CS320 Web and Internet Programming
Custom Tag Library

Chengyu Sun
California State University, Los Angeles

A 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

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>
```

NOTE: make sure your taglib uri is unique, e.g. have stu## somewhere in the uri.

AddTag.java ...

```
package cs320.stu31;

import java.io.*;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;

public class AddTag extends SimpleTagSupport {

      int op1, op2;

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

... AddTag.java

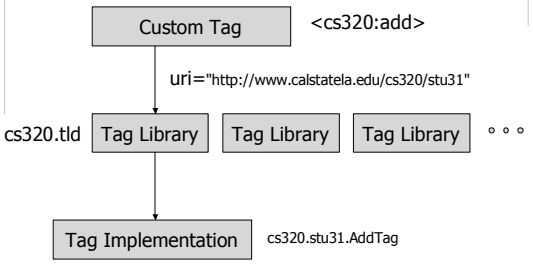
```
   public void setOp1( int i )
   {
      op1 = i;
   }

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

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

} // end of class AddTag
```

cs320.tld

```
<?xml version="1.0" ?>
<taglib>
   <uri>http://www.calstatela.edu/cs320/stu31</uri>
   <tlibversion>1.0</tlibversion>
   <jspversion>2.0</jspversion>
   <shortname>CS320 Custom Tag Examples</shortname>
   <info> Custom tag samples for CS320</info>
   <tag>
      <name>add</name>
      <tagclass>cs320.stu31.AddTag</tagclass>
      <bodycontent>empty</bodycontent>
      <attribute><name>op1</name></attribute>
      <attribute><name>op2</name></attribute>
   </tag>
</taglib>
```

JSP 2.0 Specification, Appendix JSP.C

From Tags to Their Implementations



Deploy Custom Tag Libraries

- Compile tag handler classes
  - Include jsp-api.jar in the classpath
  - On CS server: */home/tomcat/tomcat/common/lib/jsp-api.jar*
- Class files and JSP files
  - in their rightful places
- TLD files – *must have .tld suffix*
  - WAR or unpackaged: `WEB-INF` or its subdirectories, *except* `WEB-INF/classes` and `WEB-INF/lib`
  - JAR: `META-INF` or its subdirectories

More About Deploying TLDs

- Tomcat must be restarted to recognize a new TLD, even if the context is reloadable
- Changes to TLDs will be automatically reloaded, if the context is reloadable
- *Please place a dummy TLD under your WEB-INF directory on the CS server by Thursday*
  - *Make sure you have a unique URI*

More Tag Lib Examples

- A tag that accesses implicit objects
  - <cs320:log>
- A tag with body content
  - <cs320:cap>
- Repeat tag
  - http://cs.calstatela.edu:8080/jsp-examples/
  - page context attribute → page scope variable
- EL functions
- Tag files

A Closer Look at SimpleTagSupport

- http://jakarta.apache.org/tomcat/tomcat-5.0-doc/jspapi/index.html
- doTag() – to be overridden
- getJspContext() – access implicit objects
- getJspBody() – access body content

getJspContext()

- JspContext
  - getOut()
  - access to scoped attributes (variables)
- *In servlet/JSP environment, JspContext can be cast to PageContext*
- PageContext
  - access to request, response, session, servlet context ...

<cs320:log>

- Log the ip and timestamp of a HTTP request
  - LogTag.java
  - LogTag.jsp

getJspBody()

- JspFragment
  - invoke(java.io.Writer out) ??

<cs320:cap>

- Capitalize the body content of the tag
  - e.g. <cs320:cap>shout</cs320:cap> ➔ SHOUT

cs320.tld:

```
<tag>
  <name>cap</name>
  <tagclass>cs320.stu31.CapTag</tagclass>
  <bodycontent>scriptless</bodycontent>
</tag>
```

EL Functions

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

<cs320:leet>

${cs320:leet("fear my mad programming skills")}

⇩

ph34r my m4d pr0gr4mming zki11z!

- cs320.stu31.LeetTalk
  - String leet(String)

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>

<cs320:greeting>

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

Hello, cysun!

- greeting.tag
  - Just a JSP file
  - No *page* directive
  - tag directives

Custom Tags vs. Bean

- Custom tags are for the *web tier*
  - Access to HTTP objects
    - request, response, context ...
  - Good for tasks related to presentation
  - One scope
    - page
- Beans are for the *business tier*
  - No access to HTTP objects
    - Plain Old Java Object (POJO)
  - Good for tasks related to processing
  - Four scopes
    - good for sharing data

Even More Tag Lib Examples

- JSTL Source code from http://jakarta.apache.org/