JSTL

524730-1
2021년 봄학기
4/28/2021
박경신

JSTL

- JSTL (JSP Standard Tag Library)
  - JCP(Java Community Process, 자바표준화 단체) 에서 정한 표준
  - JSP 페이지에서 스크립트 요소로 인한 코드의 복잡함을 해결하기 위한 일종의 사용자 정의 태그
  - **널리 사용되는 사용자 정의 태그를 표준으로 만든 라이브러리**
  - JSP 페이지의 로직을 담당하는 부분인 if, for, while, 데이터베이스 처리 등과 관련 된 코드를 JSTL로 대치하여 코드를 깔끔하게 하고 가독성을 좋게 하는 것
  - JSP 내장 객체에 쉽게 접근할 수 있을 뿐 아니라 파라미터, 헤더, 쿠키 등의 복잡한 코드를 사용하지 않고 쉽게 직관적으로 사용
  - 객체 간의 비교를 equals() 메소드로 처리하는 대신 ==와 같은 간단한 연산자로 구현이 가능

JSTL 구성

- JSTL 구성
  - JSTL CORE Tags : 변수 선언 및 삭제, if, for, URI 처리 기능
  - JSTL Formatting Tags : 텍스트, 날짜, 숫자를 형식화하는 기능
  - JSTL SQL : JDBC를 이용한 DB처리 기능
  - JSTL XML : XML문서 처리 관련 기능
  - JSTL Functions : Collection, String 처리 관련 기능

Library	Syntax
CORE	<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
Formatting	<%@ taglib prefix="fmt" uri="http://java.sun.com/jstl/fmt" %>
SQL	<%@ taglib prefix="sql" uri="http://java.sun.com/jstl/sql" %>
XML	<%@ taglib prefix="x" uri="http://java.sun.com/jstl/xml" %>
Functions	<%@ taglib prefix="fn" uri="http://java.sun.com/jstl/functions" %>

JSTL 제공하는 태그의 종류와 사용법

- JSTL 제공하는 태그 사용
  - **JSTL 라이브러리인 jstl.jar 파일이 필요**
    - https://mvnrepository.com/artifact/javax.servlet/jstl/1.2 파일 다운로드
    - WEB-INF/lib에 복사
  - **JSP 페이지에 <%@ taglib> 지시어를 사용**
    - prefix 속성은 uri 속성에 명시된 값 대신 해당 페이지에서 prefix 속성 값으로 명시된 값을 사용하겠다는 것을 의미

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

JSTL 사용 예시

- JSTL <c:out>를 사용하여 간단한 출력

```
<%-- 스크립트릿으로 구현시 --%>
<%= pageContext.getAttribute("member").getName() %>

<%-- JSTL 사용시 --%>
<c:out value="${member.name}"/>
```

JSTL Core 태그

Core 태크	설명
<c:out>	출력에 사용
<c:set>	사용할 변수를 설정하는데 사용
<c:remove>	설정한 변수를 제거하는데 사용
<c:catch>	예외 처리에 사용
<c:if>	조건문을 처리하는데 사용
<c:choose>	다중 조건문을 처리하는데 사용
<c:when>	<choose>의 서브 태크로 조건문이 참일때 수행
<c:otherwise>	<choose>의 서브 태크로 조건문이 거짓일때 수행
<c:import>	URL을 사용하여 다른 리소스의 결과를 삽입하는데 사용
<c:forEach>	반복문을 처리하는데 사용
<c:forTokens>	구분자로 분리된 각각의 토큰을 처리하는데 사용
<c:param>	URL 관련 태크의 파라미터를 설정하는데 사용
<c:redirect>	설정한 경로로 이동하는데 사용
<c:url>	URL을 재작성하는데 사용

변수 지원

- 변수 설정
  - <c:set> **EL 변수 값** 설정 (생성 또는 변경)

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

  - <c:set> 객체의 멤버변수값 설정

```
<c:set target="대상" property="프로퍼티이름" value="값" />
<c:set target="대상" property="프로퍼티이름">값</c:set>
```

속성	표현식/EL	타입	설명
value	불가	String	저장할 변수값
target	가능	Object	값이 저장할 객체명
property	가능	String	target 객체의 멤버 변수명
var	불가	String	값이 저장될 변수명
scope	불가	String	변수 생성될 영역(page, session, request, application)

변수 지정

- 변수 설정
  - <c:set> **EL 변수 값** 설정 (생성 또는 변경)

```
<c:set var="browser" value="${header['User-Agent']}" />
<c:out value="${browser}" />
```

