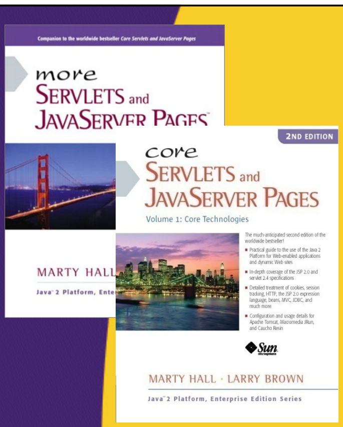
Creating Custom JSP Tag Libraries: Advanced Topics

Originals of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/csajsp2.html

Customized Java EE Training: http://courses.coreservlets.com/
Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.


more SERVLETS and JAVASERVER PAGES
MARTY HALL
core SERVLETS and JAVASERVER PAGES
2ND EDITION
Volume 1: Core Technologies
MARTY HALL · LARRY BROWN
Java 2 Platform, Enterprise Edition Series

Agenda

- Manipulating the tag body
- Tags with dynamic attribute values
- Tags with complex objects for attributes
- Looping tags
- Nested tags
- Using TagLibraryValidator to validate tag library syntax

Tags that Manipulate Their Body Content

Idea

- **Earlier, we had tags with bodies. But:**
  - Tags did not modify the body content
  - Tag behavior did not change based on the body content
- **To manipulate the body, pass a custom Writer to the invoke method**
  - The Writer should buffer the results
    - StringWriter is simplest
    - Very similar to approach of output-modifying filters
  - The tag can then modify or examine the buffer
  - The tag is responsible for outputting the buffer
    - Using getJspContext().getOut() as in normal tags

Manipulating Tag Body: Summary

- **Including tag bodies (unchanged)**

```
getJspBody().invoke(null)
```

- **Modifying tag body**

```
StringWriter stringWriter = new StringWriter();
getJspBody().invoke(stringWriter);
String modifiedBody = modifyString(stringWriter.toString());
getJspContext().getOut().print(modifiedBody);
```

- **Changing behavior based on tag body**

```
StringWriter stringWriter = new StringWriter();
getJspBody().invoke(stringWriter);
String body = stringWriter.toString();
if (hasCertainProperties(body)) {
  doThis(body);
} else  {
  doThat(body);  }
```

An HTML-Filtering Tag (Java Code)

```
public class HtmlFilterTag extends SimpleTagSupport {
  public void doTag() throws JspException, IOException {
    // Buffer tag body's output
    StringWriter stringWriter = new StringWriter();
    getJspBody().invoke(stringWriter);

    // Filter out any special HTML characters
    // (e.g., "<" becomes "&lt;")
    String output =
      ServletUtilities.filter(stringWriter.toString());

    // Send output to the client
    JspWriter out = getJspContext().getOut();
    out.print(output);
  }
}
```

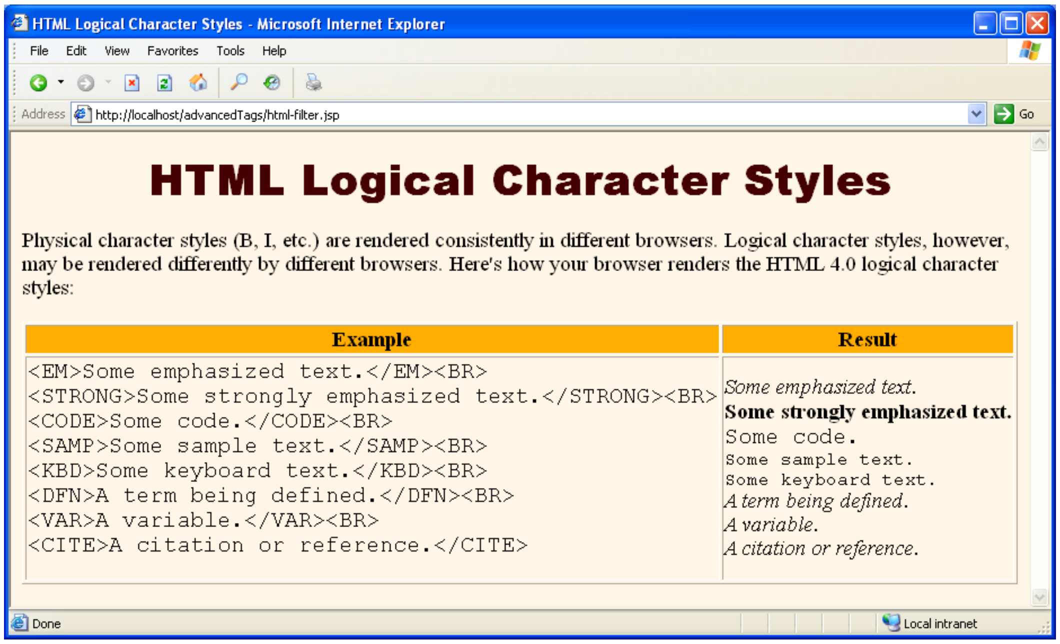
HTML-Filtering Tag (TLD File)

```
...
<tag>
  <description>
    Converts special HTML characters such as less than
    and greater than signs to their corresponding HTML
    character entities such as &lt; and &gt;.
  </description>
  <name>filterhtml</name>
  <tag-class>coreservlets.tags.HtmlFilterTag</tag-class>
  <body-content>scriptless</body-content>
</tag>
...
```

