2020년도 2학기

웹프로젝트실습 과목

- 14주차 -

장용미

H.P: 010-3309-4849

E-MAIL: changmi29@dongyang.ac.kr

eClass의 쪽지

14주차 오늘의 학습 내용

1. EL (교재 10장)

2. 커스텀태그 (교재 10장)

3. JSTL (교재 11장)

4. 최종 팀프로젝트 발표 일정 (기말고사)

프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음

Chapter 10. 표현 언어와 커스텀 태그

한빛아카데미 Hanbit Academy, Inc.

목차

학습목표

- 표현 언어의 개념을 이해하고 JSP에서의 활용 방법을 알아본다.
- 커스텀 태그의 처리 과정을 이해하고 taglib 지시어를 익힌다.
- 태그 파일 기반의 커스텀 태그 개발 방법을 익힌다.
- 태그 핸들러 기반의 커스텀 태그 개발 방법을 익힌다.

01. 표현 언어

1. 표현 언어란

- **표현 언어(Expression Language)**는 처음 JSTL(JSP Standard Tag Library)이 소개되었을 때 나온 것으로, MVC 패턴에 따라 뷰(view) 역할을 수행하는 JSP를 더욱 효과적으로 만들려는 목적으로 개발되었다.
- 표현 언어는 간단한 방법으로 데이터를 표현하기 위해 고안된 언어인 SPEL(Simplest Possible Expression Language)에 기본을 두고 있다.

JSP 표현식 사용

```
<H2>
<jsp:useBean id="test" class="TestBean" />
<%= test.getName() %> 혹은 <jsp:getProperty name="test" property="name" />
</H2>
```

EL 사용

```
<H2>
${test.name}
</H2>
```

01. 표현 언어

- 표현 언어 사용을 위해서는 현재 페이지에서 출력하고자 하는 데이터가 미리 확보 되어 있어야 한다. 예를 들면 page, request, application, session 내장 객체중 하나에 사용하고자 하는 객체가 있어야만 표현언어를 이용해 데이터 출력이 가능 하다.
- 표현언어의 기본적인 문법은 다음과 같다.

  - 표현 언어는 '$'로 시작한다.
  - 모든 내용은 '{표현식}'과 같이 구성된다.
  - 표현식에는 기본적으로 변수 이름, 혹은 '객체_이름.멤버변수_이름'구조로 이루어진다.
  - 표현식에는 부가적으로 숫자, 문자열, boolean, null과 같은 상수 값도 올 수 있다.
  - 표현식에는 기본적인 연산을 할 수 있다.

2. 표현 언어에서 사용할 수 있는 내장 객체

- 표현언어에서는 객체가 생성되어 전달된다는 것을 가정하고 있다. 따라서 표현언어 에서 사용 시점에 객체를 선언할 필요가 없음.
- 표연언어에서는 다음과 같이 객체에 접근 할 수 있음.

> ${member.id} 혹은 ${member["id"]} → member 객체의 getId() 메서드 호출과 동일
> ${row[0]} → row라는 이름의 컬렉션 객체의 첫 번째 값

- 또한 몇몇 내장객체를 통해 컨테이너가 제공하는 다른 객체에 접근할 수 있는 방법을 제공하고 있다.
- 표현언어에서 사용할 수 있는 내장객체는 [표 10-1] 참조

01. 표현 언어

[표 10-1] 표현 언어에서 사용할 수 있는 내장객체

내장객체	기능
pageScope	page 범위에 포함된 속성 값에 접근할 수 있는 객체다.
requestScope	request 범위에 포함된 속성 값에 접근할 수 있는 객체다.
sessionScope	session 범위에 포함된 속성 값에 접근할 수 있는 객체다.
applicationScope	application 범위에 포함된 속성 값에 접근할 수 있는 객체다.
param	request.getParameter('xxx')로 얻을 수 있는 값들이다. ${param.xxx}처럼 사용한다.
paramValues	request.getParameterValues('xxx')와 동일한 기능을 수행한다. ${paramValues.xxx}처럼 사용한다.
header	request.getHeader('xxx')와 동일한 기능을 수행한다. ${header.xxx}처럼 사용한다.
headerValues	request.getHeaderValues('xxx')와 동일한 기능을 수행한다. ${headerValues.xxx}처럼 사용한다.
initParam	컨텍스트의 초기화 매개변수 값이다.
cookie	쿠키 정보에 접근할 수 있는 객체다.
pageContext	pageContext 객체다.

3. 표현 언어에서 사용할 수 있는 연산자

- 표현 언어에서는 표현식 부분에서 기본적인 연산을 할 수 있으며 다음과 같은 연산자 사용이 가능하다.

[표 10-2] 산술 연산자

연산자	기능	연산자	기능
+	더하기	-	빼기
*	곱하기	/ or div	나누기
% of mod	몫		

[표 10-3] 비교/조건 연산자

연산자	기능	연산자	기능
== 혹은 eq	같다.	!= 혹은 ne	같지 않다.
< 혹은 lt	좌변이 우변보다 작다.	> 혹은 gt	좌변이 우변보다 크다.
<= 혹은 le	좌변이 우변보다 같거나 작다.	>= 혹은 ge	좌변이 우변보다 같거나 크다.
a?b :c	a가 참이면 b, 거짓이면 c를 반환한다.		

[표 10-4] 관계 연산자

연산자	기능
&& 혹은 and	AND 연산
|| 혹은 or	OR 연산
! 혹은 not	NOT

EL (Expression Language) 표현언어

원리를 알면 IT가 맛있다
IT COOKBOOK

자바 빈의 프로퍼티, 값을 JSP의 표현식 <%= %>이나 액션 태그 <jsp:useBean>를 사용하는것 보다 쉽고 간결하게 꺼낼수 있게 하는 기술이다.

