Session 14

JSP Custom Tags

1

Reading & Reference

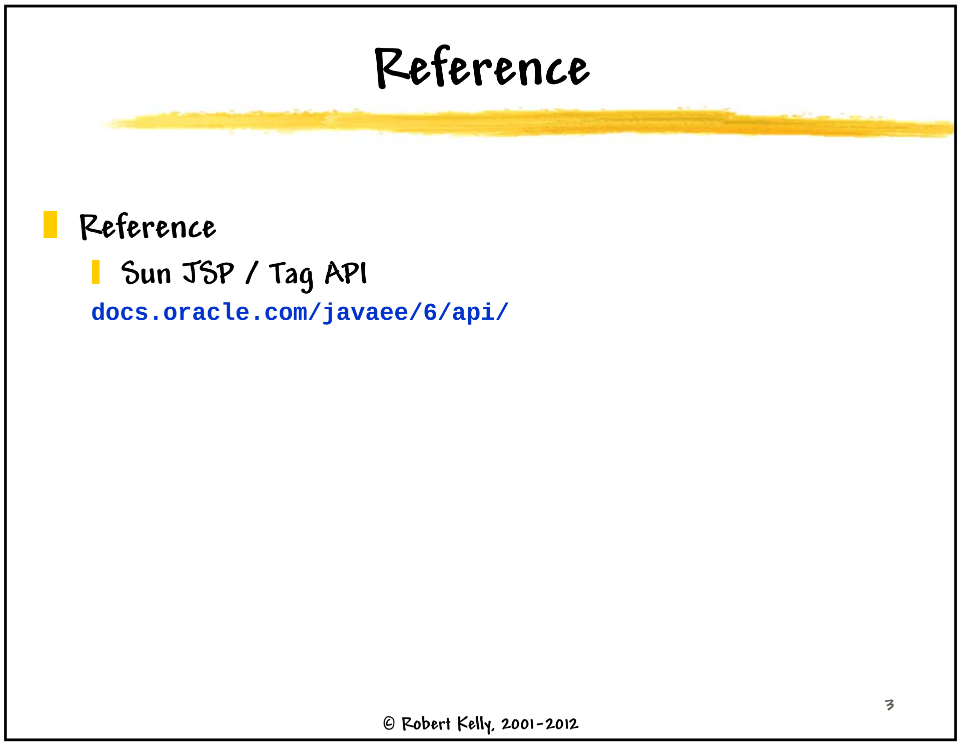
Note that Head First covers both the JSP 2.0 approach and the "classic" approach – this lecture covers mostly the JSP 2.0 approach

- Reading
  - Head First -
    - Pages 476-490
    - Chapter 10 (pages 512-528, 576-577)
  - Java EE Tutorial - Chap 13 http://docs.oracle.com/javaee/6/tutorial/doc/
  - Discussion of JSP 2.0 approach (includes class example) http://www.sitepoint.com/article/jsp-2-simple-tags

Note that the Java EE tutorial contains the most recent documentation, but refers to JSF, which we cover later in the course

2

© Robert Kelly, 2001-2012



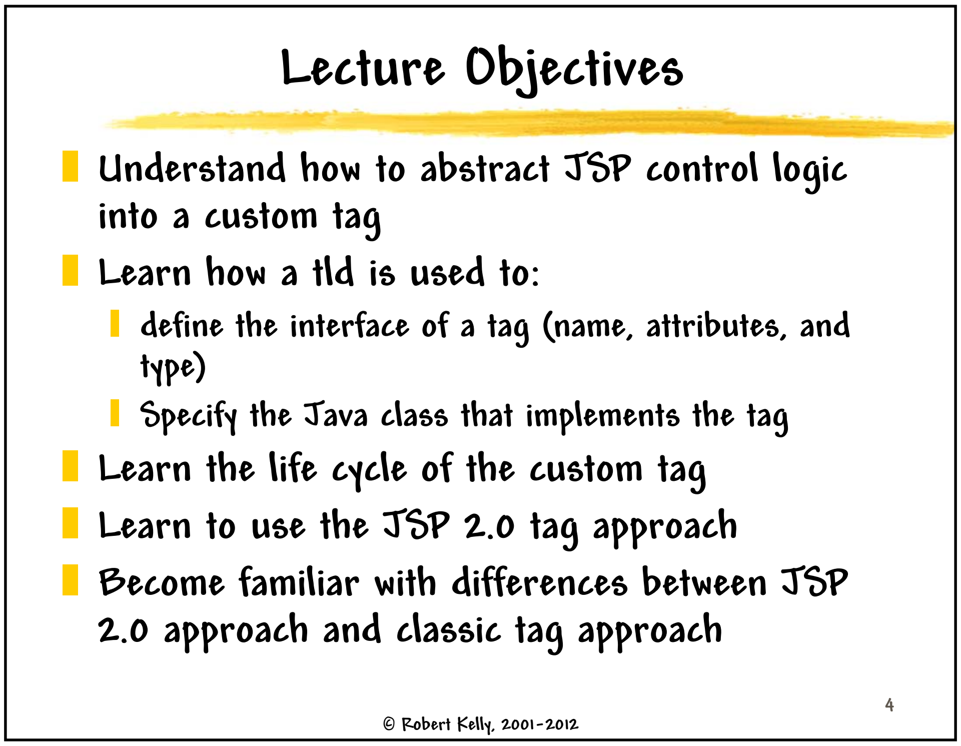
Reference

- Reference
  - Sun JSP / Tag API

  docs.oracle.com/javaee/6/api/

© Robert Kelly, 2001-2012

3



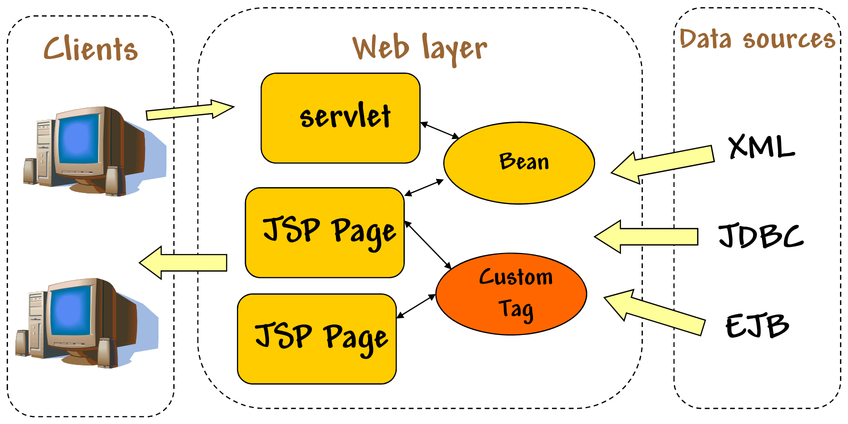
Lecture Objectives

- Understand how to abstract JSP control logic into a custom tag
- Learn how a tld is used to:
  - define the interface of a tag (name, attributes, and type)
  - Specify the Java class that implements the tag
- Learn the life cycle of the custom tag
- Learn to use the JSP 2.0 tag approach
- Become familiar with differences between JSP 2.0 approach and classic tag approach

