="1">사과</option>
            <option value="2">메론</option>
            <option value="3">바나나</option>
        </select>
        <input type="submit" value="전송">
    </form>
</body>
</html>
```

❖ 13_fruitSelect.jsp

```
<body>
  <c:choose>
    <c:when test="${param.fruit == 1}">
      <span style="color: red;">사과</span>
    </c:when>
    <c:when test="${param.fruit == 2}">
      <span style="color: green;">메론</span>
    </c:when>
    <c:when test="${param.fruit == 3}">
      <span style="color: blue;">바나나</span>
    </c:when>
  </c:choose>
</body>
</html>
```

JSTL core 태그

❖ <c:forEach>

- 배열이나 컬렉션 또는 맵 등과 같은 집합체에 저장되어 있는 값들을 순차적으로 처리할 수 있는 태그
- 기본 형식

<c:forEach [var="변수 이름"] items="배열과 같은 집합체"
몸체

</c:forEach>

원소 한 개를 저장할 변수

<c:forEach var="movie" items="{movieList}">

{movie}

</c:forEach>

배열(movieList 속성)의 크기만큼
반복적 수행하면 순서대로 원소 한 개를 출력

❖ 14_movieList.jsp

```
<body>
  <%
    String[] movieList = { "타이타닉", "시네마 천국", "혹성
탈출", "킹콩" };
    pageContext.setAttribute("movieList", movieList);
  %>
  <c:forEach var="movie" items="${movieList}">
    ${movie}<br>
  </c:forEach>
</body>
</html>
```

JSTL core 태그

❖ <c:forEach>

- varStatus 속성
 - 순회 속성 활용

프로퍼티	설명
index	items에 지정한 집합체의 현재 반복 중인 항목의 index를 알려준다. 0부터의 순서가 부여된다.
count	루핑을 돌 때 현재 몇 번째를 반복 중인지 알려준다. 1부터의 순서가 부여된다.

반복 상태 정보를 위한 변수

```
<table border="1">
<:forEach var="movie" items="${movieList}" varStatus="status">
<tr>
  <td> ${status.index} </td> — 현재 반복 중인 항목의 index를 알려 줌
  <td> ${status.count} </td> — 몇 번째 반복 중인지 알려 줌
  <td> ${movie } </td>
</tr>
</:forEach>
</table>
```


❖ 15_movieList.jsp

```
<body>
  <%
    String[] movieList = { "타이타닉", "시네마 천국", "혹성 탈출",
"킹콩" };
    pageContext.setAttribute("movieList", movieList);
  %>
  <table border="1" style="width: 100%; text-align: center;">
    <tr>
      <th>index</th>
      <th>count</th>
      <th>title</th>
    </tr>
    <c:forEach var="movie" items="${movieList}"
      varStatus="status">
      <tr>
        <td>${status.index}</td>
        <td>${status.count}</td>
        <td>${movie}</td>
      </tr>
    </c:forEach>
  </table>
</body>
</html>
```

JSTL core 태그

❖ <c:forEach>

- varStatus 기타 속성

프로퍼티	설명
first	현재 루프가 처음인지 여부를 알려준다. 첫 번째일 경우에는 true를 아니면 false를 리턴한다.
last	현재 루프가 마지막인지 여부를 알려준다. 마지막일 경우에는 true를 아니면 false를 리턴한다.

❖ <c:forEach>

○

```
<c:forEach var="movie" items="${movieList}" varStatus="status">
```

```
  <c:choose>
```

```
    <c:when test="${status.first}">
```

```
      <li style="font-weight: bold; color: red;">${movie}</li>
```

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

```
    <c:otherwise>
```

```
      <li>${movie}</li>
```

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

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

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

```
</ul>
```

```
<c:forEach var="movie" items="${movieList}" varStatus="status">
```

```
  ${movie} <c:if test="${not status.last}"> , </c:if>
```

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

❖ 16_movieList.jsp

