

Usage Rules

- PowerPoint slides for use only in for-credit courses at degree-granting institutions
- Slides can be modified however instructor desires
- Instructor can give PDF or hardcopy to students, but should protect the PowerPoint files

– This slide is suppressed in Slide Show mode

Custom Tags

www.coreservlets.com



Creating Custom JSP Tag Libraries

Lecture 12 – Chapter 14

Core Servlets & JSP book: www.coreservlets.com
More Servlets & JSP book: www.moreservlets.com
Servlet and JSP Training Courses: courses.coreservlets.com

Slides © Marty Hall, <http://www.coreservlets.com>, book © Sun Microsystems Press

Agenda

- Components of a tag library
- Basic tags
- Tags that use attributes
- Tags that use body content
- Tags that optionally use body content
- Advanced tags

Custom Tags

www.coreservlets.com

Uses of JSP Constructs

Simple Application



Complex Application

- Scripting elements calling servlet code directly
- Scripting elements calling servlet code indirectly (by means of utility classes)
- Beans
- Custom tags
- Servlet/JSP combo (MVC), with beans and possibly custom tags

Custom Tags

www.coreservlets.com

Custom Tags vs. JavaBeans

- Custom Tags accomplish some of the same goals as beans, but there are differences:
 - Beans cannot manipulate JSP content
 - But Custom Tags can.
 - Complex operations can be reduced to a single tag with Custom Tags
 - But beans require `<jsp:useBean>`, `<jsp:getProperty>`, `<jsp:setProperty>`, etc.
 - Custom Tags are a bit more work to master than beans.
 - Custom Tags are available starting with JSP 1.1
 - JavaBeans are available starting with JSP 1.0
 - Tomcat started supporting Custom Tags in v 3.1

Custom Tags

www.coreservlets.com

Components That Make Up a Tag Library

- The Tag Handler Class
 - Java code that says how to actually translate tag into code
 - Must implement `javax.servlet.jsp.tagext.Tag` interface
 - Usually extends `TagSupport` or `BodyTagSupport`
 - Goes in same directories as servlet class files and beans
- The Tag Library Descriptor File
 - XML file describing tag name, attributes, and implementing tag handler class
 - Goes with JSP file or at arbitrary URL
- The JSP File
 - Imports a tag library (referencing URL of descriptor file)
 - Defines tag prefix
 - Uses tags

Custom Tags

www.coreservlets.com

Defining a Simple Tag Handler Class

- **Extend the TagSupport class**
- **Import needed packages**

```
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
```
- **Override doStartTag**
 - Obtain the `JspWriter` by means of `pageContext.getOut()`
 - Use the `JspWriter` to generate JSP content
 - Return `SKIP_BODY`
 - Translated into servlet code at page-translation time
 - Code gets called at *request* time

Custom Tags

www.coreservlets.com

Defining a Simple Tag Handler Class: Example

```
package coreservlets.tags;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
import java.math.*;
import coreservlets.*;

public class SimplePrimeTag extends TagSupport {
    protected int len = 50;

    public int doStartTag() {
        try {
            JspWriter out = pageContext.getOut();
            BigInteger prime = Primes.nextPrime(Primes.random(len));
            out.print(prime); // Primes class defined in Section 7.3
        } catch (IOException ioe) {
            System.out.println("Error generating prime: " + ioe);
        }
        return (SKIP_BODY);
    }
}
```

Custom Tags

www.coreservlets.com

Defining a Simple Tag Library Descriptor

- **Start with XML header and DOCTYPE**
- **Top-level element is taglib**
- **Each tag defined by tag element with:**
 - **name**, whose body defines the base tag name.
In this case, I use `<name>simplePrime</name>`
 - **tagclass**, which gives the fully qualified class name of the tag handler. In this case, I use `<tagclass>coreservlets.tags.SimplePrimeTag</tagclass>`
 - **bodycontent**, which gives hints to development environments. Optional.
 - **info**, which gives a short description. Here, I use `<info>Outputs a random 50-digit prime.</info>`

Custom Tags

www.coreservlets.com

TLD File for SimplePrimeTag

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib PUBLIC
    "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
    "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
<taglib>
    ... // A few standard items -- just copy these
    <tag>
        <name>simplePrime</name>
        <tagclass>coreservlets.tags.SimplePrimeTag</tagclass>
        <info>Outputs a random 50-digit prime.</info>
    </tag>
</taglib>
```

- **Don't memorize XML header and DOCTYPE; modify version from coreservlets.com**

Custom Tags

www.coreservlets.com

Accessing Custom Tags From JSP Files

- **Import the tag library**
 - Specify location of TLD file
`<% @ taglib uri="csajsp-taglib.tld" prefix="csajsp" %>`
 - Define a tag prefix (namespace)
`<% @ taglib uri="csajsp-taglib.tld" prefix="csajsp" %>`
- **Use the tags**
 - `<prefix:tagName />`
 - Tag name comes from TLD file
 - Prefix comes from taglib directive
 - E.g., `<csajsp:simplePrime />`

Custom Tags

www.coreservlets.com

Using simplePrime Tag