© Robert Kelly, 2001-2012

4

Web Architecture



© Robert Kelly, 2001-2012

5

Recall the Bean Viewer

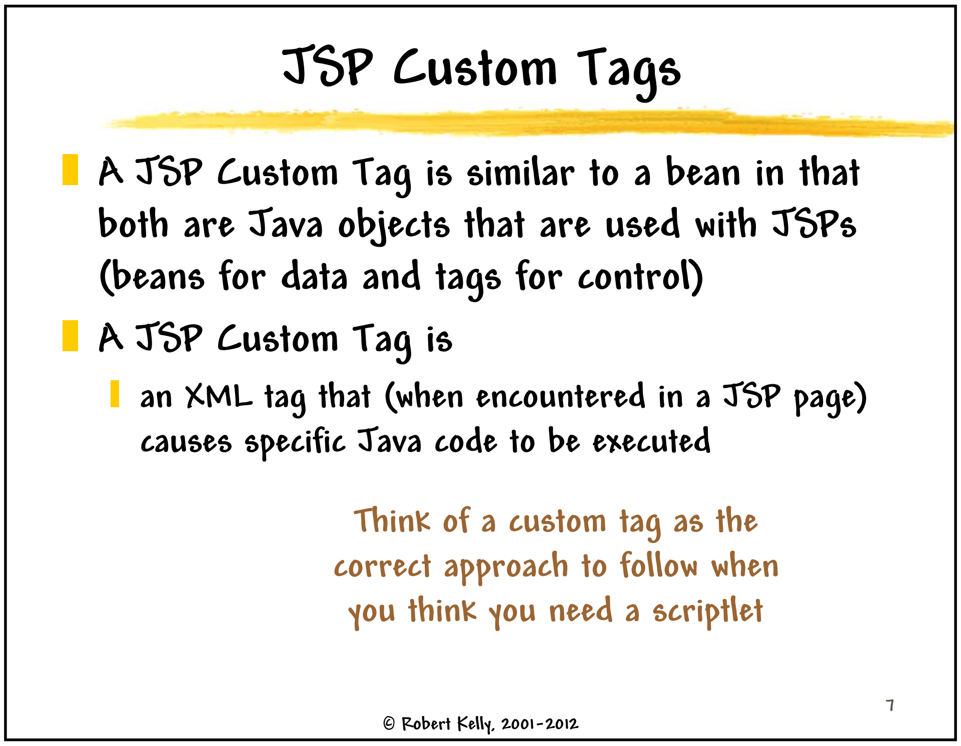
```
<h3>Bean Viewer</h3>
  <%@ page import="java.lang.reflect.*" %>
  <jsp:useBean id="bean" class="lectures.CountBean3" />
  <jsp:setProperty name="bean" property="count" value="0" />
  <%
    Class b = Class.forName("lectures.CountBean3");
    Field[] fields = b.getDeclaredFields();
    for (Field f : fields) {
      String name = f.getName();
      out.println("<p>");
      String camelName = "get" +
        (name.substring(0,1).toUpperCase() + name.substring(1));
      Method m = b.getMethod(camelName);
      out.println("Bean instance variable - " + name + ":");
      out.println(m.invoke(bean));
      out.println("</p>");
    }
%>
```

Not a good approach to use servlets in a JSP

We prefer to abstract the servlet code into a separate method, callable from the JSP

© Robert Kelly, 2001-2012

6

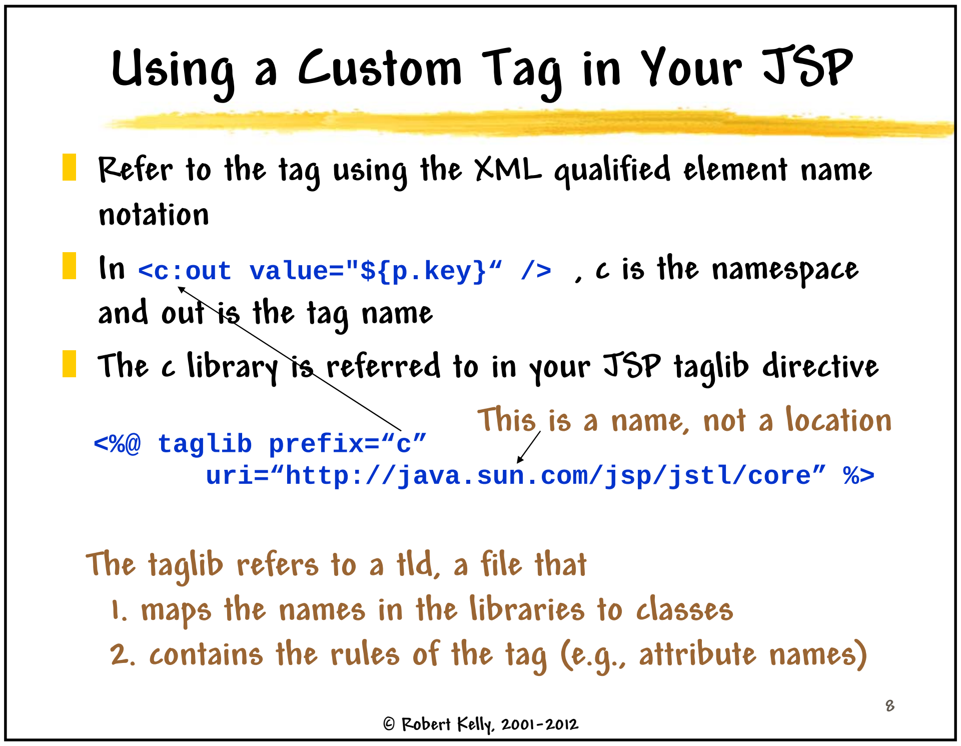


JSP Custom Tags

- A JSP Custom Tag is similar to a bean in that both are Java objects that are used with JSPs (beans for data and tags for control)
- A JSP Custom Tag is
  - an XML tag that (when encountered in a JSP page) causes specific Java code to be executed

Think of a custom tag as the correct approach to follow when you think you need a scriptlet