HTML-Filtering Tag (JSP Page)

```
<TABLE BORDER=1 ALIGN="CENTER">
<TR CLASS="COLORED"><TH>Example<TH>Result
<%@ taglib uri="/WEB-INF/tlds/csajsp-taglib-adv.tld"
           prefix="csajsp" %>
<TR>
<TD><PRE><csajsp:filterhtml>
<EM>Some emphasized text.</EM><BR>
<STRONG>Some strongly emphasized text.</STRONG><BR>
<CODE>Some code.</CODE><BR>
<SAMP>Some sample text.</SAMP><BR>
<KBD>Some keyboard text.</KBD><BR>
<DFN>A term being defined.</DFN><BR>
<VAR>A variable.</VAR><BR>
<CITE>A citation or reference.</CITE>
</csajsp:filterhtml></PRE>
<TD>
<EM>Some emphasized text.</EM><BR>
...
</TABLE>
```

HTML-Filtering Tag (Result)


HTML Logical Character Styles - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/advancedTags/html-filter.jsp
Go
Done
Local intranet

HTML Logical Character Styles

Physical character styles (B, I, etc.) are rendered consistently in different browsers. Logical character styles, however, may be rendered differently by different browsers. Here's how your browser renders the HTML 4.0 logical character styles:

Example	Result
`<EM>Some emphasized text.</EM><BR>` `<STRONG>Some strongly emphasized text.</STRONG><BR>` `<CODE>Some code.</CODE><BR>` `<SAMP>Some sample text.</SAMP><BR>` `<KBD>Some keyboard text.</KBD><BR>` `<DFN>A term being defined.</DFN><BR>` `<VAR>A variable.</VAR><BR>` `<CITE>A citation or reference.</CITE>`	*Some emphasized text.* **Some strongly emphasized text.** Some code. Some sample text. Some keyboard text. *A term being defined.* *A variable.* *A citation or reference.*

Dynamic Attributes and Looping Tags

Tags with Dynamic Attribute Values

- **Problem**
  - You need request time values for your custom tags
    - <mytags:if test="${myBean.missingValue}">
      ${myBean.errorMessage}
      </mytags:if>
    - <mytags:prime
      length="<%= (int)(Math.random()*100000) %>"/>
    - <mytags:showCalendar month="<%= new Date() %>"/>
- **Solution**
  - Use true for rtexprvalue in attribute declaration in TLD
    - <attribute>
      …
      <rtexprvalue>true</rtexprvalue>
      </attribute>

Simple Looping Tag (Java Code)

```
public class ForTag extends SimpleTagSupport {
  private int count;

  public void setCount(int count) {
    this.count = count;
  }

  public void doTag() throws JspException, IOException {
    for(int i=0; i<count; i++) {
      getJspBody().invoke(null);
    }
  }
}
```