또한 static 메소드를 호출할 수도 있는데 JSP에서는 주로 서블릿 보관소(JspContext, ServletRequest, HttpSession, ServletContext)에서 값을 꺼낼 때 사용한다.

< 구글링하여 찾은 EL 강좌 페이지 >

https://atoz-develop.tistory.com/entry/JSP-EL-%ED%91%9C%ED%98%84%EC%8B%9D-%EB%AC%B8%EB%B2%95%EA%B3%BC-%EC%82%AC%EC%9A%A9-%EB%B0%A9%EB%B2%95

02. [기본실습] 표현 언어의 기본 이해

1. 실습 개요

- 실제 표현언어가 활용되는 사례를 통해 표현 언어에 익숙해 지기 위한 간단한 애플리케이션을 개발한다.
- 애플리케이션은 상품 목록을 보여주고 상품을 선택하면 선택된 상품에 대한 정보를 보여주는 예제이다.

[표 10-5] 프로그램 소스 목록

디렉터리	파일
src₩	Product.java
WebContent₩ch10	ProductList.jsp
WebContent₩ch10	ProductSel.jsp

- Product.java : 상품 정보를 제공하는 빈즈 클래스로 jsp에 데이터를 공급하는 역할을 한다.
- ProductList.jsp : 상목 목록을 출력하는 jsp파일로, Product 클래스로부터 상품 목록을 가져와 출력.
- ProductSel.jsp : ProductList.jsp에서 상품을 선택하고 <확인> 버튼을 눌렀을때 호출되는 jsp로, 표현 언어를 이용해 데이터를 출력한다.

02. [기본실습] 표현 언어의 기본 이해

2. 소스 작성

- **Product 클래스 작성**

  Product 클래스(Product.java) – 교재 p.431 ~ 432 참고

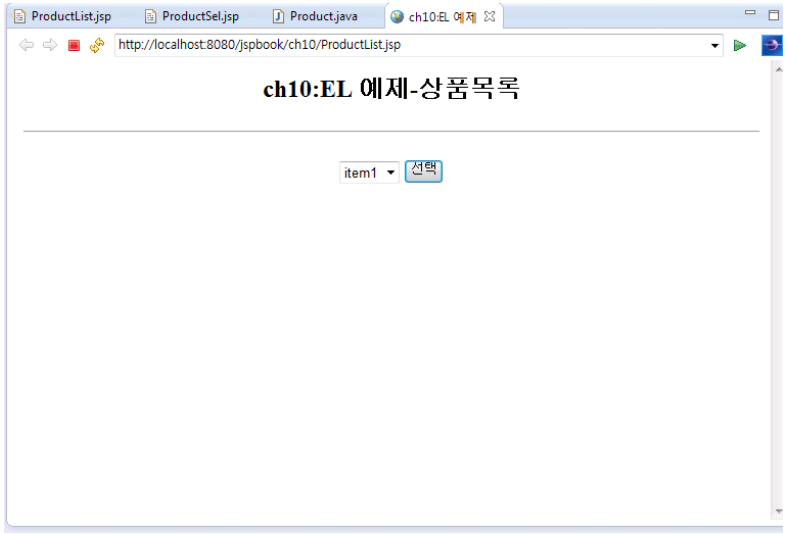
- **ProductList.jsp 작성**

  상품 목록 출력(ProductList.jsp) – 교재 p.432 ~ 433 참고

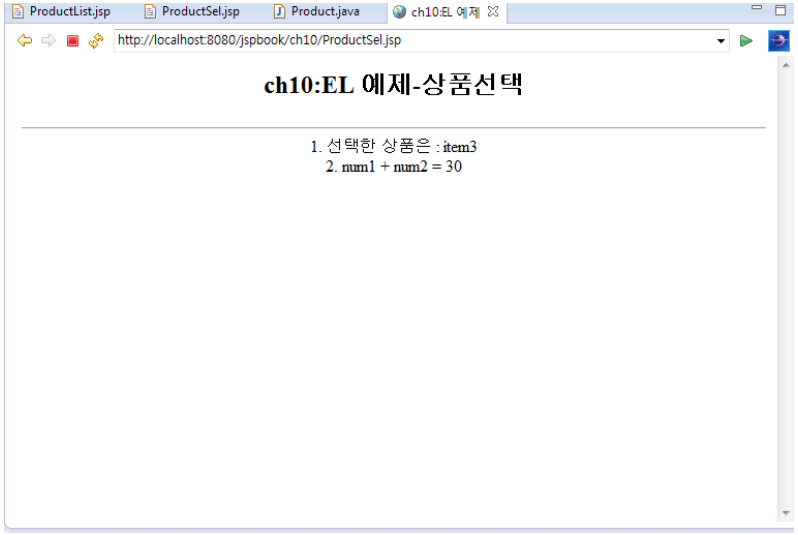
- **ProductSel.jsp 작성**

  상품 선택(ProductSel.jsp) – 교재 p.433 ~ 434 참고

3. 실행 및 결과 확인



[그림 10-1] ProductList.jsp 실행 화면



[그림 10-2] ProductSel.jsp 실행 화면

02. [기본실습] 표현 언어의 기본 이해

주요 소스코드 분석(Product.java)

- 데이터 관리를 위한 빈즈 클래스
- 상품정보는 편의상 배열로 만들어 제공
- 표현언어 실습을 위해 num1, num2 라는 변수를 미리 설정해 둠.

```
06    private String[] productList = {"item", "item2", "item3", "item4", "item5"};
07    private int num1 = 10;
08    private int num2 = 20;
```