© Robert Kelly, 2001-2012 7



Using a Custom Tag in Your JSP

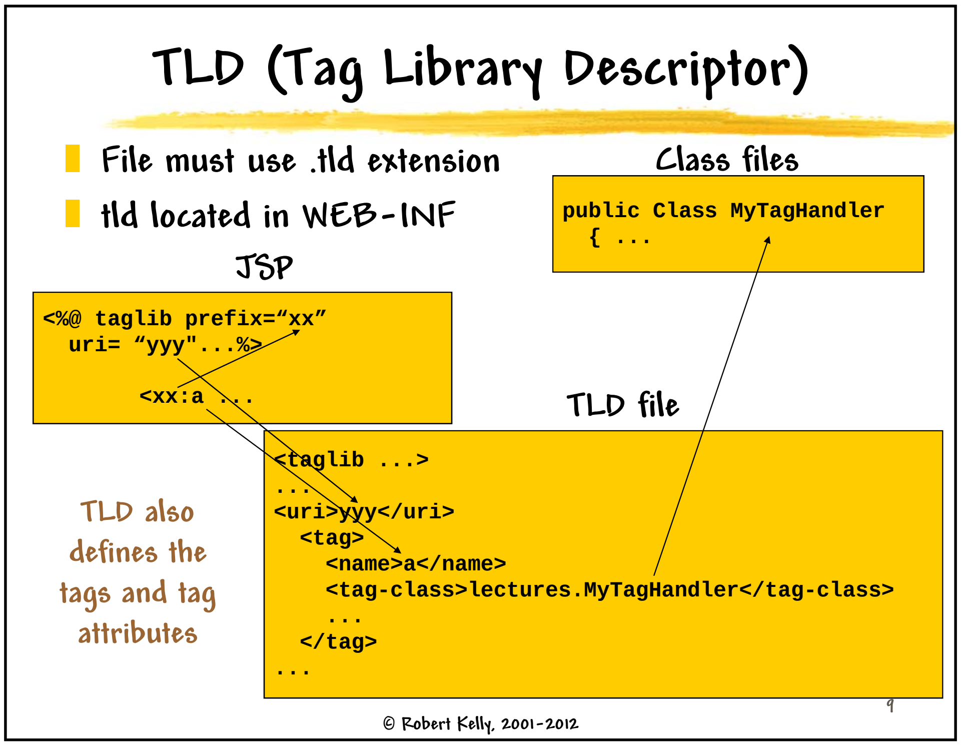
- Refer to the tag using the XML qualified element name notation
- In `<c:out value="${p.key}“ />` , c is the namespace and out is the tag name
- The c library is referred to in your JSP taglib directive

This is a name, not a location

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

The taglib refers to a tld, a file that
1. maps the names in the libraries to classes
2. contains the rules of the tag (e.g., attribute names)

© Robert Kelly, 2001-2012 8



TLD (Tag Library Descriptor)

- File must use .tld extension
- tld located in WEB-INF

JSP

```
<%@ taglib prefix="xx"
  uri= "yyy"...%>

       <xx:a  ...
```

Class files

```
public Class MyTagHandler
  { ...
```

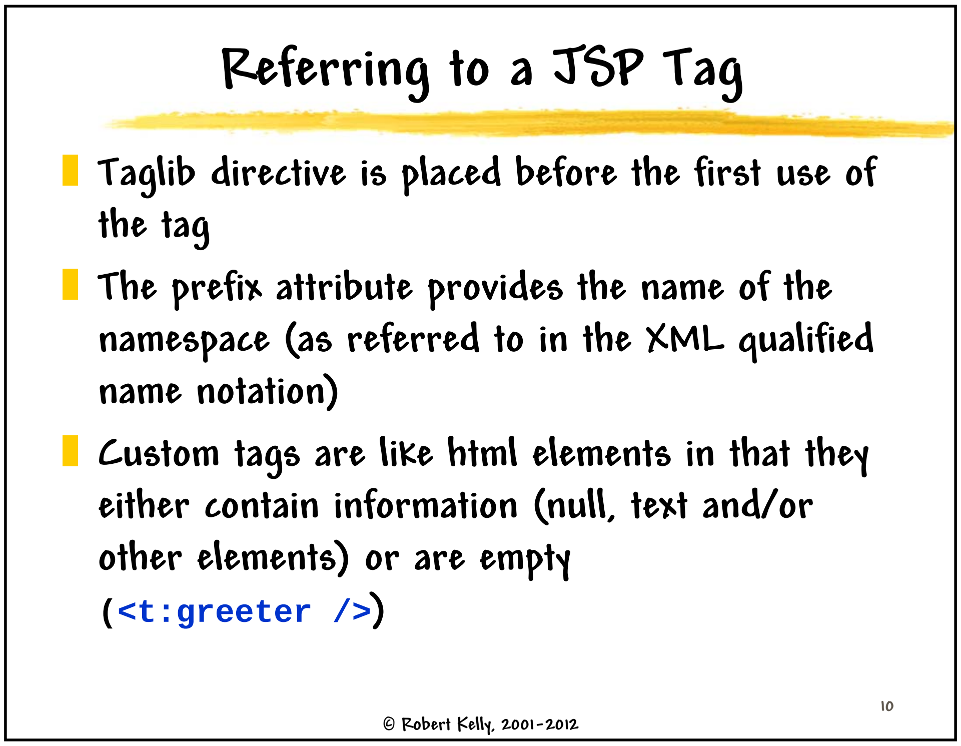
TLD file

```
<taglib ...>
...
<uri>yyy</uri>
  <tag>
    <name>a</name>
    <tag-class>lectures.MyTagHandler</tag-class>
    ...
  </tag>
...
```