```
<body>
  <%
    String[] movieList = { "타이타닉", "시네마 천국", "혹성 탈출",
"킹콩" };
    pageContext.setAttribute("movieList", movieList);
  %>
  <ul>
    <c:forEach var="movie" items="${movieList}"
               varStatus="status">
      <c:choose>
        <c:when test="${status.first }">
          <li style="font-weight: bold; color: red;">
            ${movie}</li>
        </c:when>
        <c:otherwise>
          <li>${movie}</li>
        </c:otherwise>
      </c:choose>
    </c:forEach>
  </ul>
  <c:forEach var="movie" items="${movieList}" varStatus="status">
    ${movie} <c:if test="${not status.last}">, </c:if>
  </c:forEach>
</body>
```

JSTL core 태그

❖ <c:forEach>

- 순회 구간 제어

속성	설명
begin	반복에 사용될 것 중 첫 번째 항목의 Index
end	반복에 사용될 것 중 마지막 항목의 Index

```
<c:forEach [var="변수 이름"] begin="시작 값" end="끝 값" [step="증가치"]>  
    몸체  
</c:forEach>
```

JSTL core 태그

❖ <c:forEach>

- 1부터 10까지 자연수 출력

변수 시작 값 끝 값

| | |

```
<c:forEach var="cnt" begin="1" end="10">
    ${cnt} ——— 1부터 10까지 자연수를 순차적으로 출력함
</c:forEach>
```

- count와 index의 차이

시작 값 시작 값

| |

```
<table>
  <c:forEach var="cnt" begin="7" end="10" varStatus="status">
    <tr>
      <td> ${status.index} </td> — 현재 반복 중인 항목의 index는 7, 8, 9, 10이 됨
      <td> ${status.count} </td> — 몇 번째 반복 중인지 알려 주는 count는 1, 2, 3, 4, 5가 됨
      <td> ${cnt} </td>
    </tr>
  </c:forEach>
</table>
```

JSTL core 태그

❖ <c:forEach>

- count와 index의 차이

증가치
|

```
<table>
  <c:forEach var="cnt" begin="1" end="10" step="2" varStatus="status">
    <tr>
      <td> ${status.index} </td> — 현재 반복 중인 항목의 index는 2, 4, 6, 8 10이 됨
      <td> ${status.count} </td> — 몇 번째 반복 중인지 알려 주는 count는 1, 2, 3, 4, 5가 됨
      <td> ${cnt} </td>
    </tr>
  </c:forEach>
</table>
```

❖ 17_movieList.jsp

```
<body>
  <c:forEach var="cnt" begin="1" end="10" varStatus="status">
    ${cnt} <c:if test="${not status.last }">, </c:if>
  </c:forEach>
  <br><br>
  <table border="1" style="width: 100%; text-align: center;">
    <tr>
      <th>index</th>
      <th>count</th>
      <th>cnt</th>
    </tr>
    <c:forEach var="cnt" begin="7" end="10" varStatus="status">
      <tr>
        <td>${status.index}</td>
        <td>${status.count}</td>
        <td>${cnt}</td>
      </tr>
    </c:forEach>
  </table>
  <br><br>
```


❖ 17_movieList.jsp

```
<table border="1" style="width: 100%; text-align: center;">
  <tr>
    <th>index</th>
    <th>count</th>
    <th>cnt</th>
  </tr>
  <c:forEach var="cnt" begin="1" end="10" step="2"
             varStatus="status">
    <tr>
      <td>${status.index}</td>
      <td>${status.count}</td>
      <td>${cnt}</td>
    </tr>
  </c:forEach>
</table>
</body>
</html>
```

JSTL core 태그

❖ 예제

- paramValues의 처리

파일 이름	설명
17_checkbox.jsp	체크 박스가 있는 폼 양식을 갖는다. [전송] 버튼을 클릭하면 JSP로 다중 선택된 값이 전송된다.
17_paramValues.jsp	HTML 문서에서 선택된 체크 박스 값을 처리하는 JSP이다.

❖ 17_checkbox.jsp

```
<h2>악세사리</h2>
관심항목을 선택하세요.
<hr>
<form method="get" action="17_paramValues.jsp">
  <input type="checkbox" name="item" value="신발"> 신발
  <input type="checkbox" name="item" value="가방"> 가방
  <input type="checkbox" name="item" value="벨트"> 벨트<br>