주요 소스코드 분석(ProductList.jsp)

- 상품 목록을 HTML 의 <select><option> 을 이용해 출력하는 JSP
- <jsp:useBean>을 이용해 Product 클래스 사용
- for 문을 이용해 배열 데이터를 <select><option> 문과 조합해 출력

```
13 <jsp:useBean id="product" class="jspbook.ch10.Product" scope="session" />
14    <select name="sel">
15    <%
16        for(String item : product.getProductList()) {
17            out.println("<option>"+item+"</option>");
18        }
19    %>
20    </select>
```

02. [기본실습] 표현 언어의 기본 이해

주요 소스코드 분석(ProductSel.jsp)

- 선택된 상품정보를 출력하는 JSP
- 표현언어를 통해 선택된 상품과 product 객체의 정보를 출력

```
12 1. 선택한 상품은 : ${param.sel} <BR>
13 2. num1 + num2 = ${product.num1+product.num2} <BR>
```

03. 커스텀 태그

1. 커스텀 태그란?

- 커스텀 태그(Custom Tag)는 원래 JSP 페이지에서 반복적인 프로그램 로직을 캡슐화하기 위해 고안됨.
- 기본적으로 제공되는 태그 이외 사용자가 확장한 태그라는 의미에서 붙여진 이름.
- HTML 문서는 브라우저에 의해 해석되므로 커스텀 태그를 구현할 수 없지만, JSP는 서버에서 해석되므로 커스텀 태그를 구현할 수 있다.

```
<table cellpadding=5 cellspacing=0 border="1">
    <tr bgcolor="#99CCFF"><td>번호</td><td>작성자</td><td>전화
    번호</td><td>작성일</td><td>내용</td></tr>
    <%
        for(GuestBook guestbook : datas) {
    %>
        <tr>
            <td><%=guestbook.getGb_id() %></td>
            <td><%=guestbook.getGb_name() %></td>
            <td><%=guestbook.getGb_tel() %></td>
            <td><%=guestbook.getGb_date() %></td>
        </tr>
        <%
            }
        %>
</table>
```

```
<%@ taglib uri="getListTag" prefix="boardTag" %>
<HTML>
<BODY>
<boardTag:getList />
</BODY>
</HTML>
```

03. 커스텀 태그

- 웹 화면과 프로그램이 복잡해질수록 커스텀 태그를 사용하면 소스를 간결하게 유지할 수 있고 구현된 기능의 재활용이 용이해진다.
- 커스텀태그 라이브러리를 사용했을 때의 장점은 다음과 같다.

❶ 비즈니스 로직으로부터 화면 표현을 분리할 수 있다 : 프로그램 개발자는 비즈니스 로직을 개발하고 표현을 위한 커프텀 태그를 제공하며, 화면 개발자는 화면 디자인에 집중할 수 있다.

❷ 비즈니스 로직의 캡슐화할 수 있다 : 재사용할 수 있고 유지보수에 유리하다.

❸ 보다 완벽한 MVC 패턴을 구현할 수 있다 : JSP를 순수하게 뷰 형태로 사용할 수 있으므로 더욱 완벽한 MVC 패턴을 구현할 수 있다.

03. 커스텀 태그

- 커스텀 태그를 구현하는 방법에는 크게 두 가지가 있다.
  - **태그 파일 기반의 개발 방법**
  - **태그 핸들러 클래스 기반 개발 방법**

- 태그 파일 기반의 개발은 비교적 쉽게 jsp 기술들을 활용해 커스텀 태그를 만들 수 있어 대부분의 경우에 권장되지만 구조상 소스가 노출되고 라이브러리화 하기 어렵기 때문에 고급 태그나 라이브러리 개발의 경우에는 태그 핸들러 클래스 기반으로 개발하는 것이 좋다.

[표 10-6] **유형별 커스텀 태그 개발 방법**

태그 개발 목적		개발 방법
• 많은 화면을 생성해내는 태그 • 단일 사이트에 적용	• 비교적 간단한 기능들로 구성	태그 파일 기반 개발 방법
• 다양한 기능 제공 • 여러 사이트에 적용	• 태그 라이브러리로 개발 • 공개 목적	핸들러 클래스 기반 개발 방법 (SimpleTag, SimpleTagSupport 등 태그)

03. 커스텀 태그

2. 태그의 기본 구조

- 커스텀 태그도 일반적인 태그와 동일한 구조를 가지므로, 일반적인 태그의 구조를 먼저 알아본다.
- 다음과 같은 태그의 구성 요소는 태그, 속성, 태그 바디가 된다.

```
<H2>custom tag test</H2>
<form name="form1" method="post">
<input type="text" name="item1">
</form>
```

- **태그**
  - 기본적으로 모든 태그는 쌍으로 이루어지며 속성을 포함하고 있다. <H2></H2>, <form></form>, <input>은 모두 태그인데, <input>의 경우 쌍으로 이루어지지 않았다. HTML의 경우에는 크게 상관이 없으나, XML 형식을 따르는 경우에는 <input></input>, 혹은 <input/>과 같이 해주어야 한다.
- **속성**
  - 속성은 태그 안에서 부가적인 정보를 제공하는 것으로, <form> 태그 내에 있는 name과 method, <input> 태그에 있는 type과 name 등이 속성에 해당한다. HTML 자체에서는 제약이 크지 않으나, 가급적 속성 값들은 " " 로 묶어주는 것이 좋다.
- **태그 바디**
  - 시작 태그와 종료 태그 사이에 있는 내용을 말한다. <H2> 태그의 경우 문자열이나 화면 표현과 관련된 다른 태그가 올 수 있다. <form> 태그의 경우 다른 태그나 문자열이 올 수 있지만 반드시 <form> 태그 내에 있으야 하는 <input>, <select> 등의 태그도 포함된다.