TLD also defines the tags and tag attributes

9

© Robert Kelly, 2001-2012

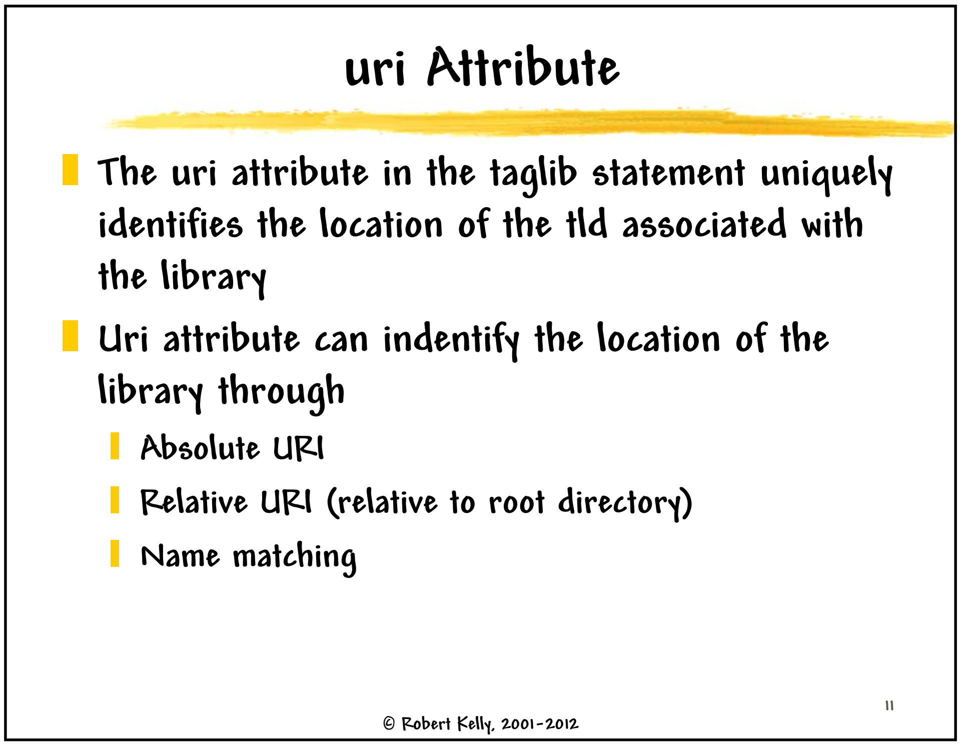


Referring to a JSP Tag

- Taglib directive is placed before the first use of the tag
- The prefix attribute provides the name of the namespace (as referred to in the XML qualified name notation)
- Custom tags are like html elements in that they either contain information (null, text and/or other elements) or are empty (`<t:greeter />`)

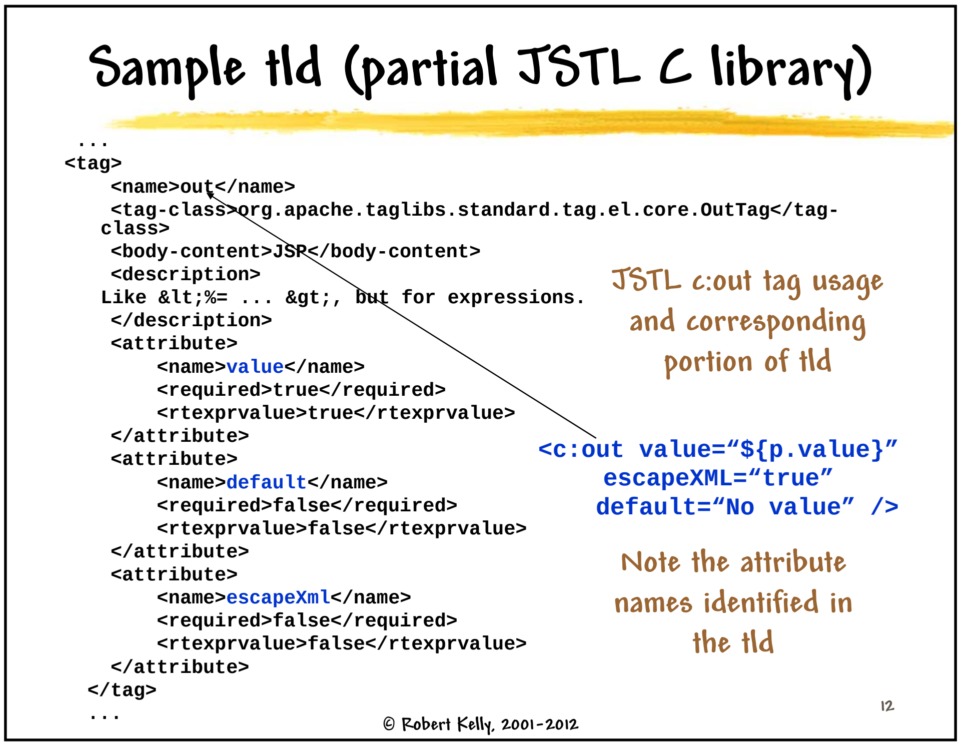
10

© Robert Kelly, 2001-2012



uri Attribute

- The uri attribute in the taglib statement uniquely identifies the location of the tld associated with the library
- Uri attribute can indentify the location of the library through
  - Absolute URI
  - Relative URI (relative to root directory)
  - Name matching



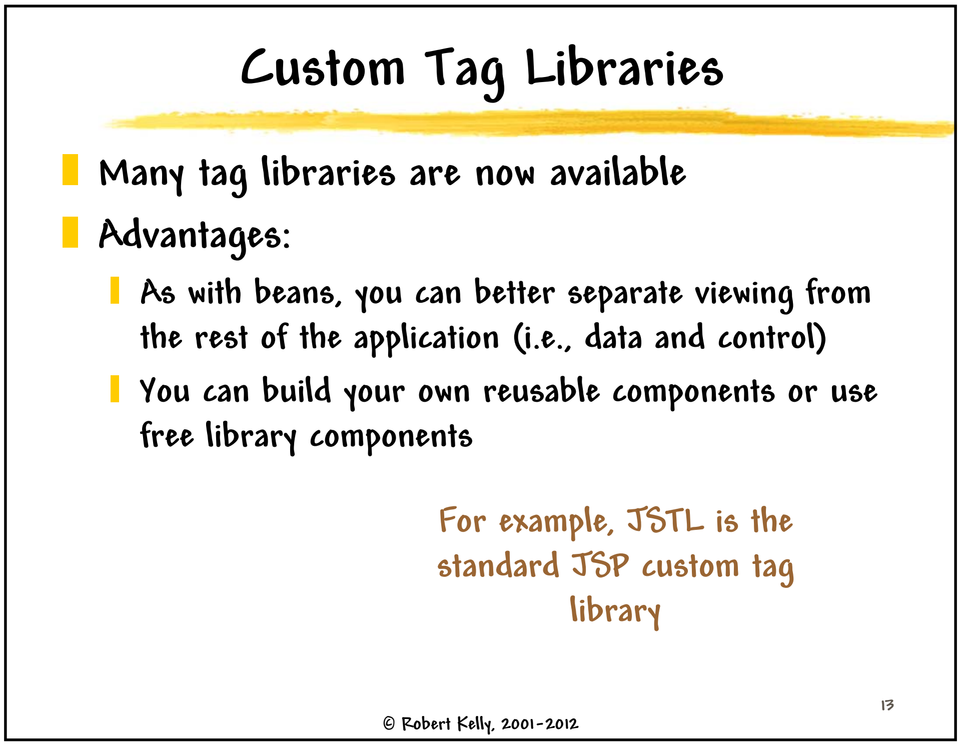
Sample tld (partial JSTL C library)

```
 ...
<tag>
    <name>out</name>
    <tag-class>org.apache.taglibs.standard.tag.el.core.OutTag</tag-
   class>
    <body-content>JSP</body-content>
    <description>
   Like &lt;%= ... &gt;, but for expressions.
    </description>
    <attribute>
        <name>value</name>
        <required>true</required>
        <rtexprvalue>true</rtexprvalue>
    </attribute>
    <attribute>
        <name>default</name>
        <required>false</required>
        <rtexprvalue>false</rtexprvalue>
    </attribute>
    <attribute>
        <name>escapeXml</name>
        <required>false</required>
        <rtexprvalue>false</rtexprvalue>
    </attribute>
  </tag>
  ...
```

