

## CS320 Web and Internet Programming Custom Tag Library

Chengyu Sun  
California State University, Los Angeles

## JSTL Tags

```
<c:if test="{!cart.notEmpty}">  
  <c:forEach items="{cart.items}" var="i">  
    {i} <br />  
  </c:forEach>  
</c:if>
```

## ASP.NET Tags

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

## ColdFusion Tags

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

## JavaServer Faces (JSF) Tags

```
<h:form id="helloForm">  
  <h:outputText value="{msg.prompt}" />  
  <h:inputText value="{personBean.personName}" />  
  <h:commandButton action="greeting"  
    value="{msg.button_text}" />  
</h:form>
```

## What Do These Tags Have in Common

- ◆ Not standard HTML tags → custom tags
- ◆ Interpreted by application server, not browser
- ◆ Generate code and/or HTML text
- ◆ Simpler than code
  - Easier to learn, especially by non-programmers
  - Easier to parse, validate, and interpret by IDEs

## Using Custom Tags in JSP

- ◆ Import the tag library
  - Taglib directive
  - URI is an unique identifier of the tag library
- ◆ Reference a tag
  - Library prefix + tag name
- ◆ 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>
```

## Create Java Custom Tags

- ◆ Implement a tag
  - Using Java code
  - Using a *tag file* (basically a JSP file)
- ◆ Declare the tag in a Tag Library Descriptor (TLD) file

## Example: Add Tag

```
<cs320: add op1="10" op2="20" />
```

↓  
30

- ◆ cs320.tld
- ◆ AddTag.java
- ◆ HelloTaglib.jsp

## Tag Library Descriptor (TLD)

- ◆ A Java custom tag must belong to a *tag library*
- ◆ TLD is an XML document that contains information about a tag library and the tags in the library
- ◆ TLDs are used by application servers and development tools to validate the tags

## An Empty TLD

```
<taglib xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-jsptaglibrary_2_1.xsd"
version="2.1">

<tlib-version>1.0</tlib-version>
<short-name>Hello</short-name>

</taglib>
```

## Elements of TLD

- ◆ description
- ◆ display-name
- ◆ icon
- ◆ tlib-version
- ◆ short-name
- ◆ uri
- ◆ validator
- ◆ listener
- ◆ tag
- ◆ tag-file
- ◆ function

- Make sure the uri is unique.
- Since JSP 2.1, the order of elements in an TLD must follow the order of the elements in the TLD schema specification.

## Declaration for AddTag

```
<tag>
  <name>add</name>
  <tag-class>cs320.stu31.AddTag</tag-class>
  <body-content>empty</body-content>
  <attribute>
    <name>op1</name>
    <required>true</required>
    <rtexprvalue>true</rtexprvalue>
  </attribute>
  <attribute>
    <name>op2</name>
    <required>true</required>
    <rtexprvalue>true</rtexprvalue>
  </attribute>
</tag>
```

## Elements of <tag>

- ◆ name
- ◆ tag-class
- ◆ body-content
  - JSP, scriptless, empty, tagdependent
- ◆ attribute
  - name
  - required
  - *rtexprvalue??*

## About TLD

- ◆ A TLD file must have the `.tld` suffix
- ◆ Locations for TLD
  - WAR or unpackaged: `WEB-INF` or its subdirectories, *except* `WEB-INF/classes` and `WEB-INF/lib`
  - JAR: `META-INF` or its subdirectories
- ◆ More information about TLD
  - JSP Specification
  - Chapter 15, J2EE 1.4 Tutorial

## AddTag.java ...

