JSP Standard Tag Library (JSTL)

Why are the standard JSP actions and EL usually not powerful enough when it comes to implementing a complex presentation logic?

They lack basic features such as looping and storing scoped variables, and they have limited conditional logic, among many others. However, the ability to create custom tags and functions more than compensates for this drawback.

However, it would be a terrible waste of time for every developer to have to write his or her own if tags, for loop tags, and so on. This is where the JSP Standard Tag Library (JSTL) comes to rescue.

---

c:out tag

The c:out is very similar to JSP scripting expressions like <%= %> and standard JSP EL use like ${...}, but it has many advantages.

Unlike the JSP scripting, it is a regular tag, so it makes the HTML look cleaner. However, the cleaner look can be accomplished with JSP EL as well.

However, unlike JSP EL, c:out tag lets you specify a default value to output if the resulting expression evaluates to null.

It also has an attribute that allows the tag to automatically escape any XML type characters like <, >, &, “, '. If the result of some expression contains any of the characters mentioned, it might break the rendering of the HTML page. Much nastier problems could arise if a malicious user inserts some HTML markup into an input field that is output to the screen unfiltered, which is known as a Cross-Site Scripting (CSS) attack. The c:out tag avoids all these problems by providing escapeXml attribute that is set to true by default.

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

<html>
<head>
 <title>JSP Page</title>
</head>

<body>
<c:out value="<c:out> tag"/><br>
Subscription ID: <c:out value="${id}" default="None"/>
</body>
</html>
```

---

c:forEach and c:forTokens Tags

The c:forEach tag provides basic iteration, accepting many different collection types. It can also function as counter-based for loop, specifying a begin, end, and a step index.

It uses the var attribute to allow the JSP developer to specify a key with which to refer to the current element through the iteration.

The c:forTokens tag functions much like c:forEach except it is designed to iterate over a string of tokens seperated by delimiters. What is considered a delimiter is customized through its required delims attribute.

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

<%
java.util.ArrayList list = new java.util.ArrayList();
list.add("One");
list.add("Two");
list.add("Three");
session.setAttribute("list", list);
%>

<c:forEach var="item" items="${list}">
   ${item}<br>
</c:forEach>

<c:forTokens var="item" items="<Once)upon,A(Time%There..." delims="<),(%>">
   ${item}<br>
</c:forTokens>
```

---

c:if Tag

The c:if tag is a simple conditional tag that evaluates its body if the supplied condition is true. The condition is evaluated through its required attribute test.

```
<c:forEach var="i" begin="1" end="10" step="1">
   i = ${i} <c:if test="${i % 2 == 0}"> (EVEN) </c:if><br>
</c:forEach>
```

c:choose Tag

The c:choose tag is a conditional tag that establishes a context for mutually exclusive conditional operations, marked by the c:when and c:otherwise tags. These three tags work much in the same way as the standard Java switch-case-default statements.

```
<UL>
  <c:forEach var="i" begin="1" end="10" step="1">
     <LI>i = ${i}
     <c:choose>
        <c:when test="${i < 3}"> (LESS THAN 3)</c:when>
        <c:when test="${i < 6}"> (LESS THAN 6)</c:when>
        <c:when test="${i == 6}"> (EQUAL TO 6)</c:when>
        <c:otherwise> (GREATER THAN 6)</c:otherwise>
     </c:choose>
     </LI>
  </c:forEach>
</UL>
```

---

c:import Tag

The JSP framework provides a couple of ways to include content from the same container. We can use the static inclusion by employing the include directive, which includes the referenced content at page translation time.

header.html	**footer.html**	**index.jsp**
<html> <head> <title>JSP Page</title> </head> <body>	</body> </html>	<%@ include file="header.html" %> Hello World! <%@ include file="footer.html" %>

We can also use the dynamic inclusion by employing the jsp:include standard action, which includes the referenced content at request time.

```
<jsp:include page="header.html" flush="true" />
<h2>Hello World!</h2>
<jsp:include page="footer.html" flush="true" />
```

Remember that with both of these types mechanisms the included content cannot be a complete HTML page unless the including page is nothing more than the inclusion statement itself. Therefore, the included content has to provide only a snippet of HTML that can fit in with the rest of the including page's HTML markup without invalidating it.

However, what if the content we want to include resides on a different server?
The include directive as well as the jsp:include standard action cannot help help us there, but but the c:import JSTL tag can. This tag can include the content, pointed to by the required attribute url, even if the content resides on a different web server.

The default behavior of c:import is to output the included content into the including page at the place where c:import appears. However, if the optional var and the likewise optional scope attributes are specified, the included content can be saved as a scoped attribute instead. If the scope attribute is omitted, it defaults to the page scope.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>JSP Page</title>
</head>
<body>
<c:import url="one.html" var="one" />
<h2>Northeastern University</h2>
<table>
<tr>
<td>${one}</td>
<td><c:import url="two.html" /></td>
</tr>
</table>
</body>
</html>
```

---

c:redirect Tag

The c:redirect tag could be used for redirecting URLs in JSP pages. If its required url attribute specifies an absolute URL, then the c:redirect tag acts like the sendRedirectURL method of the HttpServletResponse class.

If, however, the URL specified in the url attribute is a relative url, a dynamic forward occurs, which is equivalent to the forward method of the Request dispatcher class. In this case, the browser address bar still shows the original URL and not the url of the forwarded page.

We can also nest c:param tags in the body of the the c:redirect tag to supply the URL with some encoded request parameters.

redirect.jsp	**page1.jsp**
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> <c:redirect url="page1.jsp"> <c:param name="user" value="yusuf"/> <c:param name="id">12345</c:param> </c:redirect>	<html> <head> <title>JSP Page</title> </head> <body> User: ${param.user} ID: ${param.id} </body> </html>