  - <c:set> 객체의 멤버변수값 설정

```
class Person { // PersonBean
  private String name; private int age;
  public void setName(String name) {    this.name = name;   }
  public String getName() {    return name;   }
  public void setAge(int age) {    this.age = age;   }
  public int getAge() {    return age;   }
}
  <jsp:useBean id="person" class="dto.Person" />
  <c:set target="${person}" property="name" value="Park" />
  <c:set target="${person}" property="age" value="10" />
  <c:out value="${person.name}" />
  <c:out value="${person.age}" />
```

변수 지정

- 변수 삭제
  - <c:remove>
    - scope 미지정시 모든 영역의 변수 삭제

```
<c:remove var="varName" [scope="영역"] />
```

속성	표현식/EL	타입	설명
var	사용불가	String	삭제할 변수 이름
scope	사용불가	String	삭제할 변수가 포함된 영역(page, session, request, application)

흐름 제어

- if - 조건이 true일 경우 몸체 내용 실행

```
<c:if test="조건">
...
</c:if>
```

- choose - when - otherwise
  - switch - case - default와 동일

```
<c:choose>
  <c:when test="${member.level == 'trial'}" >
    ...
  </c:when>
  <c:when test="${member.level == 'regular'}" >
    ...
  </c:when>
  <c:otherwise>
    ...
  </c:otherwise>
</c:choose>
```

반복 처리

- forEach
  - 집합이나 콜렉션 데이터 사용

```
<c:forEach var="변수" items="아이템">
  … ${변수} ...
</c:forEach>
```

  - 특정 회수 반복

```
<c:forEach var="i" begin="1" end="10" [step="값"]>
  ${i} 사용
</c:forEach>
```

  - varStatus 속성

```
<c:forEach var="item" items="<%= someItemList %>" varStatus="status">
${status.index + 1} 번째 항목 : ${item.name}
</c:forEach>
```

index - 루프 실행에서 현재 인덱스, count - 루프 실행 회수
begin - begin 속성 값, end - end 속성 값, step - step 속성 값
first - 현재 실행이 첫 번째 실행인 경우 true
last - 현재 실행이 루프의 마지막 실행인 경우 true
current - 콜렉션 중 현재 루프에서 사용할 객체

반복 처리

- forTokens
  - 구분자로 분리된 각각의 토큰을 처리에 사용

```
<c:set var="rainbow" value="red,orange,yellow,green,blue,indigo,violet." />
<c:forTokens var="color" items="${rainbow}" delims=",.">
<c:out value="${color}" /> &amp;nbsp;&amp;nbsp;
</c:forTokens>
```

URL 관련 태그

- import - 외부/내부 페이지를 현재 위치에 삽입

```
<c:import url="URL" [var="변수명"] [scope="영역"] [charEncoding="캐릭터셋"]>
   <c:param name="파라미터이름" value="값" />
   ...
</c:import>
```

  - 상대 URL import 시 <jsp:include>와 동일하게 동작

- url - 절대 URL과 상대 URL을 알맞게 생성

```
<c:url value="URL" [var="varName"] [scope="영역"]>
   <c:param name="이름" value="값" />
</c:url>
```

  - 웹 컨텍스트 내에서 절대 경로 사용시 컨텍스트 경로 자동 추가

- redirect - 지정한 페이지로 리다이렉트

```
<c:redirect url="URL" [context="콘텍스트경로"]>
   <c:param name="이름" value="값" />
</c:redirect>
```

출력

- <c:out> 데이터를 출력
  - escapeXml 속성이 true일 경우 다음과 같이 특수 문자 처리
    - < → < , > → >
    - & → & , ' → ' , " → "

```
<c:out value="value" [escapeXml="(true|false)"] [default="defaultValue"] />
<c:out value="value" [escapeXml="(true|false)"]>
    default value
</c:out>
```

속성	필수	기본값	설명
value	Y	없음	출력할 내용이나 표현식
default	N	tag body	value 값에 내용이 없는 경우 출력할 내용으로 태그 바디 혹은 속성값 형태로 올수 있음
excapeXml	N	true	출력할 내용에 <,>,&,',” 등의 문자를 일반 문자로 변환할지 결정. 출력될 내용이 html tag를 포함하면 값을 false로 해야 태그가 반영된 내용이 화면에 보임. 만일 true로 하면 태그가 화면에 그대로 보임

예외처리

- catch - 몸체에서 발생한 예외를 변수에 저장

```
<c:catch var="exName">
...
예외가 발생할 수 있는 코드
...
</c:catch>
${exName} 사용
```