JSTL c:out tag usage and corresponding portion of tld

```
<c:out value="${p.value}"
     escapeXML="true"
     default="No value" />
```

Note the attribute names identified in the tld



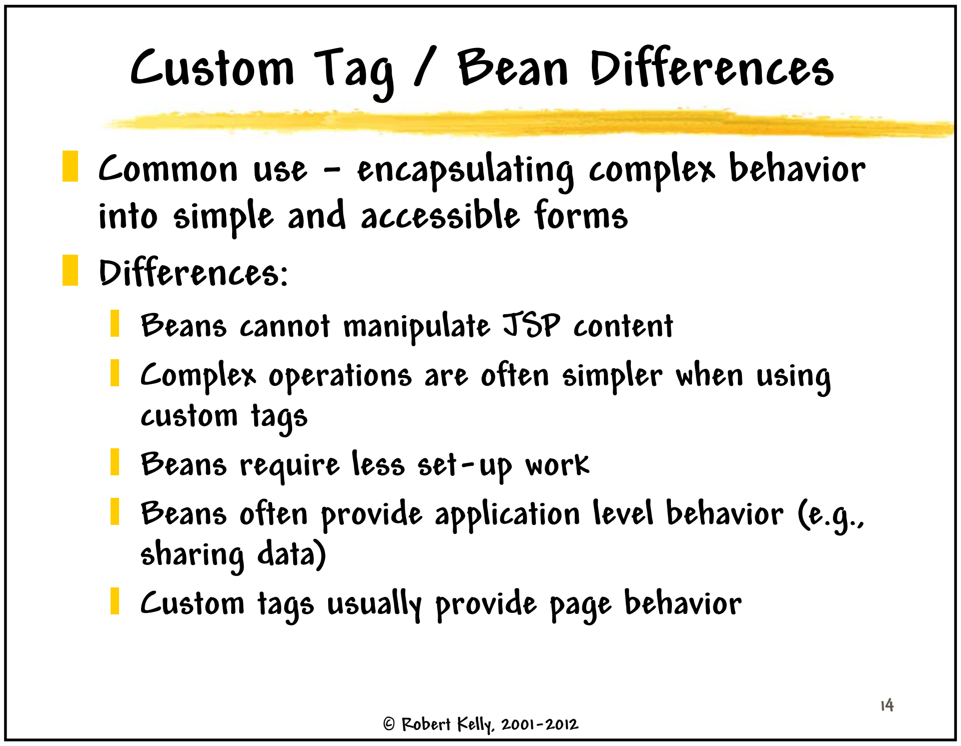
Custom Tag Libraries

- Many tag libraries are now available
- Advantages:
  - As with beans, you can better separate viewing from the rest of the application (i.e., data and control)
  - You can build your own reusable components or use free library components

For example, JSTL is the standard JSP custom tag library

© Robert Kelly, 2001-2012

13



Custom Tag / Bean Differences

- Common use – encapsulating complex behavior into simple and accessible forms
- Differences:
  - Beans cannot manipulate JSP content
  - Complex operations are often simpler when using custom tags
  - Beans require less set-up work
  - Beans often provide application level behavior (e.g., sharing data)
  - Custom tags usually provide page behavior

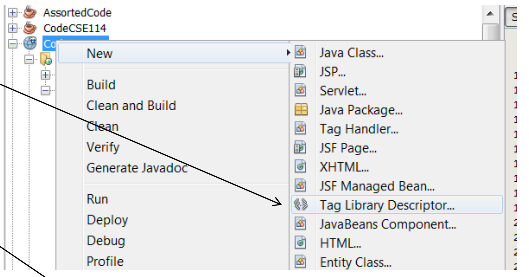
© Robert Kelly, 2001-2012

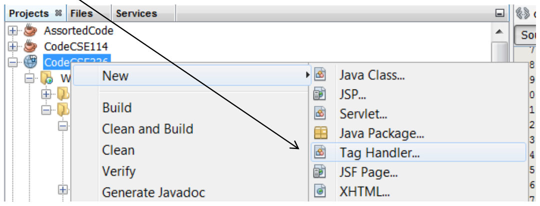
14

Creating a New Tag

- Create the tld first
- Create the tag handler
- Include the taglib statement in your JSP
- Write the tag in your JSP

Note that NetBeans may show an error in JSP when it is correct





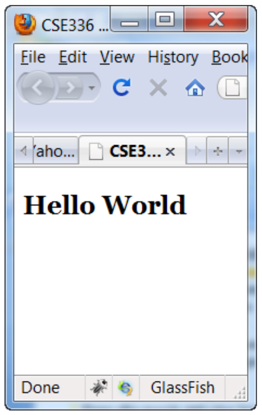
Example - Hello World (JSP 2.0)

```
<%@taglib prefix="t"
   uri= "CSE336-lectures" %>
<?xml version="1.0"
   encoding="utf-8"?>


<html>
  <head>
   <title>
     CSE336 Custom Tag Example
   </title>
  </head>
  <body>
    <h2><t:greeter1 /></h2>
  </body>
</html>
```

Tag library prefix is t

The greeter tag evaluates to "Hello World"



lectures.tld

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.1"
   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">
  <tlib-version>1.0</tlib-version>
   <short-name>lectures</short-name>
  <uri>CSE336-lectures</uri>
<tag>
    <name>greeter1</name>
    <tag-class>lectures.Greeter1</tag-class>
    <body-content>empty</body-content>
  </tag>
</taglib>
```

Matches the uri attribute value in JSP

This is an empty tag (i.e., no enclosed text or tags)

Possible values of body-content are:
1) empty, 2) scriptless (means no scriptlets and JSP expressions), and 3) tagdependent (treats body as plain text)

17

Greeter1.java

```
package lectures;