Simple Looping Tag (TLD File)

```
...
<tag>
  <description>
    Loops specified number of times.
  </description>
  <name>for</name>
  <tag-class>coreservlets.tags.ForTag</tag-class>
  <body-content>scriptless</body-content>
  <attribute>
    <description>
      Number of times to repeat body.
    </description>
    <name>count</name>
    <required>true</required>
    <rtexprvalue>true</rtexprvalue>
  </attribute>
</tag>
```

Simple Looping Tag (Servlet)

```
@WebServlet("/simple-loop-test")
public class SimpleLoopTest extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    CoinBean coin = new CoinBean();
    request.setAttribute("coin", coin);
    String address =
      "/WEB-INF/results/simple-loop-test.jsp";
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
  }
}
```

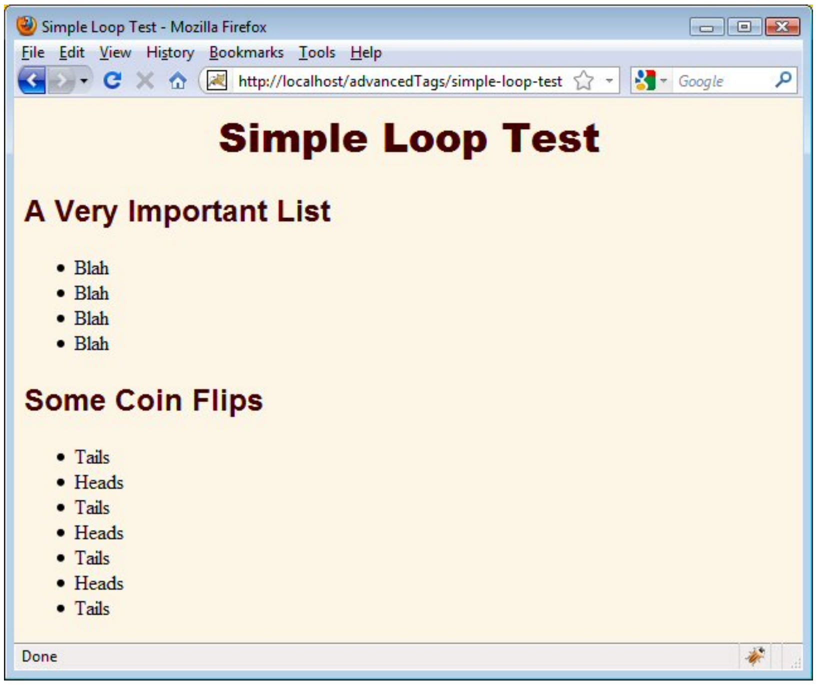
Simple Looping Tag (Bean)

```
public class CoinBean {
  public String getFlip() {
    if (Math.random() < 0.5) {
      return("Heads");
    } else {
      return("Tails");
    }
  }
}
```

Simple Looping Tag (Results Page)

```
<H1>Simple Loop Test</H1>
<P>
<%@ taglib uri="/WEB-INF/tlds/csajsp-taglib-adv.tld"
           prefix="csajsp" %>
<H2>A Very Important List</H2>
<UL>
  <csajsp:for count="<%=(int)(Math.random()*10)%>">
    <LI>Blah
  </csajsp:for>
</UL>
<H2>Some Coin Flips</H2>
<UL>
  <csajsp:for count="<%=(int)(Math.random()*10)%>">
    <LI>${coin.flip}
  </csajsp:for>
</UL>
```

Simple Looping Tag (Result)


Simple Loop Test - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/advancedTags/simple-loop-test
Google
Done

Simple Loop Test

A Very Important List

- Blah
- Blah
- Blah
- Blah

Some Coin Flips

- Tails
- Heads
- Tails
- Heads
- Tails
- Heads
- Tails

Complex (Dynamic) Attributes

Tags with Complex Objects for Attributes

- **What if you want type other than String or a primitive type for a tag attribute value?**
  – E.g., to access values stored by a servlet in the results page of an MVC response
- **Issues**
  – Must declare setter to accept the high-level type
  – Must declare attribute with rtexprvalue as true
  – Usually supply value with the JSP EL
    - Although JSP expression is technically legal
  – Harder to do error checking than with String values
    - If value is incorrect type, it never gets passed to your method, and you get a runtime error