03. 커스텀 태그

3. taglib 지시어

- 커스텀 태그를 사용하기 위해서는 태그 사용을 원하는 jsp에 taglib 지시어를 기술해 주어야 한다.
- taglib 지시어는 태그에 대한 정보 수집을 위한 uri 혹은 태그파일 디렉토리와 태그에 붙이기 위한 prefix 정보를 등록한다.

```
<%@ taglib uri="/WEB-INF/tld/MsgTag.tld" prefix="mytag" %>
          ❶                              ❷
```

❶ uri : tld 파일 위치를 지정한다. 대개 태그 라이브러리를 관리하는 단체 uri가 온다. 로컬 디렉터리에 태그 라이브러리 기술자 파일이 위치하는 경우 경로를 넣어주면 된다. 태그 파일로 구현된 커스텀 태그를 사용하려면 uri 대신 tagdir을 쓴다.

❷ prefix : 커스텀 태그를 구분하기 위한 이름으로, 한 페이지에 커스텀 태그를 사용할 경우에는 prefix를 이용하면 혼동을 피할 수 있다. 동일한 이름의 태그가 성격이 다른 여러 태그 라이브러리에 있을 수 있기 때문에 이를 구분하기 위한 목적으로 사용하는 것이다. 본문에서는 taglib 지시어에서 정한 prefix에 따라 커스텀 태그를 사용하면 된다.

- 본문에서 태그를 사용할때에는 다음과 같이 prefix:태그_이름 형태로 사용한다.

```
<prefix:태그_이름> 태그 바디</prefix:태그_이름>
```

04. 태그파일 기반 커스텀 태그

1. 태그 파일 개요

- 태그 파일을 이용하면 비교적 간단하고 JSP 페이지 개발과 유사한 구조로 태그 파일을 만들 수 있다.
- tag 지시어를 사용해 태그 파일을 선언하고 JSP 문법과 표현언어, JSTL 등을 자유롭게 사용할 수 있다.
- 기본적으로 태그 파일은 JSP와 유사하나 다음과 같은 차이가 있다.
  - **태그 파일** : .tag 파일로서, 몇 가지 제약사항을 제외하고는 대부분의 JSP 파일과 구성이 동일하다. [WEBINF₩tag] 폴더에 저장한다. 표현언어와 JSTL 등을 사용할 수 있다.
  - **JSP 파일** : 커스텀 태그를 사용하려면 JSP 파일에 taglib 지시어를 설정한 후 커스텀 태그를 사용한다.

04. 태그파일 기반 커스텀 태그

2. 태그 파일의 구조

- 태그 파일은 실제 태그의 기능을 구현한 파일.
- 기본적인 JSP 지시어와 추가된 tag 지시어 attribute 와 variable 이라는 태그를 사용할 수 있다.

[표 10-7] 태그 파일에서 사용하는 지시어

지시어	설명
taglib	JSP에서의 taglib 지시어와 동일하다. 현재 태그 파일에서 다른 커스텀 태그나 JSTL 등을 사용하려면 해당 태그 라이브러리를 사용하기 위한 taglib 지시어를 작성한다.
include	JSP에서의 include 지시어와 동일하다. 그러나 포함되는 파일의 구조가 태그 파일의 구조를 따라야만 한다.
tag	새로운 지시어로, page 지시어와 유사하다. 현재 파일이 태그 파일이라는 것을 알리고 옵션을 설정한다.
attribute	작성하는 태그에 포함될 속성 등을 정의한다.
variable	태그 내용에 필요한 표현 언어 변수를 선언한다.

- tag 지시어에 대한 자세한 속성은 [표 10-8]을 참조하도록 한다.

05. [기본실습]태그파일 기반 커스텀 태그: 기본 태그 구현

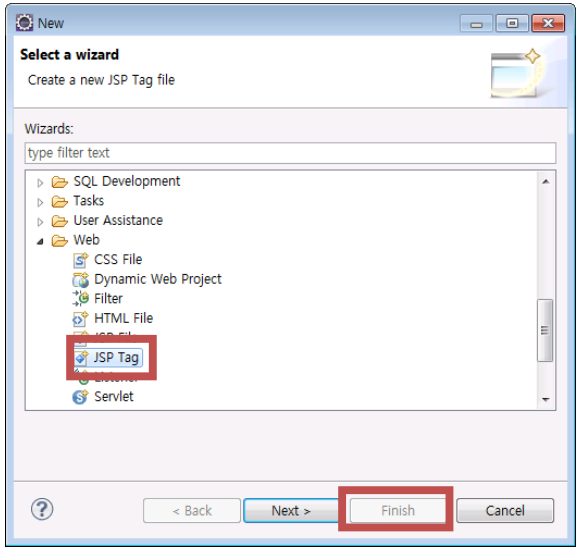
1. 실습 개요

- 간단한 메시지를 출력하는 태그 파일 개발 과정을 통해 태그 파일의 구조와 활용 방법을 익힌다.
- 여기서는 단순히 커스텀 태그의 개념과 동작 원리 정도만 살펴보는 것으로 실제 활용 가능한 수준의 커스텀 태그 개발은 [응용실습]에서 배운다.