```
...
<BODY>
<H1>Some 50-Digit Primes</H1>

<% @ taglib uri="csajsp-taglib.tld" prefix="csajsp" %>

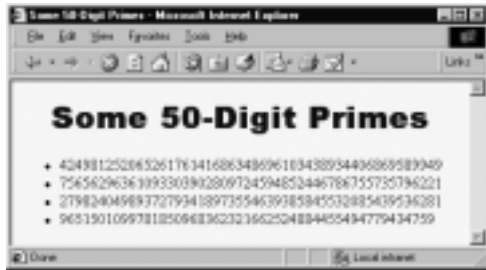
<UL>
    <LI><csajsp:simplePrime />
    <LI><csajsp:simplePrime />
    <LI><csajsp:simplePrime />
    <LI><csajsp:simplePrime />
</UL>

</BODY>
</HTML>
```

Custom Tags

www.coreservlets.com

Using simplePrime Tag: Result



Custom Tags

www.coreservlets.com

Assigning Attributes to Tags

- **Allowing tags like**
 - `<prefix:name attribute1="value1" attribute2="value2" ... attributeN="valueN" />`
- **Tags are still standalone**
 - No body between start and end tags

Custom Tags

www.coreservlets.com

Attributes: The Tag Handler Class

- **Use of an attribute called `attribute1` simply results in a call to a method called `setAttribute1`**
 - Attribute value is supplied to method as a String
- **Example**
 - To support

```
<prefix:tagName attribute1="Test" />
```

add the following to tag handler class:

```
public void setAttribute1(String value1) {  
    doSomethingWith(value1);  
}
```

Custom Tags

www.coreservlets.com

Attributes: PrimeTag.java

```
package coreservlets.tags;  
  
import javax.servlet.jsp.*;  
import javax.servlet.jsp.tagext.*;  
import java.io.*;  
import java.math.*;  
import coreservlets.*;  
  
public class PrimeTag extends SimplePrimeTag {  
    public void setLength(String length) {  
        try {  
            // len used by parent class  
            len = Integer.parseInt(length);  
        } catch (NumberFormatException nfe) {  
            len = 50;  
        }  
    }  
}
```

Custom Tags

www.coreservlets.com

PrimeTag (Continued)

```
public void release() {  
    len = 50;  
}
```

- **Why `release`?**
 - Servers are permitted to reuse tag instances
 - No synchronization issues, since reuse occurs only after tag processing is totally finished.
 - Few current servers reuse tags, but you should plan ahead
 - Not needed if attribute is required. (Why not?)

Custom Tags

www.coreservlets.com

Attributes: The Tag Library Descriptor File

- **The tag element must contain a nested attribute element**
- **The attribute element has three further-nested elements**
 - **name**, a required element that defines the case-sensitive attribute name. In this case, I use `<name>length</name>`
 - **required**, a required element that stipulates whether the attribute must always be supplied (true) or is optional (false). Here, to indicate that length is optional, I use `<required>false</required>`
 - **rtexprvalue**, an optional attribute that indicates whether the attribute value can be a JSP expression like `<%= expression %>` (true) or whether it must be a fixed string (false). The default value is false.

Custom Tags

www.coreservlets.com

Using Tag Body: HeadingTag.java (Continued)

```
public int doStartTag() {
    try {
        JspWriter out = pageContext.getOut();
        out.print("<TABLE BORDER=" + border +
            " BGCOLOR=\"" + bgColor + "\"" +
            " ALIGN=\"" + align + "\"");
        if (width != null) {
            out.print(" WIDTH=\"" + width + "\"");
        }
        ...
    } catch (IOException ioe) {
        System.out.println("Error in HeadingTag: " + ioe);
    }
    return (EVAL_BODY_INCLUDE); // Include tag body
}
```

Custom Tags

www.coreservlets.com

Using Tag Body: HeadingTag.java (cont)

```
public int doEndTag() {
    try {
        JspWriter out = pageContext.getOut();
        out.print("</SPAN></TABLE>");
    } catch (IOException ioe) {
        System.out.println("Error in HeadingTag: " + ioe);
    }
    return (EVAL_PAGE); // Continue with rest of JSP page
}
```

Custom Tags

www.coreservlets.com

Using Tag Body: The Tag Library Descriptor File

- **Only difference is bodycontent element**
 - Should be JSP instead of `empty`:

```
<tag>
  <name>...</name>
  <tagclass>...</tagclass>
  <bodycontent>JSP</bodycontent>
  <info>...</info>
</tag>
```
- **Purpose is primarily for development environments**
 - Advanced IDEs may have drag-and-drop support for custom tags
 - Defined as optional in JSP specification
- **I will omit in my examples**

Custom Tags

www.coreservlets.com

TLD File for HeadingTag

```
...
<taglib>
  <tag>
    <name>heading</name>
    <tagclass>coreservlets.tags.HeadingTag</tagclass>
    <info>Outputs a 1-cell table used as a heading.</info>
    <bodycontent>JSP</bodycontent>
    <attribute>
      <name>bgColor</name>
      <required>true</required> <!-- bgColor is required -->
    </attribute>
    <attribute>
      <name>color</name>
      <required>>false</required>
    </attribute>
    ...
  </tag>
</taglib>
```

Custom Tags

www.coreservlets.com

Using heading Tag

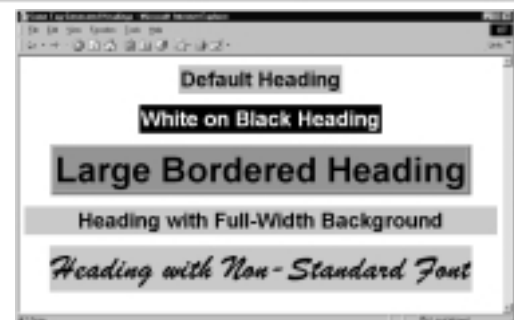
```
<%@ taglib uri="csajsp-taglib.tld" prefix="csajsp" %>
<csajsp:heading bgColor="#C0C0C0">
Default Heading
</csajsp:heading>
<P>
<csajsp:heading bgColor="BLACK" color="WHITE">
White on Black Heading
</csajsp:heading>
<P>
<csajsp:heading bgColor="#EF8429" fontSize="60" border="5">
Large Bordered Heading
</csajsp:heading>
<P>
<csajsp:heading bgColor="CYAN" width="100%">
Heading with Full-Width Background
</csajsp:heading>
...

```

Custom Tags

www.coreservlets.com

Using heading Tag: Result



Custom Tags

www.coreservlets.com

Optional Tag Bodies

- **First examples had no tag bodies**
 - doStartTag returned SKIP_BODY
- **Most recent examples always included tag bodies**
 - doStartTag returned EVAL_BODY_INCLUDE
- **Now: decide whether or not to include tag body at request time**
 - Have doStartTag return either SKIP_BODY or EVAL_BODY_INCLUDE depending on values of request time information

Custom Tags

www.coreservlets.com

Optional Tag Bodies: DebugTag.java

```
package coreservlets.tags;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
import javax.servlet.*;

public class DebugTag extends TagSupport {
    public int doStartTag() {
        ServletRequest request =
            pageContext.getRequest();
        String debugFlag = request.getParameter("debug");
        if ((debugFlag != null) &&
            (!debugFlag.equalsIgnoreCase("false"))) {
            return(EVAL_BODY_INCLUDE);
        } else {
            return(SKIP_BODY);
        }
    }
}
```

Custom Tags

www.coreservlets.com

TLD File for DebugTag

```
...
<taglib>
  <tag>
    <name>debug</name>
    <tagclass>coreservlets.tags.DebugTag</tagclass>
    <info>
      Includes body only if debug param is set.
    </info>
    <bodycontent>JSP</bodycontent>
  </tag>
</taglib>
```

Custom Tags

www.coreservlets.com

Using debug Tag

```
<@ taglib uri="csajsp-taglib.tld" prefix="csajsp" %>
Top of regular page. Blah, blah, blah.
Yadda, yadda, yadda.
<P>
<csajsp:debug>
<B>Debug:</B>
<UL>
  <LI>Current time: <%= new java.util.Date() %>
  <LI>Requesting hostname: <%= request.getRemoteHost() %>
  <LI>Session ID: <%= session.getId() %>
</UL>
</csajsp:debug>
<P>
Bottom of regular page. Blah, blah, blah.
Yadda, yadda, yadda.
```

Custom Tags

www.coreservlets.com

Using debug Tag: Results



Custom Tags

www.coreservlets.com

Manipulating the Tag Body

- **No tag body (<csajsp:prime/>)**
 - <prefix:tagName />
 - <prefix:tagName att1="val1" ... />
- **Previous uses of tag body (<heading/>)**
 - <prefix:tagName>JSP Content</prefix:tagName>
 - <prefix:tagName att1="val1" ... />
JSP Content
</prefix:tagName>
 - Content inserted verbatim
- **Now (<csajsp:filter>)**
 - Same JSP syntax for using a tag body
 - Java code can read/modify/replace tag body

Custom Tags

www.coreservlets.com

Manipulating Tag Body: The Tag Handler Class

- **Extend BodyTagSupport**
 - TagSupport does not have enough infrastructure to support reading/manipulating the tag body
- **New method to override: doAfterBody**
 - Return SKIP_BODY when done
- **getBodyContent returns object of type BodyContent that has three key methods**
 - getEnclosingWriter -- returns the JspWriter being used by doStartTag and doEndTag
 - getReader -- returns a Reader that can read tag's body
 - getString -- returns a String containing entire tag body

Custom Tags

www.coreservlets.com

BodyTagSupport

- **In Section 3.4 we filtered string output to the browser to replace <, >, “, and & with < > " and & respectively**
 - This works fine for servlets and now we wish to have a custom filter tag for JSPs that does the same work
 - Let's define a custom tag called <filter> that processes its tag body and converts the above characters to the corresponding HTML character entities
- **See example 14.6**
 - FilterTag.java

Custom Tags

www.coreservlets.com

Manipulating Tag Body: FilterTag.java