  <input type="checkbox" name="item" value="모자"> 모자
  <input type="checkbox" name="item" value="시계"> 시계
  <input type="checkbox" name="item" value="주얼리"> 주얼리<br>

  <input type="submit" value="전송">
</form>
```

❖ 17_paramValues.jsp

```
<body>
    당신이 선택한 항목입니다.
    <hr>
    <c:forEach var="item" items="${paramValues.item}"
        varStatus="status">
        ${item} <c:if test="${not status.last}">, </c:if>
    </c:forEach>
</body>
</html>
```

JSTL core 태그

❖ <c:forTokens>

- StringTokenizer와 같이 문자열을 구분자로 분리해서 하나씩 추출

```
<c:forTokens var="토큰을 저장할 변수" items="토큰으로 나눌 문자열" delims="구분자">  
    몸체  
</c:forEach>
```

❖ 18_forTokens.jsp

```
<body>
  <c:forTokens var="city" items="서울.인천,대구.부산" delims=", ">
    ${city} <br>
  </c:forTokens>
  <hr>
  <c:forTokens var="city" items="서울.인천,대구.부산" delims=",".>
    ${city} <br>
  </c:forTokens>
</body>
</html>
```

JSTL core 태그

❖ <c:import>

- <jsp:include> 처럼 다른 페이지의 내용을 포함
- 처리 결과를 변수에 저장 가능

```
<c:import url="URL" [var="변수 이름"] [scope="영역"] [charEncoding="charEncoding"]>  
</c:import>
```

- url 속성에 지정된 서버에 접속해서 데이터를 읽어와서 var 속성에 지정한 변수에 저장
- var 속성을 생략한 경우 현재 위치에 결과를 출력

❖ 19_jstlUrl.jsp

```
<body>
    <c:import
        url="http://localhost:8181/web-study-07/02_e1.jsp"
        var="data"></c:import>
    ${data}

</body>
</html>
```


JSTL core 태그

❖ <c:url>

- url 구성
- 한글 인코딩 자동 처리
- 컨텍스트 패스에 맞게 자동 구성

```
<c:url value="URL" [var="변수 이름"] [scope="영역"]>  
</c:url>
```

- url 구성 결과를 var 속성으로 지정한 변수에 저장
- var 속성 생략시 현재 위치에 출력

❖ 20_jstlUrl.jsp

```
<body>
    <c:url value="images/pic.jpg" var="data"></c:url>
    <h3>${data}</h3>
    
</body>
</html>
```

JSTL core 태그

❖ <c:redirect>

- response.sendRedirect() 메서드와 동일한 기능 제공

```
<c:redirect url="URL" [context="경로명"]>
```

❖ 21_jstlUrl.jsp

```
<body>  
    <c:redirect url="20_jstlUrl.jsp"></c:redirect>  
</body>  
</html>
```

JSTL core 태그

❖ <c:out>

- o value 속성에 지정한 문자열 혹은 변수의 내용을 출력

```
<c:out value="value" [default="기본값"] >
```

```
<c:out value="value" >
```

기본 값

```
</c:out>
```

속성	설명
value	출력할 값을 지정한다.
default	지정한 값이 없을 경우 사용할 값을 지정한다.

```
<c:out value="${age}" default="10">
```

```
<c:out value="${age}">
```

10

```
</c:out>
```

JSTL core 태그

❖ <c:catch>

- 예외 처리

```
<c:catch var="변수 이름">
```

예외가 발생할 수 있는 코드

```
</c:catch>
```

- var 속성 지정시 예외의 내용이 해당 변수에 저장

```
<c:catch var="errmsg">
```

예외 발생 전

```
<%=1/0 %>
```

예외 발생 후

```
</c:catch>
```

❖ 22_jstlUrl.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
:
<body>
    <c:set var="age" value="30" scope="page"></c:set>
    나이:<c:out value="${age}">10</c:out>
    <br>
    <c:remove var="age" scope="page"></c:remove>
    나이:<c:out value="${age}">10</c:out>
    <br>
    <c:catch var="errmsg">
        예외 발생 전<br>
        <%=1 / 0%><br>
        예외 발생 후<br>
    </c:catch>
    <c:out value="${errmsg}">></c:out>
</body>
</html>
```