[표 10-9] 프로그램 소스 목록

디렉터리(jspbook 프로젝트)	파일
WEB-INF₩tags	print.tag
WebContent₩ch10	PrintTagTest.jsp

2. 소스 작성



[그림 10-3] JSP Tag 파일 생성

05. [기본실습]태그파일 기반 커스텀 태그: 기본 태그 구현

원리를 알면 IT가 맛있다 IT COOKBOOK

주요 소스코드 분석

커스텀 태그 파일 : print.tag – 교재 p.444 참고

- tag 지시어를 사용해 태그 파일임을 알린다.
- body-content="empty" 로 설정해 태그 바디가 없는 태그로 정의 한다.
- 태그를 만나면 간단한 메시지를 출력 한다.

```
01 <%@ tag body-content="empty" pageEncoding="UTF-8" %>
02 커스텀 태그 출력 메시지: Hello!!
```

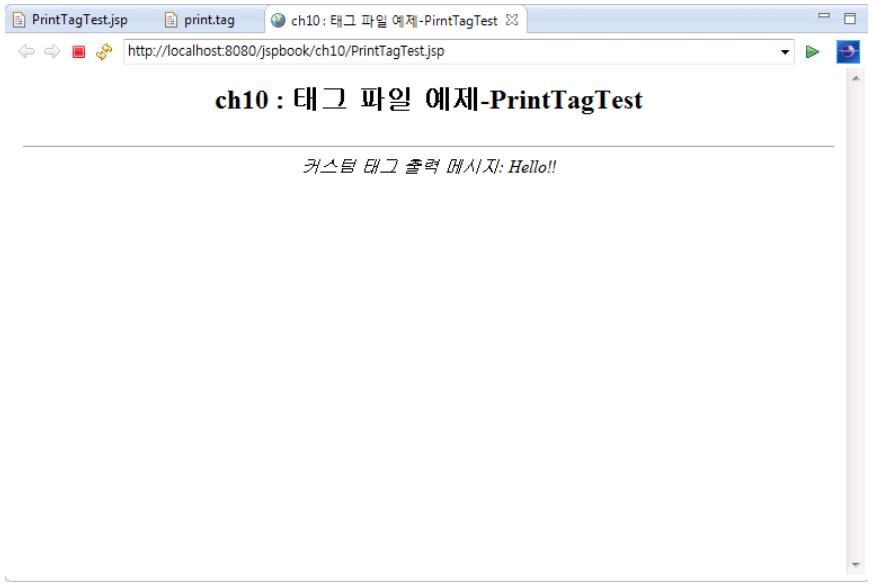
커스텀 태그 파일 : PrintTagTest.jsp – 교재 p.444 ~ 445 참고

- print.tag 커스텀 태그를 테스트 하기 위한 jsp
- 기본 JSP 파일에 taglib 지시어로 커스텀 태그 사용을 선언하고 본문에서 태그 사용

```
02 <%@ taglib tagdir="/WEB-INF/tags" prefix="mytag" %>
```

```
10 <BODY>
11 <center>
12 <H2>ch10: 태그 파일 예제 – PrintTagTest</H2>
13 <HR>
14 <I><mytag:print/></I>
15 </center>
16 <BODY>
```

3. 실행 및 결과 확인



[그림 10-4] 실행 결과

06. [응용실습]태그파일 기반 커스텀 태그: 복합 태그 구현

1. 실습 개요

- 실제 활용 가능한 수준의 태그파일 기반의 커스텀 태그 구현을 통해 태그 파일에 대한 이해 심화.
- 태그 구성요소인 태그바디, 속성을 모두 가지는 태그로 개발
- 빈즈 클래스와 연동해 데이터를 출력하는 기능을 수행(프로그램을 커스텀 태그로 캡슐화)

[표 10-10] 프로그램 소스 목록

디렉터리(jspbook 프로젝트)	파일
WEB-INF₩tags	item.tag
WebContent₩ch10	ListItem.jsp
src₩	Product.java(표현 언어에서 만들었다)

06. [응용실습]태그파일 기반 커스텀 태그: 복합 태그 구현

2. 소스 작성

item.tag 태그 파일 작성

- **커스텀 태그 파일 : item.tag** – 교재 p.448 참고
  - 표현언어에서 만들었던 Product 빈즈 클래스의 배열 데이터를 테이블 형태로 출력해 주는 커스텀 태그
  - 배경색상과 테두리를 태그 속성으로 지정 할 수 있도록 함.

