JSP EL/JSTL 정리

JSP 파일에 자바형식의 코드를 사용하면 불편한 점을 해결할 수 있는 EL (Expression Language) 과 JSTL (Jsp Standard Tag Library)에 정리하겠습니다.

EL/JSTL 은 JSP 2.0 버전에서 새로 추가된 스크립트 언어이며 EL 의 개념은 해석 그대로 표현 언어를 이해하고 속성 값들을 편리하게 출력하기 위해 제공된 언어이며, JSTL 은 표준 액션태그로 처리하기 힘든 부분을 담당합니다.

EL(Expression Language)은 <%= abc %>를 ${abc}로 간단하게 사용할 수 있게 하였고, JSTL 의 Core 에서 c 를 이용해 <%= if%>문을 <c:if>, <%=for%>문을 <c:forEach>로 대체하여 사용합니다.

■ EL (Expression Language)

▼ 사용목적

<%= %> , out.println()과 같은 자바코드를 더 이상 사용하지 않고 좀더 간편하게 출력을 지원하기 위한 도구이며 배열이나 컬렉션에서도 사용되고, JavaBean 의 프로퍼티에서도 사용됩니다.

▼ 문법

Attribute 형식에서는 ~~<%= cnt + 1 %>~~를 쓰지 않고 ${cnt + 1}로 쓰고

Parameter 형식에서는 ${param.abc}으로 씁니다.

여기서 cnt 는 자바에서는 변수 이름이고, EL 식에서는 Attribute 의 이름으로 해석됩니다.

[attribute 란? : 메소드를 통해 저장되고 관리되는 데이터]

PageContext / Request 에서 사용될때

setAttribute("key", value) → 값을 넣는다.

getAttribute("key") → 값을 가져온다.

removeAttribue("key") → 값을 지운다.

session 에서 사용될때

set / get / remove 동일하고 추가로++

invalidate( ) → 값을 전부 지운다.

값을 찾을때 Attribute 는 작은 Scope 에서 큰 Scope 로 찾습니다.

(page → request → session → application)

▼ 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}	

▼ 내장객체

1) pageScope → 페이지 Scope 에 접근

2) request Scope → 리퀘스트 Scope 에 접근

3) sessionScope → 세션 Scope 에 접근

4) applicationScope → 어플리케이션 Scope 에 접근

5) param → 파라미터값 얻어올때 (1 개의 Key 에 1 개의 Value)

6) paramValues → 파라미터값 배열로 얻어올때(1 개의 Key 에 여러개의 Value)

7) header → 헤더값 얻어올때 (1 개의 Key 에 1 개의 Value)

8) headerValues → 헤더값 배열로 얻어올때 (1 개의 Key 에 여러개의 Value)

9) cookie → ${cookie. key 값. value 값}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 방법**과 **대괄호로 묶어** 사용하는 2 가지 방법이 있습니다.

대신 ①번에서는 인덱스가 0 부터 시작하고 ②번에서는 인덱스가 1 부터 시작합니다.

JSTL (Jsp Standard Tag Library)

▼ JSTL 은?

JSP 는 자신만의 태그를 추가할 수 있는 기능을 제공하고 있으며 <jsp:include>나 <jsp:usebean>과 같은 커스텀 태그처럼 연산이나 조건문이나 반복문인 if 문, for 문, DB 를 편하게 처리할 수 있는 것이 JSTL 입니다.

▼ 태그 종류

(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)

→ DB 의 데이터를 입력 / 수정 / 삭제 / 조회 하는 기능을 제공

URI → http://java.sun.com/jsp/jstl/sql

(4) XML (prefix : x)

→ XML 문서를 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

JSTL 은 코드를 아래와 같이 선언합니다.

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

필요한 JSTL 태그는 prefix 와 uri 만 바꿔주면 됩니다.

JSTL 의 Core Library

```
1. <c:set>
<c:set var="num" value="100">
<c:set var="num" value="100" scope="page">
2. <c:out>
<c:out value="출력할 값" default="value가 null값일 경우 출력할 값">
3. <c:remove>
<c:remove var="변수명" scope="영역"/>
4. <c:if>
<c:if test="조건식" var="조건을 검사하고 return되는 bool값을 저장할 변수"
      scope="bool 변수가 사용될 범위" />
5. <c:choose>
<c:when test="조건식">
<c:otherwise>
6. <c:foreach>
<c:foreach begin="시작값" end="끝값" step="증가값" var="count값이 저장될 변수" />
<c:foreach var="변수" items="배열or컬렉션" />
7. <c:forTokens>
<c:forTokens items="배열or컬렉션" delims="구분자" var="" begin="" end="" step="" />
8. <c:catch>
try문에 해당하고 catch에 해당하는 코드는 따로 작성해야 됨
```


JSTL 의 코어 라이브러리의 태그들입니다.

1. <C:set>

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

2. <c:out>

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

3. <c:remove>

한 영역의 변수명을 지우는 코드입니다. 만약에 영역을 생략할 경우 모든 영역의 변수가 삭제됩니다.

영역에는 아까 Attribute 에서 정리했듯이 (page → request → session → application) 순서의 영역을 가집니다.

4. <c:if>

자바의 if - else 문과 동일하지만 JSTL 에서는 else 문이 없습니다. 여기서 scope 값을 생략하면 기본으로 page 영역이 지정됩니다.

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

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

```
<c:choose>
    <c:when test="${empty list }">
        등록된 글이 없습니다.
    </c:when>
    <c:when test="${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 boolean 값이 True 일 경우 블록을 수행합니다. <c:otherwise>는 <c:when>의 결과 값이 모두 False 일 경우 실행이 됩니다. 그래서 필요한 경우에만 사용됩니다.

6. <c:forEach>

자바에서는 for 문으로 불리던게 JSTL 에서는 forEach 로 변경되었습니다. 배열이나 컬렉션, Map 에 저장되어 있는 값들을 순서대로 처리 할때 사용되며, <c:forEach var=" i " begin=" 1 " end=" 10 " step=" 1 "> ${ i } </c:forEach>로 i 가 1 부터 10 까지 1 씩 증가한다는 구문을 쉽게 만들 수 있습니다.

7. <c:forTokens>

자바의 StringTokenizer 를 JSTL 를 사용하면 아주 간편하게 사용할 수 있습니다.

<c:forTokens var="abc" items="안녕/하세요/hunit 블로그/입니다" delims="/" >이렇게 코드를 작성할 수 있습니다.

8. <c:catch>

```
try{
    자바에서는 여기에 행동
  } catch (Exception err){
        에러내용 표시
   }


 <c:catch var= "abc ">
   JSTL 에서는 여기에 행동
 </c:catch>
     태그 밖에 ${abc}를 사용하여 에러내용
```

자바의 Try-catch 구문과 같죠. 단 <c:catch>태그는 에러내용을 ${abc}로 빼내서 처리해줘야 합니다.

9. <c:redirect>는 아래와 같이 파라미터 값을 지정된 url 로 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 GET 방식으로 파라미터를 전달합니다.

출처: https://hunit.tistory.com/203 [HunIT Blog]