```
package cs320.tag;

import java.io.IOException;

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.SimpleTagSupport;

public class AddTag extends SimpleTagSupport {

    int op1, op2;

    public AddTag()
    {
        op1 = op2 = 0;
    }
}
```

## ... AddTag.java

```
public void setOp1( int i )
{
    op1 = i;
}

public void setOp2( int i )
{
    op2 = i;
}

public void doTag() throws JspException, IOException
{
    JspWriter out = getJspContext().getOut();
    out.print( op1+op2 );
}
}
```

## HelloTaglib.jsp

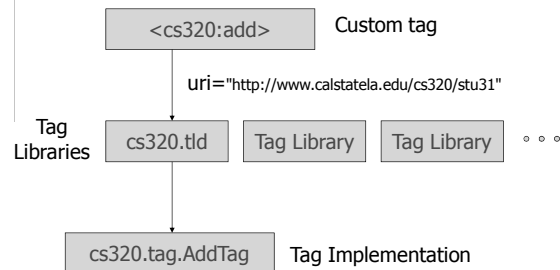
```
<%@ page contentType="text/html" %>
<%@ taglib prefix="cs320"
    uri="http://www.calstatela.edu/cs320/stu31" %>

<html><head><title>Hello Taglib</title></head>
<body>

The sum of 1 and 12 is: <cs320:add op1="1" op2="12" />

</body>
</html>
```

## How Does Server Find a Tag Implementation



## A Closer Look at SimpleTagSupport

◆ [http://docs.oracle.com/cd/E17802\\_01/products/products/jsp/2.1/docs/jsp-2\\_1-pfd2/javax/servlet/jsp/tagext/SimpleTagSupport.html](http://docs.oracle.com/cd/E17802_01/products/products/jsp/2.1/docs/jsp-2_1-pfd2/javax/servlet/jsp/tagext/SimpleTagSupport.html)

- doTag()
- getJspContext()
- getJspBody()

## Example: RequestInfo Tag

```
<cs320:requestInfo type="method" />
```

↓  
GET

- ◆ Display request information
- ◆ JspContext → PageContext → Request

## Example: Cap Tag

```
<cs320:cap>Hello World</cap>
```

↓  
HELLO WORLD

- ◆ Capitalize the body content of a tag
- ◆ getJspBody() → JspFragment → *invoke(java.io.Writer) ??*

## Tag Files

- ◆ A way to create a custom tag without writing Java code
- ◆ Tag files – *must have .tag suffix*
  - WAR or unpackaged: WEB-INF/tags or its subdirectories
  - JAR: META-INF/tags or its subdirectories
- ◆ Tag file in TLD
  - <tag-file>
    - <name>sometag</name>
    - <path>/WEB-INF/tags/sometag.tag</path>

## Example: Greeting Tag

```
<cs320:greeting name="cysun">Hello</cs320:greeting>
```

↓  
Hello, cysun!

- ◆ greeting.tag
  - A JSP file
  - No page directive
  - tag and attribute directives

## EL Functions

- ◆ Just like the *function* library in JSTL
- ◆ Any static methods can be used as EL functions
- ◆ EL function in TLD
  - `<function>`
    - ◆ `<name>`
    - ◆ `<function-class>`
    - ◆ `<function-signature>`

## Example: Leet Talk

```
#{cs320:leetTalk("fear my mad programming skills")}
```



```
ph34r my m4d pr0gr4mming zki11z!
```

- ◆ `cs320.tag.Functions`
  - `String leetTalk( String )`

## Custom Tag vs. Java Bean

- |  |   |
|--|---|
| <ul style="list-style-type: none"><li>◆ Custom tags are for the <i>web tier</i><ul style="list-style-type: none"><li>■ Access to HTTP objects<ul style="list-style-type: none"><li>◆ request, response, context ...</li></ul></li><li>■ Good for tasks related to presentation</li><li>■ One scope<ul style="list-style-type: none"><li>◆ page</li></ul></li></ul></li></ul> | <ul style="list-style-type: none"><li>◆ Beans are for the <i>business tier</i><ul style="list-style-type: none"><li>■ No access to HTTP objects<ul style="list-style-type: none"><li>◆ Plain Old Java Object (POJO)</li></ul></li><li>■ Good for tasks related to processing</li><li>■ Four scopes<ul style="list-style-type: none"><li>◆ good for sharing data</li></ul></li></ul></li></ul> |
|--|---|

## More Tag Lib Examples

- ◆ Tomcat JSP Examples - <http://cs3.calstatela.edu:8080/examples/jsp/>
- ◆ JSTL Source code from <http://jakarta.apache.org/>