core 라이브러리

변수 선언, 조건문, 반복문 등 간단한 프로그램 로직 구현을 대신할 수 있는 기능의 커스텀 태그들이 지원되는 라이브러리이다. JSP 에서 JSTL 의 core 라이브러리를 사용하려면 다음과 같이 taglib 지시자 태그를 정의해야 한다. uri 속성에는 정해진 URL 문자열을 사용해야 하면 prefix 는 임의로 설정할 수 있지만 일반적으로 core 의 약어인 c 를 많이 사용한다.

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

(1) <c:set> 태그

<c:set> 태그는 JSTL 태그에서 사용되는 EL 변수를 만들고 값을 설정할 때 또는 EL 변수의 프로퍼티 값을 설정할 때 사용한다. EL 변수란 EL 식에서 사용되는 변수로서 page, request, session 그리고 application 의 4 개 스코프에 저장된 객체의 이름이다. 표준 액션 태그의 <jsp:useBean> 태그와 <jsp:setProperty> 태그의 기능을 합친 것과 동일한 기능을 지원한다.

```
<c:set var="varName" value="varValue" [scope="영역"]></c:set>
<c:set var="varName" [scope="영역"]>varValue</c:set>
```

<c:set> 태그로 객체의 프로퍼티 값을 설정하는 형식

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

(2) <c:remove> 태그

<c:remove> 태그는 <c:set> 태그로 만든 EL 변수를 삭제할 때 사용하며, 구문은 다음과 같다.

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

삭제할 때 scope 를 지정하지 않으면 동일한 이름으로 저장된 모든 영역의 변수를 모두 삭제한다.

```
<c:set var="name" value="홍길동" scope="request"/>
<c:remove var="name"/>
```

(3) <c:out> 태그

<c:out> 태그는 JspWriter 객체를 이용하여 데이터를 출력할 때 사용하는 태그로서 다음과 같이 사용할 수 있다.

```
<%--방법 1 --%>
<c:out value="value" [escapeXml="(true | false)"] [default="defaltValue"]/>
<%--방법 2 --%>
```

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

escapeXml: 이 속성의 값이 true 일 경❹ 다음 표와 같이 문자를 변경한다. 생략할 경❹ 기본값은 true 이다.

문자	변환된 형태
<	&lt;
>	&gt;
&	&amp;
'	&#039;
"	&#034;

(4) <c:if> 태그

<c:if> 태그는 Java 언어의 if 문과 동일한 기능을 제공한다. else 절 기능을 지원하지 않으며 단순 if 문만을 지원한다.

```
<c:if test="조건">
...
</c:if>
```

다음과 같이 <c:if> 태그에 var 속성을 사용하면 test 속성의 계산 결과를 EL 변수로 저장할 수 있다.

```
<c:if test="<%= some condition %>" var="testResult">
...
</c:if>
```

(5) <c:choose>, <c:when>, <c:otherwise> 태그

<c:choose> 태그는 Java 의 switch 구문과 if-else 또는 if-else if 구문을 혼합한 형태로서 다수의 조건문을 하나의 블록에서 수행하고자 할 때 사용한다.

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

<c:forEach> 태그

<c:forEach> 태그는 배열, Collection 또는 Map 객체에 저장되어 있는 값들을 순차적으로 처리할 때 사용할 수 있는 태그로서, Java 의 for, do-while 등을 대신해서 사용할 수 있다.

```
<c:forEach var="i" begin="1" end="10" step="2">
    ${ i } 사용
</c:forEach>
```

var 속성에 지정한 변수 i 는 1 부터 10 까지의 값 중에서 2 씩 증가하는 1, 3, 5, 7, 9 의 값을 갖게 된다.

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

Java 의 for each 문처럼 List 와 같은 컬렉션을 items 에 사용할 수 있다. <c:forEach>를 사용하면 getter 메서드를 사용할 필요 없이 객체의 속성값을 간단하게 읽어올 수 있다. varStatus 속성은 <c:forEach> 태그를 사용하여 처리될 루프 정보를 담은 LoopTagStatus 객체를 저장할 변수명을 값으로 갖게 된다. 관련 프로퍼티는 다음과 같다..

프로퍼티명	기능
index	루프 실행에서의 현재 인덱스 값이다. (0 부터 시작)
count	루프 실행 횟수 값이다. (1 부터 시작)
begin	begin 속성이 지정된 경❹ begin 속성의 값이다.

end	End 속성이 지정된 경❹ end 속성의 값이다.
step	step 속성이 지정된 경❹ step 속성의 값이다.
first	현재 실행이 루프의 첫 번째 실행인 경❹ true 가 된다.
last	현재 실행이 루프의 마지막 실행인 경❹ true 가 된다.
current	컬렉션에서의 현재 루프에서 사용할 데이터값(객체)이다.

