JSP [EL]과 [JSTL] 한방에 정리 + CORE

JSP파일에 자바형식의코드를 사용하면 불편한 점을 한방에 해결할 수 있는EL (Expression Language)과 JSTL (Jsp Standard Tag Library)를 이용해 코드를 간결하게 사용하는 방법을 살펴 봅니다.

EL(Expression Language)의 개념은 해석 그대로 표현 언어를 이해하고 속성 값들을 편리하게 출력하기 위해 제공된 언어이며, JSTL은 표준 액션태그로 처리하기 힘든 부분을 담당합니다.

JSP 2.0버전에서 새로 추가된 스크립트 언어인 EL(Expression Language)은<%= abc %>를 ${abc}로 간단하게 사용할 수 있게 하였고, JSTL의 Core에서 c를 이용해 <%= if%>문을 <c:if>, <%=for%>문을 <c:forEach>로 대체하여 사용합니다. 그럼 따로따로 살펴보도록 하겠습니다. 둘을 같이 묶어서 포스팅하는 이유는 제 게시판 프로젝트에 같이 사용되기 때문입니다.

EL (Expression Language)

사용목적

<%= %> , out.println()과 같은 자바코드를 더 이상 사용하지 않고 좀 더 간편하게 출력을 지원하기 위한 도구.

배열이나 컬렉션에서도 사용되고, JavaBean의 프로퍼티에서도 사용됩니다.

문법

Attribute형식에서는<%= cnt+ 1 %>를 쓰지 않고${cnt + 1}로 쓰고
Parameter형식에서는 ${param.abc}으로 씁니다.

여기서 cnt는 자바에서는 변수 이름이고, EL 식에서는Attribute의 이름으로 해석되는데요.
값을 찾을 때 Attribute는 작은 Scope에서 큰 Scope로 찾습니다.
(page → request → session → application)

[attribute란? : 메소드를 통해 저장되고 관리되는 데이터]

PageContext / Request에서 사용될 때
setAttribute("key", value) → 값을 넣는다.
getAttribute("key") → 값을 가져온다.
removeAttribue("key") → 값을 지운다.

session에서 사용될 때
set / get / remove 동일하고 추가로++
invalidate() → 값을 전부 지운다.

연산자

단어 연산자	기호 연산자
+	+
-	-
*	*
/	div
%	mod
&&	and
||	or
!	not
>	lt (less than)
<	gt (greater than)
>=	le (less or equal)
<=	ge (greater or equal)
==	eq (equal)
!=	ne (not equal)
${empty 데이터명}	hunit.tistory.com

내장객체

1) pageScope → 페이지Scope에 접근
2) request Scope → 리퀘스트Scope에 접근
3) sessionScope → 세션Scope에 접근
4) applicationScope → 어플리케이션Scope에 접근
5) param → 파라미터값 얻어올때 (1개의 Key에 1개의 Value)
6) paramValues → 파라미터값 배열로 얻어올때(1개의 Key에 여러개의 Value)
7) header → 헤더값 얻어올때 (1개의Key에 1개의 Value)
8) headerValues → 헤더값 배열로 얻어올때(1개의Key에 여러개의 Value)
9) cookie → ${cookie . key 값. value값}으로 쿠키값 조회
10) initParam → 초기 파라미터 조회
11) pageContext → 페이지컨텍스트 객체를 참조할때

paramValues 나 headerValues 사용법

① ${paramValues.boadDto[0] }
② ${paramValues["bardDto"][1] }

Values 옆에 점을 찍는 방법과 대괄호로 묶어 사용하는 2가지 방법이 있습니다.
대신 ①번에서는 인덱스가 0부터 시작하고 ②번에서는 인덱스가 1부터 시작하네요.

JSTL(Jsp Standard Tag Library)

JSTL은?

JSP는 자신만의 태그를 추가할 수 있는 기능을 제공하고 있는데요.
<jsp:include>나 <jsp:usebean>과 같은 커스텀 태그처럼 연산이나 조건문이나 반복문인 if문, for문, DB를 편하게 처리할 수 있는 것이 JSTL입니다.

태그종류

(1) Core (prefix : c)
→ 일반 프로그래밍에서 제공하는 것과 유사한 변수선언
→ 실행 흐름의 제어 기능을 제공
→ 페이지 이동 기술 제공
URI→ http://java.sun.com/jsp/jstl/core

(2) Formatting (prefix : fmt)
→ 숫자, 날짜, 시간을 포매팅하는 기능을 제공
→ 국제화, 다국어 지원 기능 제공
URI→http://java.sun.com/jsp/jstl/fmt

(3) DataBase (prefix : sql)
→ DB의 데이터를 입력 /수정 / 삭제 / 조회 하는 기능을 제공
URI→http://java.sun.com/jsp/jstl/sql

(4) XML (prefix : x)
→ XML문서를 처리할 때 필요한 기능 제공
URI→http://java.sun.com/jsp/jstl/xml

(5) Function (prefix : fn)
→ 문자열을 제공하는 함수 제공
URI→http://java.sun.com/jsp/jstl/functions

JSTL선언

```
<%@page language ="java" contentType ="text/html;charset=UTF-8"
pageEncoding ="UTF-8" isELIgnored ="false"%>

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

<html>
<head>
<meta http-equiv ="Content-Type"content ="text/html;charset=UTF-8">
<title>테스트 </title >
</head>
<body>
</body>
</html>
```

JSTL은 코드를 아래와 같이 선언합니다.
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

필요한 JSTL 태그는 prefix 와 uri 만 바꿔주면 되겠죠.
추가로 2번째 줄에 페이지를 인코딩하는 부분 마지막에 isELIgnored="false"
를 추가하면 EL을 사용할 수 있게 됩니다.