JSTL Formatting 태그

Formatting 태크	설명
<fmt:formatNumber>	숫자 포멧팅
<fmt:parseNumber>	문자열을 분석해서 숫자로 변환
<fmt:formatDate>	Date 객체 포멧팅
<fmt:parseDate>	문자열을 분석해서 Date 로 변환
<fmt:bundle>	사용할 번들을 지정
<fmt:setBundle>	리소스 번들을 읽어와 특정 변수에 저장
<fmt:message>	지역에 알맞은 메시지를 출력
<fmt:setLocale>	Locale 을 지정
<fmt:timeZone>	시간대를 지정
<fmt:setTimeZone>	시간대 정보를 특정 변수에 저장
<fmt:requestEncoding>	Request 파라미터의 캐릭터 인코딩을 지정

로케일 지정 및 요청 파라미터 인코딩 지정

- <fmt:setLocale value="언어코드" scope="범위" />
  - 국제화 태그가 Accept-Language 헤더에서 지정한 언어가 아닌 다른 언어를 사용하도록 지정하는 기능

- <fmt:requestEncoding value="캐릭터셋" />
  - 요청 파라미터의 캐릭터 인코딩을 지정
  - request.setCharacterEncoding("캐릭터셋")과 동일

<fmt:message> 태그

- 리소스 번들 범위에서 메시지 읽기

```
<fmt:bundle basename="resource.message" [prefix="접두어"]>
   <fmt:message key="GREETING" />
</fmt:bundle>
```

- 지정한 번들에서 메시지 읽기

```
<fmt:setBundle var="message" basename="resource.message" />
...
<fmt:message bundle="${message}" key="GREETING" />
```

- <fmt:message> 태그의 메시지 읽는 순서
  - bundle 속성에 지정한 리소스 번들을 사용
  - <fmt:bundle> 태그에 중첩된 경우 <fmt:bundle> 태그에서 설정한 리소스 번들 사용
  - 1과 2가 아닐 경우 기본 리소스 번들 사용. 기본 리소스 번들은 web.xml 파일에서 javax.servlet.jsp.jstl.fmt.localizationContext 콘텍스트 속성을 통해서 설정 가능

formatNumber 태그

- 숫자를 포맷팅

```
<fmt:formatNumber value="숫자값" [type="값타입"] [pattern="패턴"]
    [currentCode="통화코드"] [currencySymbol="통화심볼"]
    [groupingUsed="(true|false)"] [var="변수명"] [scope="영역"] />
```

속성	표현식/EL	타입	설명
value	사용 가능	String 또는 Number	양식에 맞춰 출력할 숫자
type	사용 가능	String	어떤 양식으로 출력할지를 정한다. number는 숫자형식, percent는 % 형식, currency는 통화형식으로 출력. 기본 값은 number.
pattern	사용 가능	String	직접 숫자가 출력되는 양식을 지정한다. DecimalFormat 클래스에서 정의되어 있는 패턴 사용
var	사용 불가	String	포맷팅 한 결과를 저장할 변수 명. var 속성을 사용하지 않으면 결과가 곧바로 출력.
scope	사용 불가	String	변수를 저장할 영역. 기본 값은 page 이다.

parseNumber 태그

- 문자열을 숫자 데이터 타입으로 변환

```
<fmt:parseNumber value="값" [type="값타입"] [pattern="패턴"]
    [parseLocale="통화코드"] [integerOnly="true|false"]
    [var="변수명"] [scope="영역"] />
```

속성	표현식/EL	타입	설명
value	사용 가능	String	파싱할 문자열
type	사용 가능	String	value 속성의 문자열 타입을 지정. number, currency, percentage 가 올 수 있다. 기본 값은 number
pattern	사용 가능	String	직접 파싱할 때 사용할 양식을 지정
var	사용 불가	String	파싱한 결과를 저장할 변수 명을 지정
scope	사용 불가	String	변수를 저장할 영역을 지정한다. 기본 값은 page.

formatDate 태그

- 날짜 정보를 담은 객체(Date)를 포맷팅

```
<fmt:formatDate value="날짜값"
    [type="타입"] [dateStyle="날짜스타일"] [timeStyle="시간스타일"]
    [pattern="패턴"] [timeZone="타임존"]
    [var="변수명"] [scope="영역"] />
```