(6) <c:forTokens> 태그

<c:forTokens> 태그는 java.util.StringTokenizer 클래스와 동일한 기능을 제공하는 태그로서 기본 형식은 다음과 같다..

```
<c:forTokens var="token" items="문자열" delims="구분자">
      ${ token }의 사용
</c:forTokens>
```

<c:forTokens> 태그는 items 속성으로 전달받은 문자열을 구분자를 이용해서 나눈 뒤, 나누어진 각 문자열을 var 속성에 명시한 변수에 저장한다. 예를 들어, 아래 코드는 items 속성의 값을 콤마(“,”)로 나눈 “red”, “green”, “blue” 문자열을 color 변수에 차례대로 대입한다.

```
<c:forTokens var="color" items="red, green, blue" delims=",">
      ${ color }
</c:forTokens>
```

<c:forTokeans> 태그는 <c:forEach> 태그와 동일하게 begin, end, step, varStatus 속성을 제공한다. 다음은 콤마(“,”)와 점(“.”)을 구분자로 사용해서 문자열로부터 토큰을 추출하는 예이다.

```
<c:forTokens var="token"
items="빨강색,주황색.노란색.초록색,파랑색,남색.보라색" delims=",.">
      ${ token }
</c:forTokens>
```

(7) <c:import> 태그

<c:import> 태그는 특정 URL 의 결과를 읽어와 현재 위치에 삽입하거나 EL 변수에 저장할 때 사용한다.

<jsp:include>가 동일 웹 애플리케이션 내 다른 자원의 수행 결과를 포함하는 기능이라면, <c:import> 태그는 동일 웹 애플리케이션 뿐만 아니라 외부의 다른 자원을 읽어와 포함시킬 수 있게 한다.

<c:import> 태그의 기본적인 사용 형식은 다음과 같다.. var 속성, scope 속성, charEncoding 속성은 생략할 수 있다.

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

var 속성을 명시할 경❹ 해당 URL 로부터 읽어온 데이터를 var 속성에 명시한 EL 변수명에 저장한다. 이때, scope 속성은 EL 변수를 생성할 scope 를 입력한다. var 속성을 지정하지 않으면 현재 위치에 URL 로부터 읽어온 결과를 출력한다.

URL 문자열 뒤에 ? 기호와 함께 추가하는 방식의 예

```
<c:import var="weather"
   url="http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=1168064
   000">
```

<c:param>이라는 서브 태그를 사용하는 예

```
<c:import var="weather" url="http://www.kma.go.kr/wid/queryDFSRSS.jsp">
   <c:param name="zone" value="1168064000"/>
</c:import>
```

(8) <c:url> 태그

<c:url> 태그는 URL 을 생성해 주는 기능을 제공하며, 기본 사용 방법은 다음과 같다..

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

(9) <c:redirect> 태그

<c:redirect> 태그는 response.sendRedirect( )처럼 지정한 페이지로 리다이렉트 시키는 기능을 제공한다.

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

url 속성의 값이 슬래시로 시작할 경❹ 리다이렉트 되는 URL 에는 현재 콘텍스트 경로가 추가된다. 만약 다른 콘텍스트 경로에 포함된 URL 로 리다이렉트하고 싶다면 context 속성에 다른 콘텍스트의 경로를 "/xxx" 형식으로 설정한다.

(10) <c:catch> 태그

<c:catch> 태그는 발생된 예외를 EL 변수에 저장할 때 사용하는 태그로서, 다음과 같이 사용한다.

```
<c:catch
  var="exName">
  예외가 발생할 수 있는
  코드
</c:catch>
```

<c:catch> 태그 블록에서 예외가 발생할 경❹ 예외 객체는 exName 변수에 할당된다. 다음과 같이 <c:catch> 태그 블록 밖에서 exName 변수를 사용하여 예외 처리를 수행하게 구현한다.

```
<c:catch var="ex">
  name 파라미터의 값 = <%= request.getParameter("name") %><br>
  <%  if(request.getParameter("name").equals("test")){  %>
      ${param.name }은 test 이다.
<% } %>
</c:catc
      h>
<c:if test="${ ! empty ex }
  "> 예외가 발생하였습니다 :
  <br>
  ${ ex }
</c:if>
```