Table Formatting Tag (Java Code)

```
public class MakeTableTag extends SimpleTagSupport {
  private Object[][] rowItems;
  private String headerClass;
  private String bodyClass;

  public void setRowItems(Object[][] rowItems) {
    this.rowItems = rowItems;
  }

  public void setHeaderClass(String headerClass) {
    this.headerClass = headerClass;
  }

  public void setBodyClass(String bodyClass) {
    this.bodyClass = bodyClass;
  }
```

Table Formatting Tag (Java Code, Continued)

```
public void doTag() throws JspException, IOException {
   if (rowItems.length > 0) {
     JspContext context = getJspContext();
     JspWriter out = context.getOut();
     out.println("<TABLE BORDER=1>");
     Object[] headingRow = rowItems[0];
     printOneRow(headingRow, getStyle(headerClass),
                 out);
     for(Object[] bodyRow: rowItems) {
       printOneRow(bodyRow, getStyle(bodyClass), out);
     }
     out.println("</TABLE>");
   }
 }
```

Table Formatting Tag (Java Code, Continued)

```
private void printOneRow(Object[] columnEntries,
                          String style,
                          JspWriter out)
     throws IOException {
   out.println("  <TR" + style + ">");
   for(Object columnEntry: columnEntries) {
     out.println("    <TD>" + columnEntry + "</TD>");
   }
   out.println("  </TR>");
 }
private String getStyle(String className) {
   if (className == null) {
     return("");
   } else {
     return(" CLASS=\"" + headerClass + "\"");
 }
}
```

Table Formatting Tag (TLD File)

```
 ...
 <tag>
   <description>
     Given an array of arrays, puts values into a table
   </description>
   <name>makeTable</name>
   <tag-class>coreservlets.tags.MakeTableTag</tag-class>
   <body-content>scriptless</body-content>
   <attribute>
     <description>
       An array of arrays. The top-level arrays
       represents the rows, the sub-arrays represent
       the column entries.
     </description>
     <name>rowItems</name>
     <required>true</required>
     <rtexprvalue>true</rtexprvalue>
   </attribute>
```

Table Formatting Tag (TLD File, Continued)

```
 <attribute>
    <description>
      Style sheet class name for table header.
    </description>
    <name>headerClass</name>
    <required>false</required>
  </attribute>
  <attribute>
    <description>
      Style sheet class name for table body.
    </description>
    <name>bodyClass</name>
    <required>false</required>
  </attribute>
</tag>
...
```

Table Formatting Tag (Servlet)

```
@WebServlet("/show-records")
public class ShowRecords extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    Object[][] records = WorldRecords.recentRecords();
    request.setAttribute("records", records);
    String address =
      "/WEB-INF/results/show-records.jsp";
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
  }
}
```

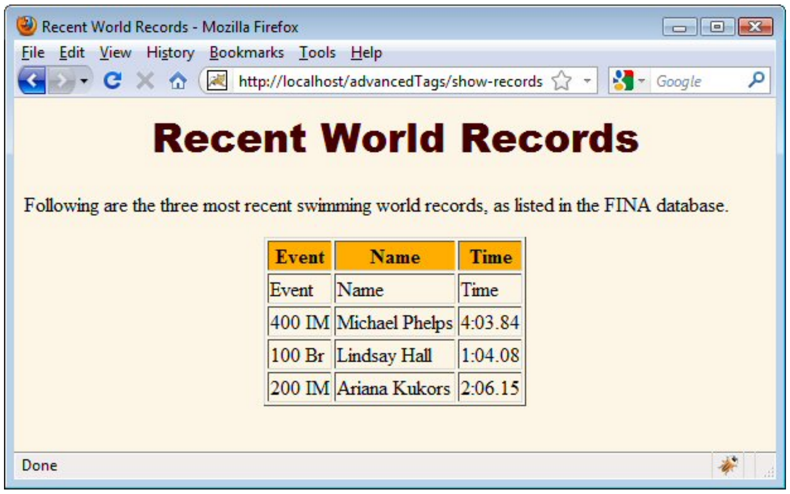
Table Formatting Tag (Supporting Class)

```
public class WorldRecords {
  public static Object[][] recentRecords() {
    Object[][] records = {
      { "Event", "Name", "Time" },
      { "400 IM", "Michael Phelps", "4:03.84"},
      { "100 Br", "Lindsay Hall", "1:04.08"},
      { "200 IM", "Ariana Kukors", "2:06.15"}};
    return(records);
  }
}
```