속성	표현식/EL	타입	설명
value	사용 가능	java.util.Date	포맷팅할 날짜 및 시간 값
type	사용 가능	String	날짜, 시간 또는 둘 다 포맷팅 할 지의 여부를 지정
dateStyle	사용 가능	String	날짜에 대한 포맷팅 스타일을 지정
timeStyle	사용 가능	String	시간에 대한 포맷팅 스타일을 지정
pattern	사용 가능	String	직접 파싱할 때 사용할 양식을 지정
var	사용 불가	String	파싱한 결과를 저장할 변수 명을 지정
scope	사용 불가	String	변수를 저장할 영역을 지정

timeZone과 setTimeZone

- 국제화 태그가 사용할 시간대 설정

```
<fmt:timeZone value="Hongkong">
  <!-- 사용하는 시간을 Hongkong 시간대에 맞춘다. -->
  <fmt:formatDate ... />
</fmt:timeZone>
```

web.xml, 국제화 태그 콘텍스트 속성

속성 이름	설명
javax.servlet.jsp.jstl.fmt.localizationContext	기본으로 사용할 리소드 번들을 지정한다. 리소스 번들의 basename을 입력한다.
javax.servlet.jsp.jstl.fmt.locale	기본으로 사용할 로케일을 지정한다.
javax.servlet.jsp.jstl.fmt.timeZone	기본으로 사용할 시간대를 지정한다.

JSTL SQL 태그

Sql 태크	설명
<sql:setDataSource>	DataSource를 설정하는데 사용
<sql:query>	조회 쿼리문을 실행하는데 사용
<sql:update>	삽입, 수정, 삭제 쿼리문을 실행하는데 사용
<sql:dateParam>	쿼리문에 문자열형식의 파라미터 설정하는데 사용
<sql:param>	쿼리문에 날짜형식의 파라미터 설정하는데 사용
<sql:transaction>	트랜잭션을 구현하는데 사용

JSTL XML 태그

XML 태크	설명
<x:out>	XPath 표현식 <%= >
<x:parse>	XML 데이터 파싱
<x:set>	XPath 표현의 변수 지정
<x:if>	XPath 표현 조건 처리
<x:forEach>	XML 도큐먼트의 노드 for
<x:choose>	다중 조건문을 처리하는데 사용
<x:when>	<choose>의 서브 태크로 조건문이 참일때 수행
<x:otherwise>	<choose>의 서브 태크로 조건문이 거짓일때 수행
<x:transform>	XML 도큐먼트에서 XSL 변환
<x:param>	<transform>의 서브 태크로 파라미터 지정

JSTL XML 태그

```
<c:import var="WOEIDList" url="WOEIDList.xml" />
<x:parse xml = "${WOEIDList}" var = "output"/>
<c:forEach var="i" items="${WOEIDList}">
${i}<br>
</c:forEach>
<p></p>
<x:out select = "$output/ArrayOfWOEID/WOEID[1]/Code" />
<x:out select = "$output/ArrayOfWOEID/WOEID[1]/City" />
<x:out select = "$output/ArrayOfWOEID/WOEID[1]/Country" />
<x:out select = "$output/ArrayOfWOEID/WOEID[1]/Latitude" />
<x:out select = "$output/ArrayOfWOEID/WOEID[1]/Longitude" />
<br>
```

4118 Toronto Canada 43.64856 -79.385368 44418 London United Kingdom 51.507702 -0.12797 2487956 San Francisco United States 37.747398 -122.439217 1132447 Busan South Korea 35.170429 128.999481

4118 Toronto Canada 43.64856 -79.385368

JSTL Function 태그

Function 태크	설명
fn:contains()	검색대상 문자열의 포함 여부를 확인
fn:containsIgnoreCase()	대소문자에 상관없이 검색대상 문자열 포함 여부를 확인
fn:starsWith()	특정 문자열로 시작하는지 여부를 확인
fn:endsWith()	특정 문자열로 끝나는지 여부를 확인
fn:escapeXml()	문자열에 포함된 특수문자를 특정 코드로 변환
fn:indexOf()	검색 대상 문자열의 첫 위치 값을 반환
fn:join()	배열 형태의 문자열을 설정한 구분자로 연결하여 반환
fn:split()	문자열을 설정한 구분자로 분리하여 배열 형태로 반환
fn:length()	문자열의 길이를 반환
fn:replace()	검색대상 문자열을 설정한 문자로 변경하여 반환
fn:substring()	특정 위치의 문자열을 반환
fn:toLowerCase()	모두 소문자로 변환
fn:toUpperCase()	모두 대문자로 변환
fn:trim()	문자열 앞뒤의 공백을 제거하여 반환