JSTL fmt

❖ JSTL fmt

- 국제화 지역화, 다양한 언어 지원, 날짜와 숫자 형식 처리

기능	태그	설명
숫자 날짜 형식	formatNumber	숫자를 양식에 맞춰서 출력한다.
	formatDate	날짜 정보를 담고 있는 객체를 포매팅하여 출력할 때 사용한다.
	parseDate	문자열을 날짜로 파싱한다.
	parseNumber	문자열을 수치로 파싱한다.
	setTimeZone	시간대별로 시간을 처리할 수 있는 기능을 제공한다.
	timeZone	시간대별로 시간을 처리할 수 있는 기능을 제공한다.
로게일 지정	setLocale	국제화 태그들이 사용할 로케일을 지정한다.
	requestEncoding	요청 파라미터의 인코딩을 지정한다.
메시지 처리	bundle	태그 몸체에서 사용할 리소스 번들을 지정한다.
	message(param)	메시지를 출력한다.
	setBundle	특정 리소스 번들을 사용할 수 있도록 로딩한다.

JSTL fmt

❖ JSTL fmt

- taglib 지시자

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

❖ 숫자 날짜 형식 지정 관련 태그

- `formatNumber`
- `formatDate`
- `parseDate`
- `parseNumber`
- `setTimeZone`
- `timeZone`

❖ <fmt:formatNumber>

- 원하는 패턴대로 수치 데이터를 표현

```
<fmt:formatNumber value="수치 데이터"  
    [type="{number|currency|percent}"]  
    [pattern="패턴"]  
    [currencySymbol="화폐 단위"]  
    [groupingUsed="{true|false}"]  
    [var="변수 이름"]  
    [scope="{page|request|session|application}"]>
```

❖ <fmt:formatNumber>

속성	표현식	타입	설명
value	true	String 또는 Number	형식화할 수치 데이터
type	true	String	숫자(number), 통화(currency), 퍼센트(percent) 중 어느 형식으로 표시할 지 지정
pattern	true	String	사용자가 지정한 형식 패턴
currencySymbol	true	String	통화 기호. 통화 형식(type="currency")일 때만 적용
groupingUsed	true	boolean	콤마와 같이 단위를 구분할 때 사용하는 기호를 표시할지의 여부를 결정한다. true이면 10,000과 같이 구분 기호가 사용되며 false이면 10000로 출력된다. 기본값은 true이다.
var	false	String	형식 출력 결과 문자열을 담는 scope에 해당하는 변수 이름
scope	false	String	var 속성에 지정한 변수가 효력을 발휘할 수 있는 영역 지정

❖ <fmt:formatNumber>

- 디폴트로 수치 데이터를 천단위로 콤마 출력

```
<fmt:formatNumber value="1234567.89"/>
```

1,234,567.89 — value 속성에 지정한 수치 데이터를
3자리마다 콤마로 구분해서 출력

구분 기호를 표시하지 않기 위해 false를 지정

```
<fmt:formatNumber value="1234567.89" groupingUsed="false"/>
```

1234567.89 — value 속성에 지정한 수치 데이터를
구분 기호 없이 출력

JSTL fmt

❖ <fmt:formatNumber>

○ 퍼센트 출력

퍼센트 형식 지정

`<fmt:formatNumber value="0.5" type="percent"/>`

50% — value 속성에 지정한 수치 데이터 0.5를
퍼센트 형식인 50%로 변환하여 출력

○ 통화 기호 출력

통화 형식 지정

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

₩10,000 — value 속성에 지정한 수치 데이터를 세 자리마다
콤마로 구분해서 출력되고 화폐 단위를 표시함

JSTL fmt

❖ <fmt:formatNumber>

- 통화 기호 지정

통화 기호 지정

`<fmt:formatNumber value="10000" type="currency" currencySymbol="$
$10,000 — 통화 기호가 변경되어 표시됨`

❖ <fmt:formatNumber>

- 출력 패턴 지정
 - #, 0과 소수점을 위한 닷으로 지정
 - # : 채워야할 자리에 비해서 데이터가 모자라는 경우 공백 표시
 - 0 : 채워야할 자리에 비해서 데이터가 모자라는 경우 0 표시
 - # : 소수점 자리수보다 수치 데이터 길이가 길 경우 자릿수 만큼만 출력