Table Formatting Tag (Results Page)

```
<H1>Recent World Records</H1>
Following are the three most recent swimming
world records, as listed in the FINA database.
<P>
<%@ taglib uri="/WEB-INF/tlds/csajsp-taglib-adv.tld"
           prefix="csajsp" %>
<CENTER>
<csajsp:makeTable rowItems="${records}"
                  headerClass="COLORED" />
</CENTER>
```

Table Formatting Tag (Result)


Recent World Records - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/advancedTags/show-records
Google
Done

Recent World Records

Following are the three most recent swimming world records, as listed in the FINA database.

Event	Name	Time
Event	Name	Time
400 IM	Michael Phelps	4:03.84
100 Br	Lindsay Hall	1:04.08
200 IM	Ariana Kukors	2:06.15

General-Purpose Looping Tags

Problems with makeTable

- **HTML in tag**
  - HTML written by Java author, not Web designer
- **Always makes a table**
  - Can't change to bulleted list, or headings, or plain text
- **Limited customization**
  - If tag designer didn't build in option, you can't do it
    - Since no HTML exposed to page author
- **Requires very specific data format**
  - Array of arrays. What about lists? What about arrays where data is in different order?
- **Only for displaying fixed results**
  - No ability to operate on cell values

Looping Tags

- **What if you want a tag that outputs its body more than once?**
  - Of course, the body should give different values each time
- **Issues**
  - Attribute should accept a collection
    - Covered in previous section
  - Attribute should be defined with rtexprvalue as true
    - Covered in section before that
  - Body should have access to each item in collection
    - New feature needed: tag should call Use getJspContext().setAttribute(key, object) to place a bean that is accessible *only* within the body of the tag, i.e., in tag scope

ForEach Tag (Java Code)

```
public class ForEachTag extends SimpleTagSupport {
  private Object[] items;
  private String attributeName;

  public void setItems(Object[] items) {
    this.items = items;
  }

  public void setVar(String attributeName) {
    this.attributeName = attributeName;
  }

  public void doTag() throws JspException, IOException {
    for(Object item: items) {
      getJspContext().setAttribute(attributeName, item);
      getJspBody().invoke(null);
    }
  }
}
```

ForEach Tag (TLD File)

```
...
<tag>
  <description>
    Loops down each element in an array
  </description>
  <name>forEach</name>
  <tag-class>coreservlets.tags.ForEachTag</tag-class>
  <body-content>scriptless</body-content>
```

ForEach Tag (TLD File, Continued)

```
<attribute>
     <description>
       The array of elements.
     </description>
     <name>items</name>
     <required>true</required>
     <rtexprvalue>true</rtexprvalue>
   </attribute>
   <attribute>
    <description>
       The name of the local variable that
       each entry will be assigned to.
     </description>
     <name>var</name>
     <required>true</required>
   </attribute>
 </tag> ...
```

ForEach Tag (Servlet)

```
@WebServlet("/loop-test")
public class LoopTest extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    String[] servers =
      {"Tomcat", "Resin", "Jetty", "WebLogic",
       "WebSphere", "JBoss", "Glassfish" };
    request.setAttribute("servers", servers);
    Object[][] records = WorldRecords.recentRecords();
    request.setAttribute("records", records);
    String address = "/WEB-INF/results/loop-test.jsp";
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
  }
}
```

