

CS320 Web and Internet Programming JSP Standard Tag Library (JSTL)

Chengyu Sun
California State University, Los Angeles

Custom Tags in ASP.NET

```
<p>Enter a number from 1 to 100:  
  
<asp:TextBox id="tbox1" runat="server" />  
<asp:Button Text="Submit" runat="server" />  
  
</p>
```

Custom Tags in ColdFusion

```
<cform action = "cftextinput.cfm" method="post">  
  
  <cftextinput name = "myInput"  
    message = "This field must not be blank"  
    required = "Yes" />  
  
  <input type = "Submit" value ="submit" />  
  
</cform>
```

Simple JSTL Example

```
<%@ page contentType="text/html" %>  
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
  
<html><head><title>JSTL Hello</title></head>  
<body>  
  
  <c:out value="Hello World in JSTL." />  
  
</body>  
</html>
```

taglib Directive

- **URI**
 - A unique identifier for the tag library
 - NOT a real URL
- **Prefix**
 - A short name for the tag library
 - Could be an arbitrary name

JSP Standard Tag Library (JSTL)

Library	URI	Prefix
Core	http://java.sun.com/jsp/jstl/core	c
XML Processing	http://java.sun.com/jsp/jstl/xml	x
I18N Formatting	http://java.sun.com/jsp/jstl/fmt	fmt
Database Access	http://java.sun.com/jsp/jstl/sql	sql
Functions	http://java.sun.com/jsp/jstl/functions	fn

http://download.oracle.com/docs/cd/E17802_01/products/products/jsp/jstl/1.1/docs/tldocs/index.html

JSTL Core

- ◆ Flow control
 - `<c:if>`
 - `<c:choose>`
 - `<c:when>`
 - `<c:otherwise>`
 - `<c:forEach>`
 - `<c:forEachToken>`
- ◆ Variable support
 - `<c:set>`
 - `<c:remove>`
- ◆ URL
 - `<c:url>`
 - `<c:redirect>`
 - `<c:import>`
 - `<c:param>`
- ◆ Output
 - `<c:out>`
- ◆ Exception handling
 - `<c:catch>`

JSTL Usage Examples

- ◆ Login.jsp, Members.jsp, Logout.jsp
- ◆ GuestBook with Bean, EL, and JSTL
 - GuestBookEntry.java
 - GuestBook.java
 - GuestBook.jsp
 - AddComment.jsp
 - EditCommnet.jsp

Branch Tags

```
<c:if test="{um.size == 0}">No user yet.</c:if>
```

```
<c:choose>
  <c:when test="{um.size == 0}">
    No user yet.
  </c:when>
  <c:otherwise>
    <%-- display the users --%>
  </c:otherwise>
</c:choose>
```

Loop Tags

```
<%-- iterator style --%>
<c:forEach items="{um.users}" var="user">
  {user.username} <br />
</c:forEach>

<%-- for loop style --%>
<c:forEach begin="0" end="{um.size}" step="1" var="i">
  {um.users[i].username} <br />
</c:forEach>
```

<forEachToken> → Exercise

Set and Remove Scope Variables

In Login.jsp

```
<c:set var="authorized" value="true" scope="session"/>
```

In CheckLogin.jsp

```
<c:if test="{empty sessionScope.authorized}">
  <c:redirect url="Login.jsp" />
</c:if>
```

In Logout.jsp

```
<c:remove var="authorized" scope="session" />
```

Set Bean Property

- ◆ `<jsp:setProperty>`
 - `name` attribute must be a bean declared by `<jsp:useBean>`
 - ◆ `<c:set>`
 - `target` attribute can be any object with a writable property
 - E.g. set the property `a` in bean `b` with the value of request parameter `c`
- ```
<c:set target="{b}" property="a" value="{param.c}" />
```

## URL Tags

```
<c:import url="/books.xml" var="something" />
<x:parse doc="{something}"
 var="booklist"
 scope="application" />

<c:url var="url" value="/catalog" >
 <c:param name="Add" value="{bookId}" />
</c:url>
Get book
```

## Output

```
<c:out value="100" /> ⇒ ${100}
<c:out value="{price}" /> ⇒ ${price}
```

- ◆ You want to use `<c:out>` if
  - `escapeXML=true`
  - `value` is a `Java.io.Reader` object

## Character Conversion

- ◆ When `escapeXML=true`

|   |        |
|---|--------|
| < | &lt;   |
| > | &gt;   |
| & | &amp;  |
| ' | &#039; |
| " | &#034; |

## Format Date and Time

```
<fmt:formatDate value="{date}" type="date" />
<fmt:formatDate value="{date}" type="time" />
<fmt:formatDate value="{date}" type="both" />

<fmt:formatDate value="{date}"
 pattern="yyyy-MM-dd hh:mm:ss a" />
```

See <http://download.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html> for the date formatting patterns.

## JSTL Functions

- ◆ `fn:length()`
- ◆ `fn:contains()`
- ◆ `fn:containsIgnoreCase()`
- ◆ `fn:startsWith()`
- ◆ `fn:endsWith()`
- ◆ `fn:indexOf()`
- ◆ `fn:replace()`
- ◆ `fn:trim()`
- ◆ `fn:toUpperCase()`
- ◆ `fn:toLowerCase()`
- ◆ `fn:substring()`
- ◆ `fn:substringAfter()`
- ◆ `fn:substringBefore()`
- ◆ `fn:split()`
- ◆ `fn:join()`
- ◆ `fn:escapeXML()`