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

public class Greeter1 extends SimpleTagSupport {
  public void doTag() throws JspException {
    PageContext pageContext = (PageContext) getJspContext();
    JspWriter out = pageContext.getOut();
    try {
      out.println("Hello World");
    } catch (Exception e) {
      // Ignore.
    }
  }
}
```

SimpleTagSupport is a support class that implements the SimpleTag interface

18

SimpleTag Interface

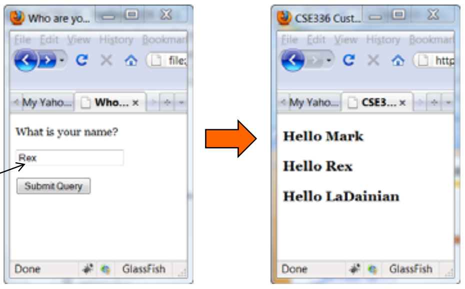
- doTag – called by the container to invoke the tag
- Other methods to access the JSP context
  - setJspContext() is called by the container prior to doTag(), and makes the current JSP context information available via getJspContext()

© Robert Kelly, 2001-2012

19

Example - Tags with Attributes

```
<%@taglib prefix="t"
   uri="CSE336-lectures“ %>
<html>
  <head>
    <meta http-equiv="Content-Type"
       content="text/html;
   charset=UTF-8">
    <title>CSE336 Custom Tag
   Example</title>
  </head>
  <body>
    <h3>
    <t:greeterName name=“Mark" />
    </h3>
    <h3>
      <t:greeterName
        name="${param['name']}"/>
    </h3>
    <h3> <t:greeterName /> </h3>
  </body>
</html>
```



What is your name?

Rex

Submit Query

Hello Mark

Hello Rex

Hello LaDainian

Notice in one case the attribute is an EL expression

© Robert Kelly, 2001-2012

20

lectures.tld

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.0" xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-
   jsptaglibrary_2_0.xsd">
  <tlib-version>1.0</tlib-version>
  <short-name>demo</short-name>
  <uri>CSE336-lectures</uri>
...
<tag>
   <name>greeterName</name>
   <tag-class>lectures.GreeterName</tag-class>
   <body-content>empty</body-content>
   <attribute>
       <name>name</name>
       <rtexprvalue>true</rtexprvalue>
       <required>false</required>
    </attribute>
 </tag>
</taglib>
```

GreeterName.java

```
public class GreeterName extends SimpleTagSupport {

  private String defaultName = “LaDainian";
  private String name;
  public void setName(String name){
       this.name = name;
       if name.equals(“”) name=defaultName;
   }
  public void doTag() throws JspException {
    PageContext pageContext = (PageContext) getJspContext();
    JspWriter out = pageContext.getOut();
    try {
      out.println("Hello " + name);
    } catch (Exception e) {
      // Ignore.
    }
  }
}
```

Your tag handler contains a setter method for each attribute in the tag – this is how you pass the "parameters" to your tag handler method

Deployment

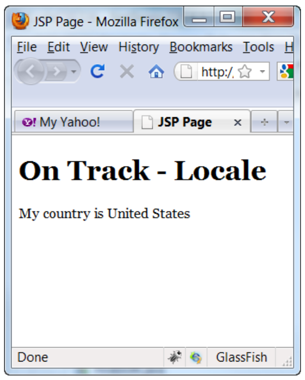
- tld file is placed in the WEB-INF directory (or a subdirectory of WEB-INF)
- Tag handler classes are placed in the WEB-INF/classes directory

© Robert Kelly, 2001-2012

23

Are We on Track? ...

```
<h1>On Track - Locale</h1>
<cse336:locale country="us" />
```


JSP Page - Mozilla Firefox
File Edit View History Bookmarks Tools H
http:/.
My Yahoo!
JSP Page
Done
GlassFish

On Track - Locale

My country is United States

© Robert Kelly, 2001-2012

24

… Are We on Track?

- Create a custom tag that will display information about a given locale (name your tag handler class "MyLocale")
- In your NetBeans Tag Handler wizard
  - Select "empty" for Body Content
  - Remember to specify the country attribute (NetBeans will add the setCountry method to your tag handler)
- Replace the entire try block in your tag handler with the code to display the country information
- The Locale class is in the java.util package
- Use English as the language when you construct the MyLocale object in your tag handler

You might see an error message in your JSP stating that NetBeans cannot load the tag handler class – everything should be OK, this appears to be a NetBeans error

25

© Robert Kelly, 2001-2012

Were We on Track? (Tag Handler)

```
public class MyLocale extends SimpleTagSupport {

    private String country;
    private Locale l;

   public void doTag() throws JspException {
        JspWriter out = getJspContext().getOut();
        out.println("<p>");
        out.println("My country is " + l.getDisplayCountry());
        out.println("</p>");
   }

    public void setCountry(String country) {
        this.country = country;
        l = new Locale("en", country);
    }
}
```

26

© Robert Kelly, 2001-2012

Were We on Track? (tld)