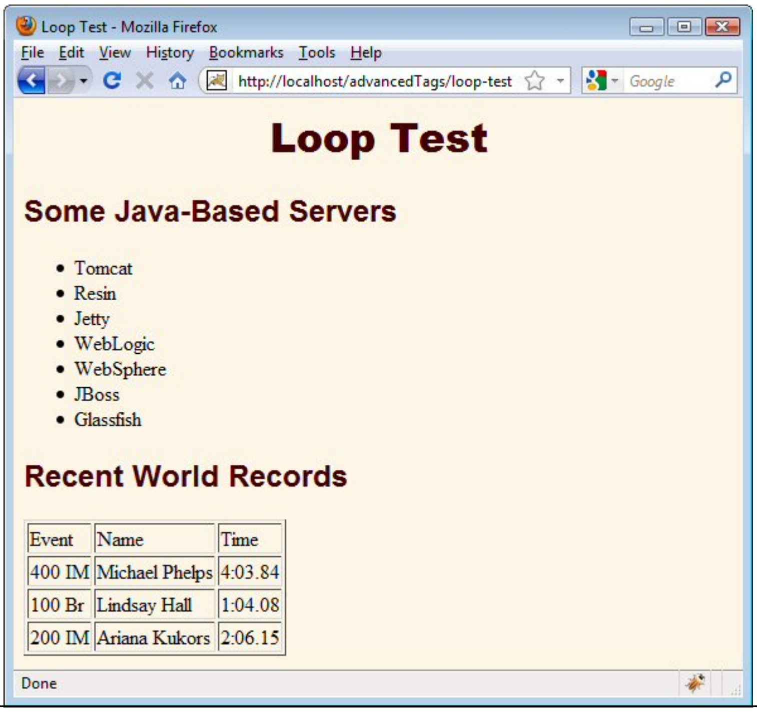
ForEach Tag (Results Page)

```
<%@ taglib uri="/WEB-INF/tlds/csajsp-taglib-adv.tld"
           prefix="csajsp" %>
<H2>Some Java-Based Servers</H2>
<UL>
  <csajsp:forEach items="${servers}" var="server">
    <LI>${server}
  </csajsp:forEach>
</UL>
```

ForEach Tag (Results Page, Continued)

```
<H2>Recent World Records</H2>
<TABLE BORDER=1>
<csajsp:forEach items="${records}" var="row">
  <TR>
  <csajsp:forEach items="${row}" var="col">
    <TD>${col}</TD>
  </csajsp:forEach>
  </TR>
</csajsp:forEach>
</TABLE>
```

ForEach Tag (Result)


Loop Test - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/advancedTags/loop-test
Google
Done

Loop Test

Some Java-Based Servers

- Tomcat
- Resin
- Jetty
- WebLogic
- WebSphere
- JBoss
- Glassfish

Recent World Records

Event	Name	Time
400 IM	Michael Phelps	4:03.84
100 Br	Lindsay Hall	1:04.08
200 IM	Ariana Kukors	2:06.15

Note that JSTL (covered in later lecture) already has an even better version of the forEach tag already builtin. The point is not to use *this* forEach tag, but to illustrate the types of powerful tags that can be built with a combination of looping and accepting complex runtime types as attribute values.

Nested Tags

Nested Tags

- **What if tag behavior depends on surrounding tag or earlier tag?**
  - <mytags:if test="<%= Math.random() < 0.5 %>">
        <mytags:then>Heads</mytags:then>
        <mytags:else>Tails</mytags:else>
    </mytags:if>
- **Communicating with surrounding tag**
  - getParent returns directly surrounding tag
    - Returns null if there is no surrounding custom tag
  - findAncestorWithClass(this, OuterTag.class) finds possibly indirectly surrounding tag of given type
    - Returns null if no surrounding tag of given type is found
- **Communicating with earlier tag**
  - Earlier tag finds surrounding tag and stores result
  - Later tag finds surrounding tag and retrieves result

If Tag (Java Code)

```
public class IfTag extends SimpleTagSupport {
  private boolean test;

  public void setTest(boolean test) {
    this.test = test;
  }

  public boolean getTest() {
    return(test);
  }

  public void doTag() throws JspException, IOException {
    getJspBody().invoke(null);
  }
}
```

Then Tag (Java Code)

```
public class ThenTag extends SimpleTagSupport {
  public void doTag() throws JspException, IOException {
    try {
      IfTag ifTag = (IfTag)getParent();
      if (ifTag.getTest()) {
        getJspBody().invoke(null);
      }
    } catch(Exception e)  {
      String msg =
        "Error: 'then' must be inside 'if'.";
      throw new JspTagException(msg);
    }
  }
}
```