JSTL Function 태그

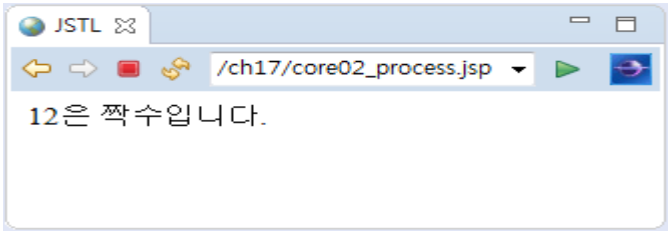
```
<c:set var="rainbow" value="red,orange,yellow,green,blue,indigo,violet." />
rainbow :
<c:forTokens var="color" items="${rainbow}" delims=",.">
<c:out value="${color}" /> &amp;nbsp;&amp;nbsp;
</c:forTokens>
<c:if test="${fn:contains(rainbow,  'blue')}">
<p>rainbow contains blue<p>
</c:if>
<c:set var="rainbow2" value="${fn:split(rainbow,  ',.')}" />
rainbow2 :
<c:forEach var="i" items="${rainbow2}">
${i} &amp;nbsp;&amp;nbsp;
</c:forEach>
<br>
<c:set var="rainbow3" value="${fn:join(rainbow2,  '-')}" />
<c:out value="rainbow3: ${rainbow3}" />
```

```
rainbow: red orange yellow green blue indigo violet
rainbow contains blue
rainbow2: red orange yellow green blue indigo violet
rainbow3: red-orange-yellow-green-blue-indigo-violet
```

JSTL이 제공하는 태그의 종류와 사용법

JSTL Core 태그 <c:choose> <c:when> <c:otherwise> <c:out>

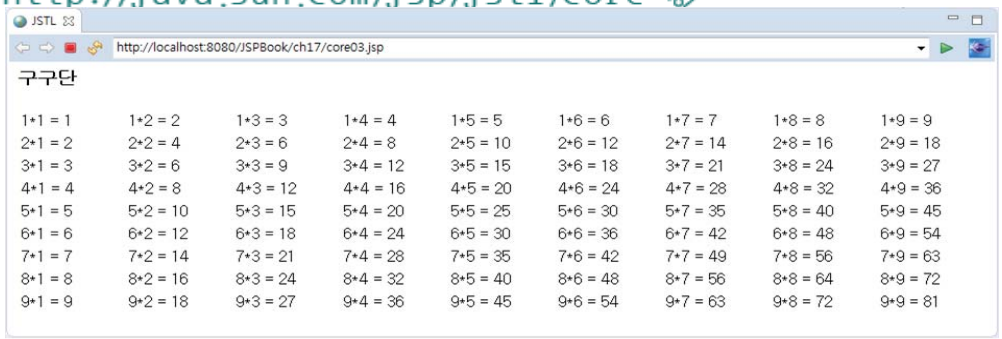
```jsp
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<html>
<head>
<title>JSTL</title>
</head>
<body>
    <%
        String number = request.getParameter("number");
    %>
    <c:set var="number" value="<%=number%>" />
    <c:choose>
        <c:when test="${number%2==0}">
            <c:out value="${number}" />은 짝수입니다.
        </c:when>
        <c:when test="${number%2==1}">
            <c:out value="${number}" />은 홀수입니다.
        </c:when>
        <c:otherwise>
            숫자가 아닙니다.
        </c:otherwise>
    </c:choose>
</body>
</html>
```



JSTL이 제공하는 태그의 종류와 사용법

JSTL Core 태그 <c:forEach>

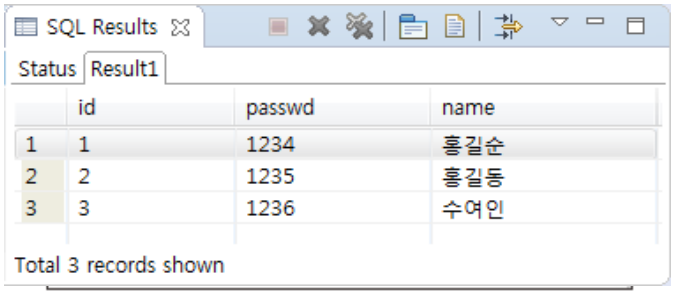
```
01  <%@ page contentType="text/html;charset=euc-kr"%>
02  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
03
04  <html>
05  <head>
06  <title>JSTL</title>
07  </head>
08  <body>
09      <h3>구구단</h3>
10      <table>
11          <c:forEach var="i" begin="1" end="9">
12              <tr>
13                  <c:forEach var="j" begin="1" end="9">
14                      <td width=100>${ i }*${ j } = ${ i * j }</td>
15                  </c:forEach>
16              </tr>
17          </c:forEach>
18      </table>
19  </body>
20  </html>
```