ListItem.jsp 파일 작성

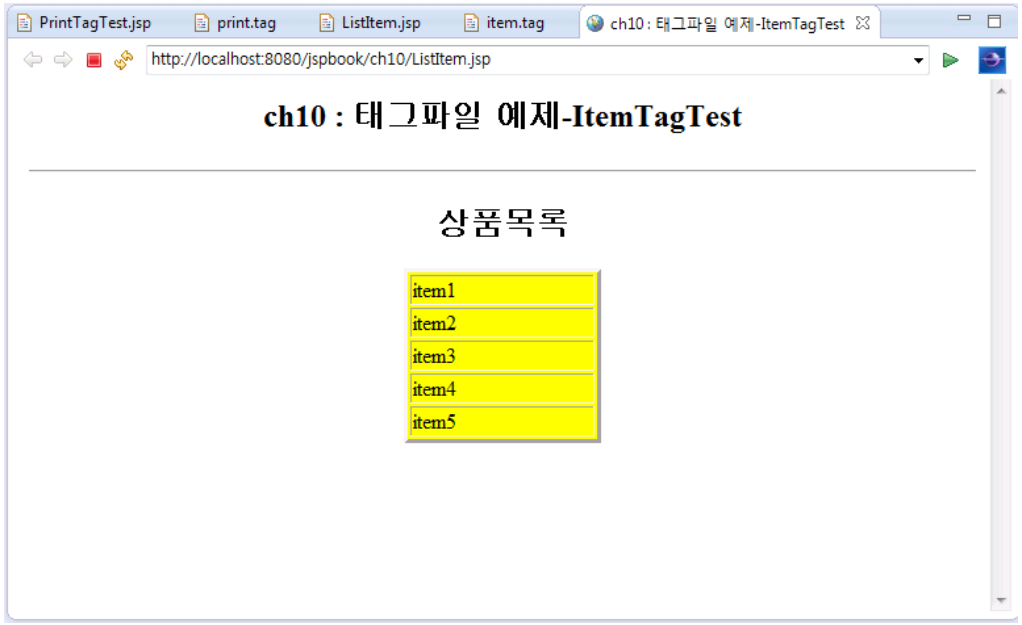
- **JSP 파일 : ListItem.jsp** – 교재 p.449 참고
  - 만들어진 item 태그를 사용해 상품 목록을 출력하기 위한 jsp 페이지
  - taglib 지시어로 태그 사용을 선언하고 본문에서 속성과 태그 바디를 넣어 태그 작성

```
03 <%@ taglib tagdir="/WEB-INF/tags" prefix="mytag" %>
```

```
13 <H2>ch10 : 태그 파일 예제-ListItem</H2>
14 <HR>
15 <mytag:item border="3" bgcolor="yello">상품 목록</mytag:item>
```

06. [응용실습]태그파일 기반 커스텀 태그: 복합 태그 구현

3. 실행 및 결과 확인



[그림 10-5] ListItem.jsp 실행 화면

07. 태그파일 기반 커스텀 태그

1. 태그 핸들러 개요

- 태그 핸들러란 커스텀 태그를 처리하는 객체를 말한다. 앞에서 배운 태그 파일과 달리 자바 클래스를 이용해 커스텀 태그를 구현하는 방법이다.
- SimpleTag 인터페이스가 제공되면서 SimpleTagSupport 클래스를 상속받아 이전보다는 비교적 쉽게 구현할 수 있게 되었다. 그러나 태그 파일을 이용한 방식보다는 많이 복잡하고 자바 프로그램의 비중이 높으므로 구현 난이도는 높은 편이다.
- 태그 핸들러 기반 커스텀 태그는 세가지 구성요소를 가지면 태그 파일의 경우에는 이러한 부분이 간소화 되어 있다.

[표 10-11] **태그 핸들러 기반 커스텀 태그 구성요소**

구성요소	비고
태그 핸들러 클래스(Tag Handler Class)	태그 파일의 경우에는 별도 핸들러 클래스가 없다.
태그 라이브러리 기술자(Tag Library Descriptor)	태그 파일의 경우 태그 파일이 기술자를 겸한다.
taglib 지시어(taglib Directives)	커스텀 태그를 사용하는 모든 JSP에서 선언해야 한다.

07. 태그파일 기반 커스텀 태그

- **태그 핸들러 클래스**

  커스텀 태그를 실제 구현한 자바 클래스다. 태그 라이브러리 기술자에서 설계된 내용을 구현해야 한다. 태그 라이브러리 기술자와 마찬가지로 태그 파일 기반의 커스텀 태그에서는 필요 없다.

- **태그 라이브러리 기술자(Tag Library Descriptor)**

  xml 규격으로 커스텀 태그에 대한 구조를 정의하는 파일이다. .tld 파일로 만들어지며 태그 파일 기반의 커스텀 태그에서는 필요하지 않다.

- **taglib 지시어**

  jsp 지시어의 한 종류로, JSP 페이지에 공통으로 필요한 정보를 기술하는 부분이다. 커스텀 태그 사용을 위한 태그 파일 혹은 태그 라이브러리 기술자의 위치 등을 설정한다. 따라서 커스텀 태그를 사용하는 모든 JSP 페이지에 taglib 지시어를 사용해야 한다.

07. 태그파일 기반 커스텀 태그

2. 태그 핸들러 클래스 구조

- 기본적으로 커스텀 태그 개발은 SimpleTag 인터페이스를 구현하는 것이지만 보통은 SimpleTag 인터페이스를 구현하고 있는 SimpleTagSupport 클래스를 상속받아 자신만의 커스텀 태그를 만들게 된다.

[표 10-12] SimpleTagSupport 클래스 주요 메서드

메서드	설명
`void doTag()`	시작 태그를 만나면 호출되는 메서드, 즉 실제 태그 기능을 정의하는 메서드다.
`JspFragment getJspBody()`	태그 바디를 처리하려고 JspFragmet를 얻어오기 위한 메서드다. JspFragment 객체를 이용해 바디를 처리한다.
`JspContext getJspContext()`	JspContext를 얻어오는 메서드로써, out 객체의 참조 등 해당 JSP로부터 다양한 정보를 얻을 수 있다.

- 실제 개발에 가장 중요한 메서드는 doTag() 메서드로, 태그의 실제 기능을 구현하는 메서드라 할 수 있다.

08. [기본실습]태그 핸들러 기반 커스텀 태그: 기본 태그 구현

1. 실습 개요

- 간단한 메시지를 출력하는 태그 파일 개발 과정을 통해 태그 핸들러 기반 커스텀 태그 구조와 활용 방법을 익힌다.
- 여기서는 태그 파일로 만들었던 print.tag 와 동일한 기능을 태그 핸들러 기반으로 개발해 본다.