Else Tag (Java Code)

```
public class ElseTag extends SimpleTagSupport {
  public void doTag() throws JspException, IOException {
    try {
      IfTag ifTag = (IfTag)getParent();
      if (!ifTag.getTest()) {
        getJspBody().invoke(null);
      }
    } catch(Exception e)  {
      String msg =
        "Error: 'else' must be inside 'if'.";
      throw new JspTagException(msg);
    }
  }
}
```

If Tag (TLD File)

```
...
<tag>
  <description>If tag</description>
  <name>if</name>
  <tag-class>coreservlets.tags.IfTag</tag-class>
  <body-content>scriptless</body-content>
  <attribute>
    <description>Condition of the if</description>
    <name>test</name>
    <required>true</required>
    <rtexprvalue>true</rtexprvalue>
  </attribute>
</tag>
```

Then/Else Tags (TLD File)

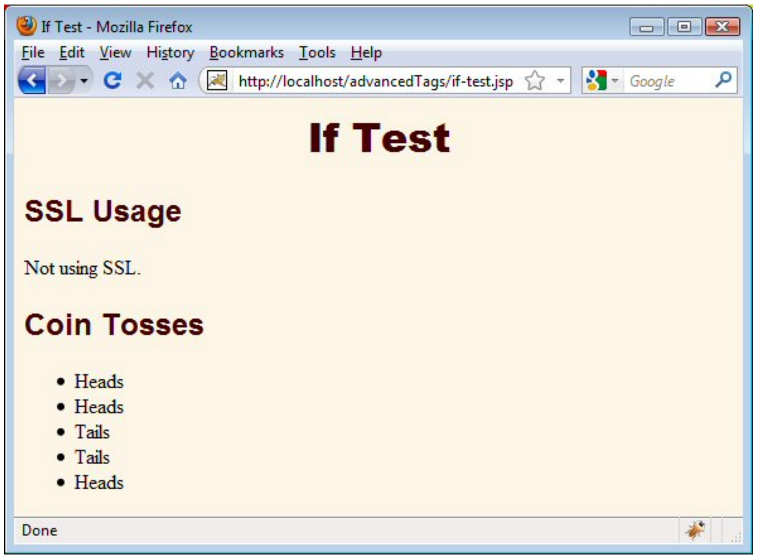
```
...
<tag>
  <description>Then tag (goes with If tag)</description>
  <name>then</name>
  <tag-class>coreservlets.tags.ThenTag</tag-class>
  <body-content>scriptless</body-content>
</tag>

<tag>
  <description>Else tag (goes with If tag)</description>
  <name>else</name>
  <tag-class>coreservlets.tags.ElseTag</tag-class>
  <body-content>scriptless</body-content>
</tag>
```

If Tag (JSP Page)

```
<%@ taglib uri="/WEB-INF/tlds/csajsp-taglib-adv.tld"
           prefix="csajsp" %>
<H2>SSL Usage</H2>
<csajsp:if
    test="${pageContext.request.protocol==https}">
  <csajsp:then>Using SSL.</csajsp:then>
  <csajsp:else>Not using SSL.</csajsp:else>
</csajsp:if>
<H2>Coin Tosses</H2>
<UL>
  <csajsp:for count="5">
    <LI><csajsp:if test="<%=Math.random()<0.5%>">
          <csajsp:then>Heads</csajsp:then>
          <csajsp:else>Tails</csajsp:else>
        </csajsp:if>
  </csajsp:for>
</UL>
```

If Tag (Result)


If Test - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/advancedTags/if-test.jsp
Google
Done

If Test

SSL Usage

Not using SSL.

Coin Tosses

- Heads
- Heads
- Tails
- Tails
- Heads

Page Translation Time Syntax Checking

Semantics of Custom Tag Usage

- **System already uses the JSP DTD to verify that the *standard* tags are used properly.**
- **System will already verify basic custom tag syntax**
  - Tags are well formed
  - All tag and attribute names spelled properly
  - Required attributes supplied
  - No undeclared attributes used
- **But, what about deeper issues?**
  - Certain custom tags must be nested in certain patterns
  - A custom tag has two attributes: both must appear or neither must appear.