JSTL이 제공하는 태그의 종류와 사용법

JSTL SQL 태그

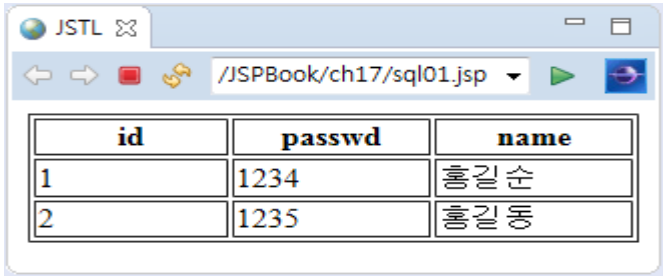
```
01  drop table member;
02
03  CREATE TABLE IF NOT EXISTS member(
04     id VARCHAR(20) NOT NULL,
05     passwd  VARCHAR(20),
06     name VARCHAR(30),
07     PRIMARY KEY (id)
08  );
09  INSERT INTO member VALUES('1', '1234', '홍길순');
10  INSERT INTO member VALUES('2', '1235', '홍길동');
11
12  select * from member;
```



JSTL이 제공하는 태그의 종류와 사용법

JSTL SQL 태그 <sql:setDataSource> <sql:query>

```
02  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
03  <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
04  <html>
05  <head>
06  <title>JSTL</title>
07  </head>
08  <body>
09      <sql:setDataSource var="dataSource"
10          url="jdbc:mysql://localhost:3306/JSPBookDB"
11          driver="com.mysql.jdbc.Driver" user="root" password="1234" />
12
13      <sql:query var="resultSet" dataSource="${dataSource}">
14          select * from member
15      </sql:query>
16
17      <table border="1">
18          <tr>
19              <c:forEach var="columnName" items="${resultSet.columnNames}">
20                  <th width="100"><c:out value="${columnName}" /></th>
21              </c:forEach>
22          </tr>
23          <c:forEach var="row" items="${resultSet.rowsByIndex}">
24          <tr>
```



JSTL이 제공하는 태그의 종류와 사용법

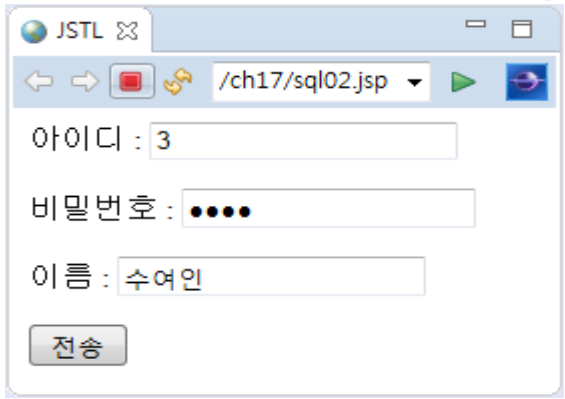
JSTL SQL 태그

```
25                <c:forEach var="column" items="${row}" varStatus="i">
26                <td>
27                     <c:if test="${column != null}">
28                          <c:out value="${column}" />
29                     </c:if>
30                     <c:if test="${column == null}">
31                           
32                     </c:if>
33                </td>
34                </c:forEach>
35           </tr>
36           </c:forEach>
37      </table>
38 </body>
39 </html>
```

JSTL이 제공하는 태그의 종류와 사용법

JSTL SQL 태그 <sql:update> INSERT

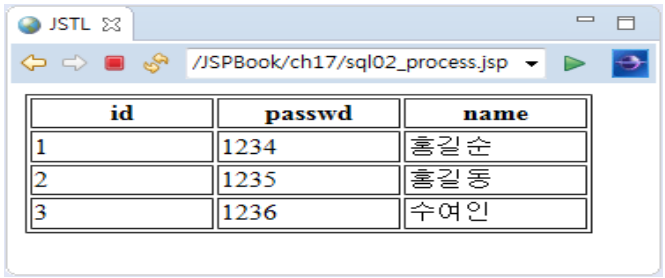
```
01  <%@ page contentType="text/html; charset=utf-8"%>
02  <html>
03  <head>
04  <title>Database SQL</title>
05  </head>
06  <body>
07      <form method="post" action="insert02_process.jsp">
08          <p>아이디 : <input type="text" name="id">
09          <p>비밀번호 : <input type="password" name="passwd">
10          <p>이름 : <input type="text" name="name">
11          <p><input type="submit" value="보내기">
12      </form>
13  </body>
14  </html>
```