[표 10-13] 프로그램 소스 목록

디렉터리(jspbook 프로젝트)	파일
WebContent₩ch10	MsgTagSimpleTest.jsp
src	MsgTagSimpleHandler.java
WEB-INF₩tld	MsgTagSimple.tld

- **MsgTagSimpleTest.jsp** : 커스텀 태그를 사용할 때 필요한 jsp 파일로, 내용상 taglib 지시어를 제외한 나머지 내용은 PrintTagTest.jsp와 동일하다.
- **MsgTagSimpleHandler.java** : 커스텀 태그 구현 클래스로, SimpleTagSupport 클래스를 상속받아 구현한다.
- **MsgTagSimple.tld** : 커스텀 태그 정의 파일이 된다. 태그 파일의 경우 별도의 태그 정의 파일 없이 태그 파일만으로 처리할 수 있었으나 여기서는 별도의 xml 규격 파일로 우리가 만드는 태그를 먼저 정의해주어야 한다.

2. 소스 작성

■ 태그 핸들러 클래스 : MsgTagSimpleHandler.java 작성

▪ 태그 핸들러 클래스(MsgTagSimpleHandler.java) – 교재 p.454 참고

- SimpleTagSupport 인터페이스를 상속해 구현한다.
- doTag() 메서드에서 태그에서 처리할 내용을 구현한다.

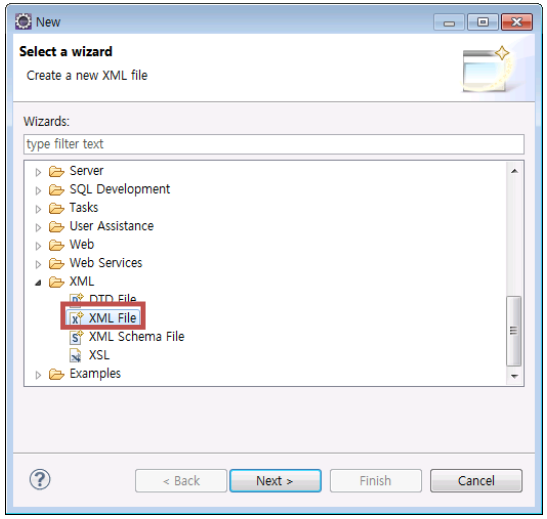
```
09 public class MsgTagSimpleHandler extends SimpleTagSupport {
10
11     public void dotag() throws IOException {
12         JspWriter out = getJspContext().getOut();
13         out.println("커스텀 태그 출력 메시지: Hello !!");
14     }
15 }
```

08. [기본실습]태그 핸들러 기반 커스텀 태그: 기본 태그 구현

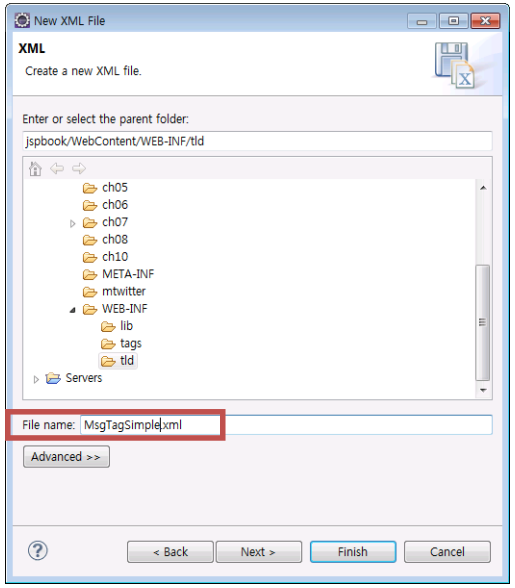
태그 라이브러리 기술자 : MsgTagSimple.tld 작성

❶ [WebContent₩WEB-INF₩tld] 폴더를 생성한 후 [File] → [New] → [XML] → [XML File]을 선택한다.
tld 폴더를 선택한 다음 파일 이름을 MsgTagSimple.tld로 입력한다.

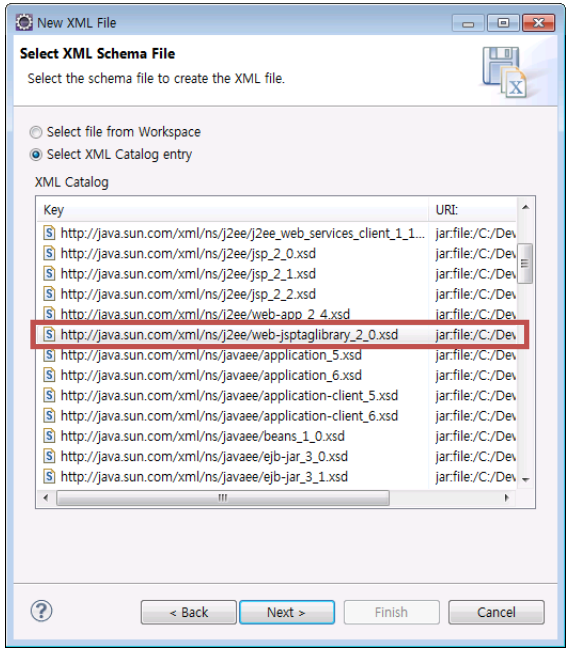
❷ Create XML file from a XML schema file을 선택하고, XML Schema를 선택하는 화면에서는 Select XML Catalog Entry를 선택한 다음 "http://java.sun.com/xml/ns/j2ee/web-Jsptaglibrary_2_0.xsd"를 선택한다.