마음 같아서는 JSTL의 5가지 태그에 대해서 전부 정리해보고 싶지만, 지금은 게시판 프로젝트를 하는 도중이기 때문에 나중에 따로 게시물 하나당 잡아서 정리하도록 하겠습니다.
그러면 제 게시판 jsp파일에 들어갈 코어를 정리해볼게요.

JSTL의 Core Library

일단 JSTL의 코어 라이브러리의 태그들입니다. 자바문법 이름과 비슷해서 그리 낯설지는 않지만 처음에는 적응이 잘 안되는게 사실입니다. 익숙해지려면 여러 번 반복해서 써봐야 손에 익을 것 같아요. 태그는 위 사진과 같이 쓰시면 되지만 그래도 하나하나 살펴볼게요.

1. <c:set>
자바의 int num = 100; 을 <c:set var="num" value="100">으로 바꿔 쓴 코드입니다. 어렵지 않죠??

2. <c:out>
역시 자바의 system.out.println(" 안녕하세요 ");을 간단하게 <c:out value=" 안녕하세요 ">로 변경되었습니다. 또한 제 생각에는 장점이라고 생각하는데, 이 태그는특수문자를 그대로 출력합니다.

3. <c:remove>

한 영역의 변수명을 지우는 코드입니다. 만약에 영역을 생략할 경우 모든 영역의 변수가 삭제됩니다. 영역에는 아까 Attribute에서 정리했다 시피(page → request → session → application)순서의 영역을 가집니다.

4. <c:if>

자바의 if - else 문과 동일하지만JSTL에서는 else문이 없습니다. 여기서scope값을 생략하면 기본으로 page영역이 지정됩니다.

5. <c:choose> / <c:when> / <c:otherwise>

자바의 switch 구문과 if-else 구문을 혼합한 형태로 다수의 조건문을 걸고 싶을 때 사용합니다.

```
<c:choose >
        <c:when test ="${emptylist}">
        등록된글이없습니다.
        </c:when >

        <c:when test ="${abc}">
        안녕하세요
        </c:when >

        <c:otherwise >
                <c:set var ="doneLoop"value ="false"/>
        </c:otherwise >
</c:choose >
```

이렇게 <c:choose> 태그안에 <c:when>이 중복되어 사용이 가능하며 boolean값이 True일 경우 블록을 수행합니다. <c:otherwise>는 <c:when>의 결과 값이모두 False 일경우실행이 됩니다. 그래서 필요한 경우에만 사용됩니다.

6. <c:forEach>

자바에서는 for문으로 불리던게 JSTL에서는 forEach로 변경되었습니다. 배열이나 컬렉션, Map에 저장되어 있는 값들을 순서대로 처리 할때 사용되며,
<c:forEach var=" i " begin=" 1 " end=" 10" step=" 1 "> ${ i } </c:forEach>로 i가 1부터 10까지 1씩 증가한다는 구문을 쉽게 만들 수 있습니다.

7. <c:forTokens>

자바의 StringTokenizer 를 JSTL를 사용하면 아주 간편하게 사용할 수 있습니다.
<c:forTokens var="abc" items="안녕/하세요/hunit블로그/입니다" delims="/" >이렇게 코드를 작성할 수 있겠죠.

8. <c:catch>

```
try{
    자바에서는여기에행동
}catch(Exceptionerr){
```

```
    에러내용표시
}
<c:catch var ="abc">
    JSTL에서는여기에행동
 </c:catch >
 태그밖에 ${abc}를 사용하여 에러 내용
```

자바의 Try-catch 구문과 같죠. 단 <c:catch>태그는 에러내용을 ${abc}로 빼내서 처리해줘야 합니다. 추가로 <c:redirect>와 <c:import>, <c:url> 태그가 있지만 글이 길어진 관계로 짤막하게 정리하고 끝내도록 하겠습니다. 별로 설명할게 없다는 것도 하나의 이유지만요.

9. <c:redirect>는 아래와 같이 파라미터 값을 지정된 url로 보냅니다.

```
<c:redirect url="baordList.jsp">
    <c:param name="abc" value="안녕하세요" />
</c:redirect>
```

10. <c:import>는 <jsp:include>와 비슷합니다.

11. <c:url>은 <c:set>과 비슷하며 GET방식으로 파라미터를 전달합니다.

출처:http://hunit.tistory.com/203 [Ara Blog]