JSTL이 제공하는 태그의 종류와 사용법

JSTL SQL 태그 <sql:update> INSERT

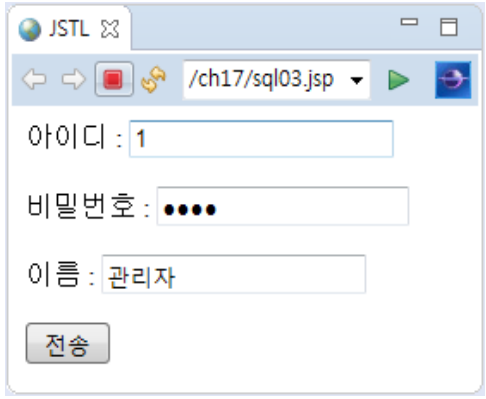
```
02  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
03  <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
04  <html>
05  <head>
06  <title>JSTL</title>
07  </head>
08  <body>
09      <%
10          request.setCharacterEncoding("utf-8");
11
12          String id = request.getParameter("id");
13          String passwd = request.getParameter("passwd");
14          String name = request.getParameter("name");
15      %>
16      <sql:setDataSource var="dataSource"
17          url="jdbc:mysql://localhost:3306/JSPBookDB"
18          driver="com.mysql.jdbc.Driver" user="root" password="1234" />
19
20      <sql:update dataSource="${dataSource}" var="resultSet">
21          INSERT INTO member(id, name, passwd) VALUES (?,?,?)
22          <sql:param value="<%=id%>" />
23          <sql:param value="<%=name%>" />
24          <sql:param value="<%=passwd%>" />
25      </sql:update>
26      <c:import var="url" url="sql01.jsp"  />
27      ${url}
28  </body>
29  </html>
```



JSTL이 제공하는 태그의 종류와 사용법

JSTL SQL 태그 <sql:update> UPDATE

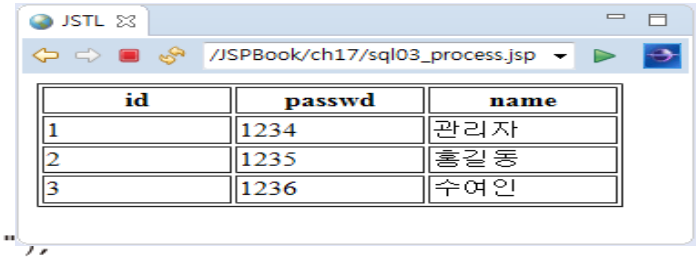
```
01 <%@ page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>JSTL</title>
05 </head>
06 <body>
07     <form method="post" action="sql03_process.jsp">
08         <p>아이디 : <input type="text" name="id">
09         <p>비밀번호 : <input type="password" name="passwd">
10         <p>이름 : <input type="text" name="name">
11         <p><input type="submit" value="전송">
12     </form>
13 </body>
14 </html>
```



JSTL이 제공하는 태그의 종류와 사용법

JSTL SQL 태그 <sql:update> UPDATE

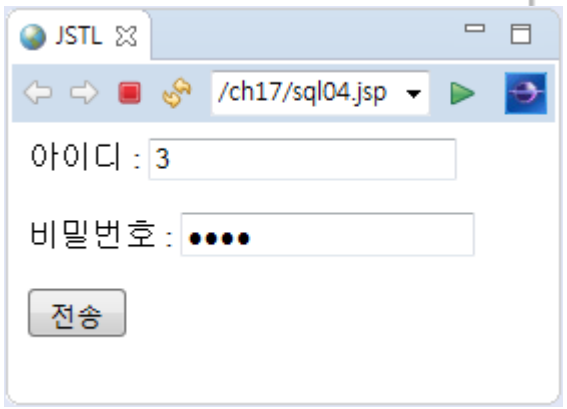
```
02  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
03  <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
04  <html>
05  <head>
06  <title>JSTL</title>
07  </head>
08  <body>
09      <%
10          request.setCharacterEncoding("utf-8");
11
12          String id = request.getParameter("id");
13          String passwd = request.getParameter("passwd");
14          String name = request.getParameter("name");
15      %>
16      <sql:setDataSource var="dataSource"
17          url="jdbc:mysql://localhost:3306/JSPBookDB"
18          driver="com.mysql.jdbc.Driver" user="root" password="1234" />
19
20      <sql:update dataSource="${dataSource}" var="resultSet">
21          UPDATE member SET name =?  where id =? and passwd =?
22          <sql:param value="<%=name%>" />
23          <sql:param value="<%=id%>" />
24          <sql:param value="<%=passwd%>" />
25      </sql:update>
26      <c:import var="url" url="sql01.jsp"  />
27      ${url}
28  </body>
29  </html>
```