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.0" xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-
   jsptaglibrary_2_0.xsd">
  <tlib-version>1.0</tlib-version>
  <short-name>cse336</short-name>
  <uri>CSE336-lectures</uri>
  <tag>
    <name>locale</name>
    <tag-class>lectures.MyLocale</tag-class>
    <body-content>empty</body-content>
    <attribute>
      <name>country</name>
      <rtexprvalue>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
  </tag>
</taglib>
```

How Do the Tag Handler & Container Interact?

- For your tag handler to be powerful, it needs to interact with the Web Container.
  - Access data in the JSP page
  - Control execution of the JSP that called the tag handler
  - Cause evaluation of the tag body, if needed
- JSP 2.0 and classic tag hander approaches are different

Which Tag Approach to Use

- Use classic if your container does not support JSP 2.0
- Use classic if you are maintaining classic code
- Use JSP 2.0 otherwise (for almost all possible requirements)

Extra benefit:
JSP 2.0 approach is a little simpler

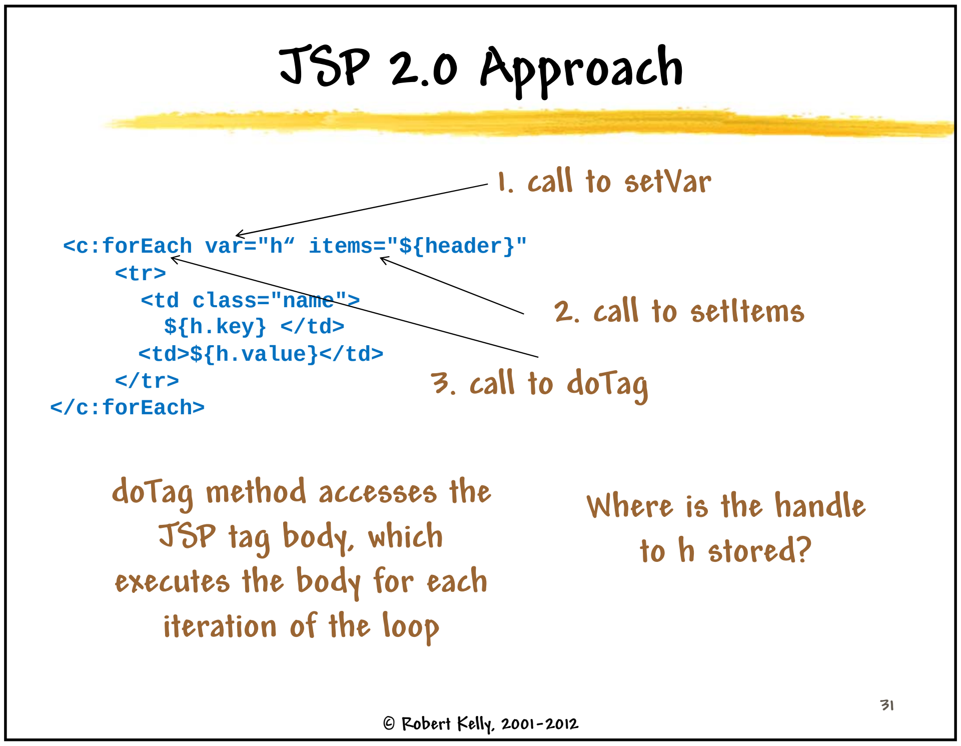
JSP Container / Tag Handler Interaction

Classic

- JSP Container calls the setter method for attributes
- Tag handler accesses the PageContext
- JSP Container invokes the doStartTag and doEndTag methods
- Tag handler provides a return parameter instructing the JSP contain what to do next

JSP 2.0

- JSP Container calls the setter method for attributes
- Tag handler accesses the PageContext (or its parent class, JspContext)
- JSP Container invokes the doTag methods
- Tag handler accesses the JSP tag body
- Tag handler throws an exception that the JSP container handles



JSP 2.0 Approach

1. call to setVar

```
<c:forEach var="h“ items="${header}"
    <tr>
      <td class="name">
        ${h.key} </td>
      <td>${h.value}</td>
    </tr>
</c:forEach>
```

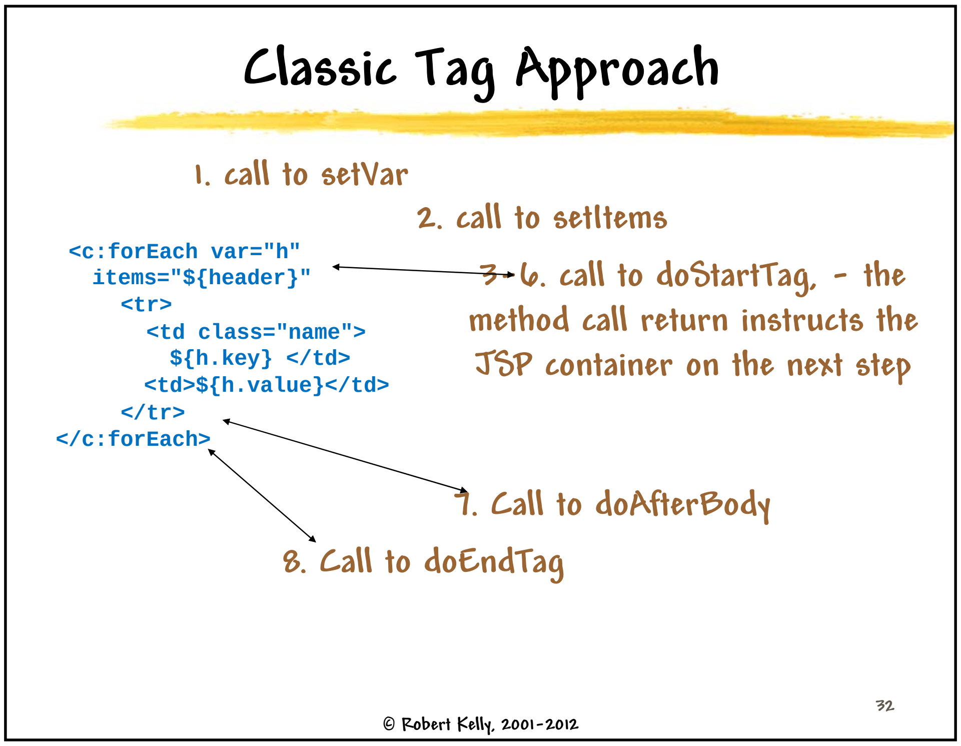
2. call to setItems

3. call to doTag

doTag method accesses the JSP tag body, which executes the body for each iteration of the loop

Where is the handle to h stored?

31



Classic Tag Approach

1. call to setVar

2. call to setItems

```
<c:forEach var="h"
  items="${header}"
    <tr>
      <td class="name">
        ${h.key} </td>
      <td>${h.value}</td>
    </tr>
</c:forEach>
```

~~3~~6. call to doStartTag, - the method call return instructs the JSP container on the next step

7. Call to doAfterBody

8. Call to doEndTag

32

Access to Shared Objects

- Your custom tag may need to access some shared object
- You can obtain a handle to many of these objects through the PageContext object, which is available in the TagSupport (Classic) class as an instance variable
- Example (getting the request object):

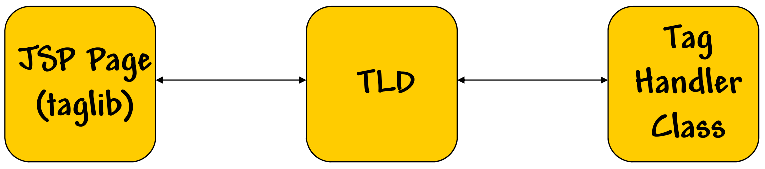
```
ServletRequest req = pageContext.getRequest();
```

In JSP 2.0, the shared objects are available through the JspContext

33

Building Your Own Tag Library

- Tag Library Descriptor (TLD) file – maps XML element names to the tag implementations
- With servlets, we used the web.xml file to associate a URL string to a Java class
- With custom tags, we use a TLD to perform an association of a JSP element name to a class



34

JSP 2.0 Tag Example

- Let's re-code our JSP Form Tester so that we are not using the JSTL c:forEach tag

Previous approach

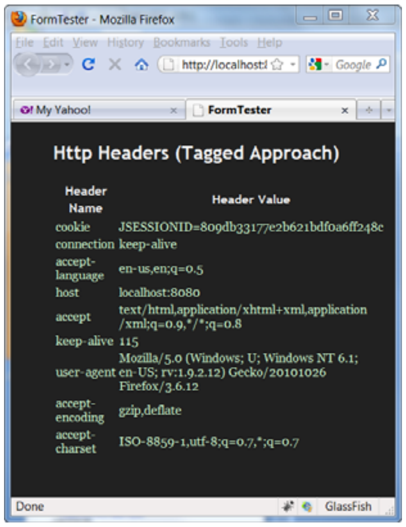
```
<h2>Http Headers</h2>
   <table>
     <tr>
     <th></th>
     <th>Header Name</th>
     <th>Header Value</th></tr>
     <c:forEach var="h" items="${header}" >
       <tr>
         <td class="name"> ${h.key} </td>
         <td>${h.value}</td>
       </tr>
     </c:forEach>
   </table>
```

35

© Robert Kelly, 2001-2012

Form Tester With a Custom Tag