소수점 이하 2자리까지 표시하는 패턴을 지정

```
<fmt:formatNumber value="1234567.8912345" pattern="#,#00.0#"/>
```

1,234,567.89 — value 속성에 지정한 수치 데이터를 소수점에 맞추어 소수점 이하 2자리까지 출력하기 위해서 나머지는 잘라냄

❖ <fmt:formatNumber>

○ 출력 패턴 지정

- # : 소수점 자리수 보다 수치 데이터가 모자라는 경우 공백 표시

소수점 이하 2자리까지 표시하는 패턴을 지정

```
<fmt:formatNumber value="1234567.8" pattern="#.##0.0#"/>
```

1,234,567.8 — value 속성에 지정한 수치 데이터를 소수점에 맞추어 소수점 이하 2자리까지 출력하되 빈 자리는 공백으로 표시

- 0 : 모자라는 자리에 0 출력

소수점 이하 3자리까지 표시하는 패턴을 지정

```
<fmt:formatNumber value="1234567.89" pattern=".###000"/>
```

1234567.890 — value 속성에 지정한 수치 데이터를 소수점에 맞추어 소수점 이하 3자리까지 출력하기 위해서 빈 자리를 0으로 채움

❖ <fmt:formatDate>

- 날짜를 포맷된 형태로 출력

```
<fmt:formatDate value="date"  
    [type="{time|date|both}"]  
    [dateStyle="{default|short|medium|long|full}"]  
    [timeStyle="{default|short|medium|long|full}"]  
    [pattern="customPattern"]  
    [timeZone="timeZone"]  
    [var="varName"]  
    [scope="{page|request|session|application}"]>
```

❖ <fmt:formatDate>

속성	표현식	Type	설명
value	true	java.util.Date	형식화될 Date와 time
type	true	String	형식화할 데이터가 시간(time), 날짜(date), 모두(both) 셋 중 하나를 지정.
dateStyle	true	String	미리 정의된 날짜 형식으로 default, short, medium, long, full 넷 중에 하나를 지정.
timeStyle	true	String	미리 정의된 시간 형식으로 short, medium, long, full 넷 중에 하나를 지정.
pattern	true	String	사용자 지정 형식 스타일
timeZone	true	String 또는 java.util.TimeZone	형식화 시간에 나타날 타임존
var	false	String	형식 출력 결과 문자열을 담는 scope에 해당하는 변수 이름
scope	false	String	var의 scope

❖ <fmt:formatDate>

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

현재 시간 정보를 포함한 날짜 객체 생성

`${now}` — 현재 시간 정보를 포함한
날짜 객체 정보를 출력한다.

Thu Sep 26 04:02:09 KST 2013 — 출력 결과

날짜 객체

```
<fmt:formatDate value="$ {now}" />
```

2013. 9. 26 — yyyy. MM. dd 형태로 출력됨

JSTL fmt

❖ <fmt:formatDate>

- 시간만 출력하는 경우

시간(time) 형태로 출력할 형식을 지정

time: <fmt:formatDate value="\$ {now}" type="time"/>

오전 4:38:26 — 오전 hh:MM:ss 형태로 출력됨

- 날짜와 시간 모두 출력

날짜와 시간을 모두 출력하는 형식을 지정

time: <fmt:formatDate value="\$ {now}" type="both"/>

2013. 9. 26 오전 4:38:26 — yyyy. MM. dd 오전 hh:MM:ss 형태로 출력됨

JSTL fmt

❖ <fmt:formatDate>

- 출력 패턴 지정하기

원하는 포맷을 지정

```
<fmt:formatDate value="$ {now}" pattern="yyyy년 MM월 dd일 hh시 mm분 ss초"/>
```

2013년 09월 26일 04시 38분 26초 ————— 지정한 포맷 형태로 출력됨

❖ 26_jstlFmt.jsp

```
<pre>
<c:set var="now" value="<%=new java.util.Date()%>"></c:set>
\${now} : ${now}
<fmt:formatDate value="\${now}"></fmt:formatDate>
```