```
public class FilterTag extends BodyTagSupport {
    public int doAfterBody() {
        BodyContent body = getBodyContent();
        String filteredBody =
            // The filter method is defined in Section 3.6
            ServletUtilities.filter(body.getString());
        try {
            JspWriter out = body.getEnclosingWriter();
            out.print(filteredBody);
        } catch (IOException ioe) {
            System.out.println("Error in FilterTag: " + ioe);
        }
        // SKIP_BODY means I'm done. If I wanted to evaluate
        // and handle body again, I'd return EVAL_BODY_AGAIN.
        return(SKIP_BODY);
    }
}
```

Custom Tags

www.coreservlets.com

Manipulating Tag Body: The Tag Library Descriptor File

- **No new capabilities needed**

```
<tag>
  <name>filter</name>
  <tagclass>coreservlets.tags.FilterTag</tagclass>
  <info>
    Replaces HTML-specific
    characters in body.
  </info>
  <bodycontent>JSP</bodycontent>
</tag>
```

Custom Tags

www.coreservlets.com

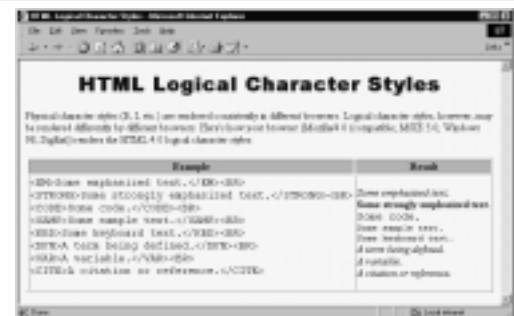
Using the filter Tag

```
...
<%@ taglib uri="csajsp-taglib.tld" prefix="csajsp" %>
<TABLE BORDER=1 ALIGN="CENTER">
<TR CLASS="COLORED"><TH>Example</TH><TH>Result</TH>
<TR>
<TD><PRE><csajsp:filter>
<EM>Some emphasized text.</EM><BR>
<STRONG>Some strongly emphasized text.</STRONG><BR>
<CODE>Some code.</CODE><BR>
...
</csajsp:filter></PRE>
</TD>
<TD>
<EM>Some emphasized text.</EM><BR>
<STRONG>Some strongly emphasized text.</STRONG><BR>
<CODE>Some code.</CODE><BR>
...
</TABLE>
```

Custom Tags

www.coreservlets.com

Using the filter Tag: Results



Custom Tags

www.coreservlets.com

Summary

- **For each custom tag, you need:**
 - A tag handler class (usually extending TagSupport or BodyTagSupport)
 - An entry in a Tag Library Descriptor file
 - A JSP file that imports it, specifies prefix, and uses it
- **Simple tags**
 - Generate output in doStartTag, return SKIP_BODY
- **Attributes**
 - Define setAttributeName method. Update TLD file
- **Body content**
 - doStartTag returns EVAL_BODY_INCLUDE
 - Add doEndTag method

Custom Tags

www.coreservlets.com



Questions?

Core Servlets & JSP book: www.coreservlets.com
More Servlets & JSP book: www.moreservlets.com
Servlet and JSP Training Courses: courses.coreservlets.com

Slides © Marty Hall, <http://www.coreservlets.com>, book © Sun Microsystems Press

More Information

- **Source code for all examples**
 - <http://www.coreservlets.com>
- **Servlet/JSP Training Courses**
 - <http://courses.coreservlets.com>
- **Core Servlets & JSP**
 - <http://www.coreservlets.com>
- **More Servlets & JSP**
 - Sequel to *Core Servlets & JSP*
 - <http://www.moreservlets.com>
- **Servlet home page**
 - <http://java.sun.com/products/servlet/>
- **JavaServer Pages home page**
 - <http://java.sun.com/products/jsp/>



Custom Tags

www.coreservlets.com