```
 <%@taglib prefix= "t"
       uri=« CSE336"  %>
...
<h2>Http Headers
   (Tagged Approach)</h2>
    <table>
      <tr>
      <th>Header Name</th>
      <th>Header Value</th></tr>
      <t:showMap items="${header}">
        <tr>
          <td>${name} </td>
          <td>${value}</td>
        </tr>
      </t:showMap>
    </table>
```


FormTester - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:
Google
My Yahoo!
FormTester
Done
GlassFish

Http Headers (Tagged Approach)

Header Name	Header Value
cookie	JSESSIONID=809db33177e2b621bdf0a6ff248c
connection	keep-alive
accept-language	en-us,en;q=0.5
host	localhost:8080
accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
keep-alive	115
user-agent	Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.12) Gecko/20101026 Firefox/3.6.12
accept-encoding	gzip,deflate
accept-charset	ISO-8859-1,utf-8;q=0.7,*;q=0.7

What is ${name}?

36

© Robert Kelly, 2001-2012

tld

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.0" xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-
   jsptaglibrary_2_0.xsd">
  <tlib-version>1.0</tlib-version>
  <short-name>demo</short-name>
  <uri>CSE336</uri>
...
  <tag>
    <name>showMap</name>
    <tag-class>lectures.ShowMap</tag-class>
    <body-content>scriptless</body-content>
    <attribute>
      <name>items</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
      <type>java.util.Map</type>
    </attribute>
  </tag>
</taglib>
```

37

ShowMap Class

```
public class ShowMap extends SimpleTagSupport {

  private Map items;

  public void doTag() throws JspException, IOException {
    JspWriter out=getJspContext().getOut();
    JspFragment f=getJspBody();
    Set<Map.Entry> s = items.entrySet();
      for (Map.Entry me : s) {
        getJspContext().setAttribute("name", me.getKey());
        getJspContext().setAttribute("value", me.getValue());
        if (f != null) {
          f.invoke(out);
        }
      }
  }

  public void setItems(Map in) {
       this.items = in;
  }
}
```

Extract the Collection of Map.Entry pairs

Executes the JSP body once for each iteration of the loop

Note: the parameter is a Map

In this example, the tag items attribute is ${header}, so items is the map of header names/values

38

Controlling Subsequent JSP Execution

- If we decided that the page should not continue to be evaluated when the input map was null, we would include the statement:

```
if (items == null) throw new SkipPageException();
```

This is very useful in your security tags (when you do a CSE308 project)

© Robert Kelly, 2001-2012

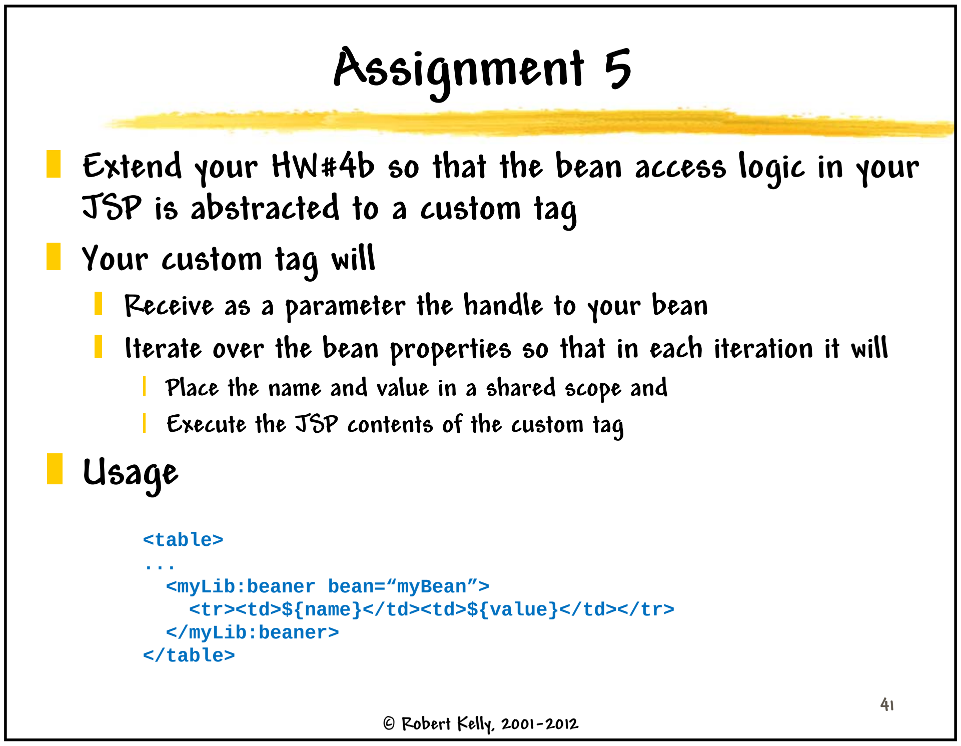
39

Have You Satisfied the Lecture Objectives?

- Understand how to abstract JSP control logic into a custom tag
- Learn how a tld is used to:
  - define the interface of a tag (name, attributes, and type)
  - Specify the Java class that implements the tag
- Learn the life cycle of the custom tag
- Become familiar with differences between JSP 2.0 approach and classic tag approach

© Robert Kelly, 2001-2012

40



Assignment 5

- Extend your HW#4b so that the bean access logic in your JSP is abstracted to a custom tag
- Your custom tag will
  - Receive as a parameter the handle to your bean
  - Iterate over the bean properties so that in each iteration it will
    - Place the name and value in a shared scope and
    - Execute the JSP contents of the custom tag
- Usage

```
<table>
...
  <myLib:beaner bean="myBean">
    <tr><td>${name}</td><td>${value}</td></tr>
  </myLib:beaner>
</table>
```

41