❖ 26_jstlFmt.jsp

```
date : <fmt:formatDate value="${now}" type="date"></fmt:formatDate>
time : <fmt:formatDate value="${now}" type="time"></fmt:formatDate>
both : <fmt:formatDate value="${now}" type="both"></fmt:formatDate>
default : <fmt:formatDate value="${now}" type="both"
dateStyle="default"
        timeStyle="default"></fmt:formatDate>
short : <fmt:formatDate value="${now}" type="both" dateStyle="short"
        timeStyle="short"></fmt:formatDate>
medium : <fmt:formatDate value="${now}" type="both" dateStyle="medium"
        timeStyle="medium"></fmt:formatDate>
long : <fmt:formatDate value="${now}" type="both" dateStyle="long"
        timeStyle="long"></fmt:formatDate>
full : <fmt:formatDate value="${now}" type="both" dateStyle="full"
        timeStyle="full"></fmt:formatDate>
```


❖ 26_jstlFmt.jsp

```
pattern="yyyy년 MM월 dd일 hh시 mm분 ss초" :  
<fmt:formatDate value="${now}"  
    pattern="yyyy년 MM월 dd일 hh시 mm분 ss초"></fmt:formatDate>  
</pre>  
</body>  
</html>
```

JSTL fmt

❖ <fmt:setTimeZone>, <fmt:timeZone>

- 특정 지역의 타임존을 설정하는 태그

```
<fmt:setTimeZone value="timeZone"  
    [var="varName"]  
    [scope="{page|request|session|application}"]>
```

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

 몸체

```
</fmt:timeZone>
```

❖ 27_jstlFmt.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
:
<body>
    <jsp:useBean id="now" class="java.util.Date"></jsp:useBean>

    <pre>
default: <c:out value="${now}"></c:out>

Korea KST : <fmt:formatDate value="${now}" type="both" dateStyle="full"
            timeStyle="full"></fmt:formatDate>
<fmt:timeZone value="GMT">

Swiss GMT : <fmt:formatDate value="${now}" type="both" dateStyle="full"
            timeStyle="full"></fmt:formatDate>

            </fmt:timeZone>
```

❖ 27_jstlFmt.jsp

```
<fmt:timeZone value="GMT-8">
NewYork GMT-8: <fmt:formatDate value="${now}" type="both"
                dateStyle="full" timeStyle="full"></fmt:formatDate>

                </fmt:timeZone>
</pre>
</body>
</html>
```

❖ 로케일 지정을 위한 태그

- `<fmt:setLocale>`
 - 해당 지역의 통화기호, 날짜 포맷 등을 적용

```
<fmt:setLocale value="locale">
```

❖ 28_jstlFmt.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
:
<body>
    <c:set var="now" value="<%=new java.util.Date()%>"></c:set>
    <pre>
톰캣 서버의 기본 로케일 : <%=response.getLocale()%>

<fmt:setLocale value="ko_kr"></fmt:setLocale>
로케일을 한국어로 설정후 로케일 확인 : <%=response.getLocale()%>

통화(currency) : <fmt:formatNumber value="10000"
type="currency"></fmt:formatNumber>
날짜 : <fmt:formatDate value="${now}"></fmt:formatDate>
```

❖ 28_jstlFmt.jsp

```
<fmt:setLocale value="ja_JP"></fmt:setLocale>
로케일을 일본어로 설정후 로케일 확인 : <%=response.getLocale()%>

통화(currency) : <fmt:formatNumber value="10000"
type="currency"></fmt:formatNumber>
날짜 : <fmt:formatDate value="${now}"></fmt:formatDate>

<fmt:setLocale value="en_US"></fmt:setLocale>
로케일을 영어로 설정후 로케일 확인 : <%=response.getLocale()%>

통화(currency) : <fmt:formatNumber value="10000"
type="currency"></fmt:formatNumber>
날짜 : <fmt:formatDate value="${now}"></fmt:formatDate>

</pre>
</body>
</html>
```

JSTL fmt

❖ <fmt:requestEncoding>

- `request.setCharacterEncoding()` 메서드와 동일