Big Ideas

- **All JSP pages are turned into XML**
  – Before they are turned into servlets, they are turned into XML documents.
- **You can get at the XML version**
  – After the JSP page is turned into XML document, you can run arbitrary checks on it. If you throw an exception, page translation fails.
- **Examples**
  – Certain tags must be nested inside others
  – If tag uses the foo attribute, it cannot use the bar attribute
  – The baz tag can appear no more than three times in page
  – Almost anything you want to enforce

Example: XML Version of JSP

Original Page	Internal Representation

`<DOCTYPE …>`
`…`
`<%= Math.random() %>`
`<myTags:doSomething>`
`Blah`
`</myTags:doSomething>`
`</BODY></HTML>`
	`<?xml version="1.0" ?>`
	`<jsp:root …>`
	`<jsp:text>`
	`<![CDATA[ <DOCTYPE…>… ]]>`
	`</jsp:text>`
	`<jsp:expression>`
	`Math.random()`
	`</jsp:expression>`
	`<myTags:doSomething>`
	`Blah`
	`</myTags:doSomething>`
	`<jsp:text>`
	`<![CDATA[ </BODY>… ]]>`
	`</jsp:text>`
	`</jsp:root>`

Checking Tag Library Syntax with TagLibraryValidator

- **Create a subclass of TagLibraryValidator.**
- **Override the validate method.**

```
public ValidationMessage[] validate(String prefix,
                                    String uri,
                                    PageData page) {
  InputStream stream = page.getInputStream();
  // Pass stream to SAX parser; return null if valid
```

  - The InputStream reads a pure-XML version of the JSP page. E.g, **<%= foo %>** will be read as **<jsp:expression>foo</jsp:expression>**.

- **Declare the validator in the TLD file.**

```
<taglib>…
  <validator>
    <validator-class>
      somePackage.SomeValidatorClass
    </validator-class>
  </validator>
… </taglib>
```

Example: Enforcing Nesting Order

- **outerTag cannot be nested**
- **innerTag can only appear within outerTag**
  – Directly or indirectly
- **innerTag can be nested arbitrarily**

Legal:

```
<test:outerTag>
  <test:innerTag>
    <test:innerTag/>
  </test:innerTag>
  <test:innerTag>
    <test:innerTag>
      <test:innerTag/>
    </test:innerTag>
   </test:innerTag>
</test:outerTag>
<test:outerTag/>
```

Illegal:

```
<test:innerTag/>
```

Also Illegal:

```
<test:outerTag>
  <test:outerTag/>
</test:outerTag>
```

Enforcing Nesting Order: SAX Handler

```
public void startElement(String namespaceUri,
                         String localName,
                         String qualifiedName,
                         Attributes attributes)
    throws SAXException {
  String tagName = mainTagName(qualifiedName);
  if (tagName.equals(outerTagName)) {
    if (inOuterTag) {
      throw new SAXException("\nCannot nest "
                             + outerTagName);
    }
    inOuterTag = true;
  } else if (tagName.equals(innerTagName)
             && !inOuterTag) {
    throw new SAXException("\n" + innerTagName +
                           " can only appear within " +
                           outerTagName);
  }
}
```

Enforcing Nesting Order: SAX Handler (Continued)

```
public void endElement(String namespaceUri,
                       String localName,
                       String qualifiedName)
    throws SAXException {
  String tagName = mainTagName(qualifiedName);
  if (tagName.equals(outerTagName)) {
    inOuterTag = false;
  }
}
```

Wrap-Up

Summary

- **Manipulating or checking the tag body**
  - Pass custom writer (esp. StringWriter) to invoke
- **Tags with dynamic attribute values**
  - Specify true for rtexprvalue
- **Tags with complex objects for attributes**
  - Have setter accept complex type, use true for rtexprvalue
- **Looping tags**
  - Call jspContext.setAttribute; read it via EL in tag body
- **Nested tags**
  - Call getParent or findAncestorWithClass, cast to tag type, check for null
- **Using TagLibraryValidator (rare!)**
  - Extend TagLibraryValidator, override validate
  - Get InputStream to read XML representation of page

Questions?