JSTL이 제공하는 태그의 종류와 사용법

JSTL SQL 태그 <sql:update> DELETE

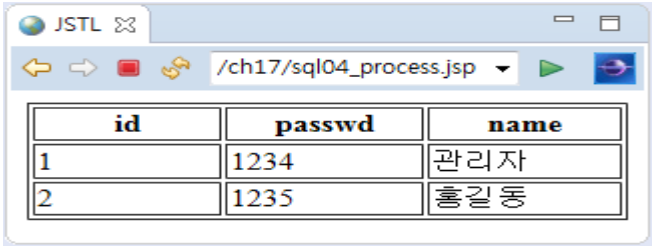
```
01  <%@ page contentType="text/html; charset=utf-8"%>
02  <html>
03  <head>
04  <title>JSTL</title>
05  </head>
06  <body>
07      <form method="post" action="sql04_process.jsp">
08          <p>아이디 : <input type="text" name="id">
09          <p>비밀번호 : <input type="password" name="passwd">
10          <p><input type="submit" value="전송">
11      </form>
12  </body>
13  </html>
```



JSTL이 제공하는 태그의 종류와 사용법

JSTL SQL 태그 <sql:update> DELETE

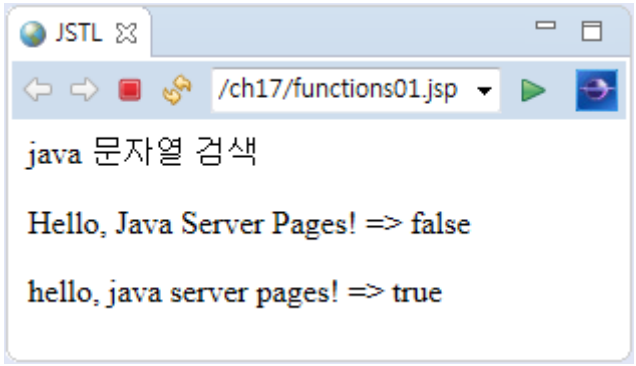
```
02  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
03  <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
04  <html>
05  <head>
06  <title>JSTL</title>
07  </head>
08  <body>
09      <%
10          request.setCharacterEncoding("utf-8");
11
12          String id = request.getParameter("id");
13          String passwd = request.getParameter("passwd");
14      %>
15      <sql:setDataSource var="dataSource"
16          url="jdbc:mysql://localhost:3306/JSPBookDB"
17          driver="com.mysql.jdbc.Driver" user="root" password="1234" />
18
19      <sql:update dataSource="${dataSource}" var="resultSet">
20          DELETE FROM member where id =? and passwd =?
21          <sql:param value="<%=id%>" />
22          <sql:param value="<%=passwd%>" />
23      </sql:update>
24      <c:import var="url" url="sql01.jsp"  />
25      ${url}
26  </body>
27  </html>
```



JSTL이 제공하는 태그의 종류와 사용법

JSTL functions 태그 <fn:contains> <fn:containsIgnore>

```
01 <%@ page contentType="text/html; charset=utf-8"%>
02 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
03 <html>
04 <head>
05 <title>JSTL</title>
06 </head>
07 <body>
08     <p>java 문자열 검색
09     <p>Hello, Java Server Pages! => ${fn:contains("Hello, Java Server Pages!",
       "java")}
10     <p>hello, java server pages! => ${fn:containsIgnoreCase("hello, java
       server pages!", "java")}
11 </body>
12 </html>
```



JSTL이 제공하는 태그의 종류와 사용법

JSTL functions 태그 <fn:split> <fn:join>

```
01  <%@ page contentType="text/html; charset=utf-8"%>
02  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
03  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
04  <html>
05  <head>
06  <title>JSTL</title>
07  </head>
08  <body>
09      <c:set var="texts" value="${fn:split('Hello,
        Java Server Pages!', ' ')}" />
10      <c:forEach var="i" begin="0" end="${fn:length(
11          <p>text[${i}] =${texts[i]}
12      </c:forEach>
13      <p><c:out value="${fn:join(texts, '-')}" />
14  </body>
15  </html>
```