[그림 10-6] xml 파일 생성 화면



[그림 10-7] MsgTagSimple.tld 생성 화면



[그림 10-8] MsgTagSimple.tld 생성 화면

08. [기본실습]태그 핸들러 기반 커스텀 태그: 기본 태그 구현

❸ 기본 코드 수정

- MsgTagSimple.tld – 교재 p.456 참고
  - MsgTagSimpleHandler 클래스를 사용하는 커스텀 태그에 대한 정의 파일이다.
  - xml 형식이며 [표 10-14], [표 10-15] 를 참고해 작성한다.

```
03    <description>간단한 커스텀 태그 예제</description>
04    <tlib-version>1.0</tlib-version>
05
06        <short-name>MsgTagSimple</short-name>
07
08    <tag>
09        <name>print</name>
10        <tag-class>jspbook.ch10.MsgTagSimpleHandler</tag-class>
11        <body-content>empty</body-content>
12    </tag>
```

08. [기본실습]태그 핸들러 기반 커스텀 태그: 기본 태그 구현

MsgTagSimpleTest.jsp 파일 작성

MsgTagSimpleTest.jsp – 교재 p.458 참고

- MsgTagSimple.tld를 참조해 커스텀 태그를 사용하는 jsp 파일이다.
- 태그 사용을 위한 taglib 지시어에는 uri에 tld 파일 경로를 적어 준다.

```
03 <%@ taglib uri="/WEB-INF/tld/MsgTagSimple.tld" prefix="mytag" %>
```

- 본문에서의 태그 사용은 태그 파일과 동일하다.

```
13 <H2>ch10 : 커스텀 태그 예제-MsgTagTest</H2>
14 <HR>
15 <I><mytag:print/></I>
```

08. [기본실습]태그 핸들러 기반 커스텀 태그: 기본 태그 구현

3. 실행 및 결과 확인

[그림 10-9] MsgTagSimple.tld 생성 화면

09. [응용실습]태그 핸들러 기반 커스텀 태그: 복합 태그 구현

1. 실습 개요

- 실제 활용 가능한 수준의 태그 핸들러 기반 커스텀 태그 구현을 통해 태그 파일에 대한 이해 심화.
- 태그 구성요소인 태그바디, 속성을 모두 가지는 태그로 개발
- 태그 파일에서 구현 했던 Product 빈즈 클래스를 사용해 목록을 출력하는 태그를 핸들러 기반으로 구현.

[표 10-16] 프로그램 소스 목록

디렉터리(jspbook 프로젝트)	파일
WebContent₩ch10	ItemTagTest.jsp
src	ItemTagHandler.java
WEB-INF₩tld	ItemTag.tld

- **ItemTagTest.jsp** : 커스텀 태그를 이용해서 목록을 출력하려는 jsp 파일로, 메인 화면이 된다.
- **ItemTagHandler.java** : 커스텀 태그 핸들러 클래스로, 사용자 속성을 전달받고 Product 클래스로부터 item 목록을 가져와 화면에 출력하는 구조로 되어 있다.
- **ItemTag.tld** : Item 태그에 대한 태그 정의 파일이다.

09. [응용실습]태그 핸들러 기반 커스텀 태그: 복합 태그 구현

2. 태그 핸들러 클래스 작성

- 태그 핸들러 클래스(ItemTagHandler.java) – 교재 p.461 ~ 462 참고
  - 표현언어에서 만들 었던 Product 빈즈 클래스의 배열 데이터를 테이블 형태로 출력해 주는 커스텀 태그
  - 배경색상과 테두리를 태그 속성으로 지정할 수 있도록 함.
  - 속성 지정을 위한 멤버변수

```
10 private String bgcolor;
11 private String border;
```

  - 태그 바디를 가져오는 부분

```
15    JspFragment body = getJspBody();
```

  - 테이블 태그 구성 (문자열 처리 성능 향상을 위해 StringBuffer 를 사용)

```
25      msg.append("<table border=")
26      .append(border)
27      .append(" bgcolor=")
28      .append(bgcolor)
29      .append(" width=150>");
30
31      out.println(msg.toString());
```

09. [응용실습]태그 핸들러 기반 커스텀 태그: 복합 태그 구현

3. 태그 라이브러리 기술자 작성

- 태그 라이브러리 기술자(ItemTag.tld) – 교재 p.463 ~ 464 참고
  - 태그 정의를 위한 tld 파일.
  - 태그 기본정보 및 속성 정보 포함.
  - 태그 바디가 단순 문자열임을 지정.

```
11 <body-content>scriptless</body-content>
```

  - 속성 지정

```
12 <attribute>
13 <name>border</name>
14 <required>true</required>
15 <rtexprvalue>true</rtexprvalue>
16 </attribute>
17 <attribute>
18 <name>bgcolor</name>
19 <required>true</required>
20 <rtexprvalue>false</rtexprvalue>
21 </attribute>
```

09. [응용실습]태그 핸들러 기반 커스텀 태그: 복합 태그 구현

4. ItemTagTest.jsp 작성

- ITemTagTest.jsp – 교재 p.465 참고
  - 앞에서 만들었던 ListItem.jsp와 동일함.
  - taglib 지시어에 따라 태그 핸들러 기반 커스텀 태그를 참조함.

```
03 <%@ taglib uri="/WEB-INF/tld/ItemTag.tld" prefix="mytag" %>
```

  - 본문에서의 태그 사용은 태그 파일과 동일함.

```
15 <mytag:item border="3" bgcolor="yello">상품 목록</mytag:item>
```

5. 실행 및 결과 확인

[그림 10-10] ItemTagTest.jsp 실행 화면

프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음