Classic (Pre-JSP2) JSP Tag Libraries

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

For live J2EE training, see training courses on JSP, servlets, Struts, JSF, AJAX, and Java 5 at http://courses.coreservlets.com/.

Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.
Additional topics available upon request.

Agenda

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

Motivation

- **Older servers do not support JSP 2**
  - IBM WebSphere 5
  - WebLogic 8.1
  - Oracle 9i
  - Tomcat 4
- **SCWCD exam requires knowledge of classic syntax**
  - But just skim notes: use practice time to work on tags that use current syntax

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

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

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 Sect. 7.3
    } catch(IOException ioe) {
      System.out.println("Error generating prime: " + ioe);
    }
    return(SKIP_BODY);
  }}
```


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>
  - **tag-class**, which gives the fully qualified class name of the tag handler. In this case, I use <tag-class>coreservlets.tags.SimplePrimeTag</tag-class>
  - **body-content**, which gives hints to development environments. Optional.
  - **description**, which gives a short description. Here, I use <description>Outputs a random 50-digit prime.</description>

TLD File for SimplePrimeTag

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<taglib …>
  ... // A few standard items -- just copy these
  <tag>
    <name>simplePrime</name>
    <tag-class>
      coreservlets.tags.SimplePrimeTag
    </tag-class>
    <description>
      Outputs a random 50-digit prime.
    </description>
  </tag>
</taglib>
```

- **Don't memorize XML header; modify version from moreservlets.com**

Accessing Custom Tags From JSP Files

- **Import the tag library**
  - Specify location of TLD file
    <%@ taglib uri="/WEB-INF/csajsp-taglib.tld"
    prefix="csajsp" %>
  - Define a tag prefix (namespace)
    <%@ taglib uri="/WEB-INF/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 />

Using simplePrime Tag

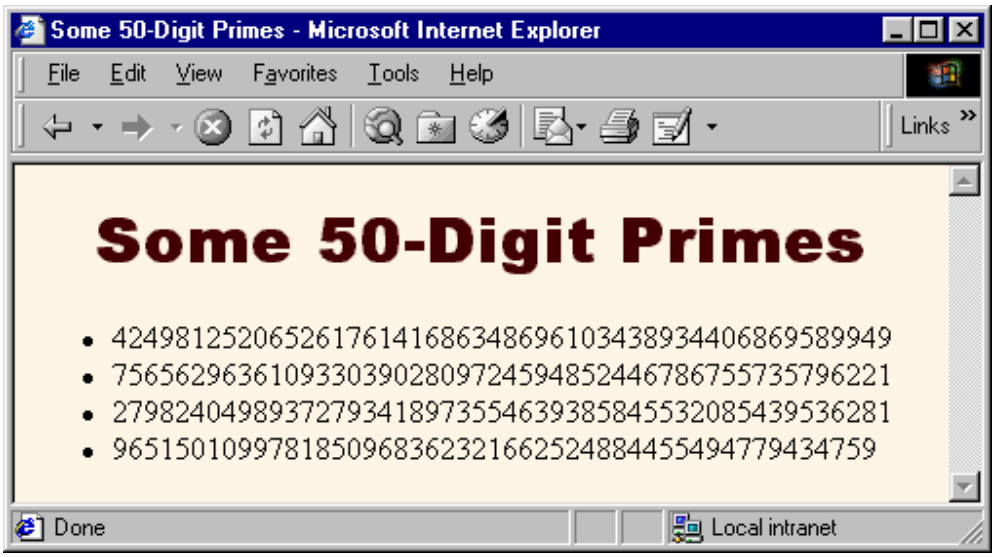
```
...
<BODY>
<H1>Some 50-Digit Primes</H1>

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

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

</BODY>
</HTML>
```


Using simplePrime Tag: Result



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
- **Exactly the same as with SimpleTagSupport**
  - JSP 2 approach did not change this in any way

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);
}
```

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;
    }
  }
```


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.

TLD File for PrimeTag

```
...
<taglib>
  <tag>
    <name>prime</name>
    <tag-class>coreservlets.tags.PrimeTag</tag-class>
    <description>
      Outputs a random N-digit prime.
    </description>
    <attribute>
      <name>length</name>
      <required>false</required>
    </attribute>
  </tag>
</taglib>
```


Using prime Tag

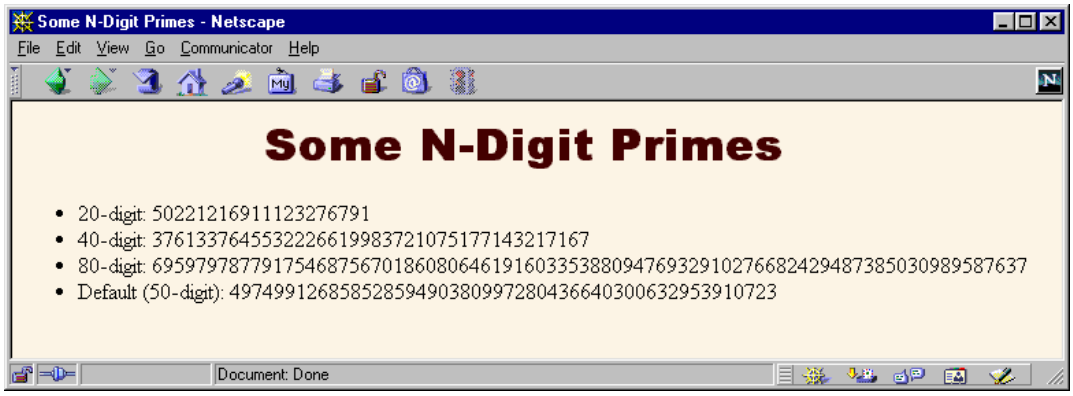
```
...
<BODY>
<H1>Some N-Digit Primes</H1>

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

<UL>
  <LI>20-digit: <csajsp:prime length="20" />
  <LI>40-digit: <csajsp:prime length="40" />
  <LI>80-digit: <csajsp:prime length="80" />
  <LI>Default (50-digit): <csajsp:prime />
</UL>

</BODY>
</HTML>
```

Using prime Tag: Result



Including the Tag Body

- **Simplest tags**
  - <prefix:tagName />
- **Tags with attributes**
  - <prefix:tagName att1="val1" ... />
- **Now**
  - <prefix:tagName>
    JSP Content
    </prefix:tagName>

  - <prefix:tagName att1="val1" ... >
    JSP Content
    </prefix:tagName>

Including Tag Body: The Tag Handler Class

- **doStartTag**
  - Return EVAL_BODY_INCLUDE instead of SKIP_BODY
- **doEndTag**
  - Define this method if you want to take action after handling the body
  - Return EVAL_PAGE
- **Major difference from JSP 2 (SimpleTagSupport)**
  - Body can contain arbitrary JSP scripting elements

Including Tag Body: HeadingTag.java

```
public class HeadingTag extends TagSupport {
  private String align;
  private String bgColor;
  private String border;
  private String fgColor;
  private String font;
  private String size;

  public void setAlign(String align) {
    this.align = align;
  }

  public void setBgColor(String bgColor) {
    this.bgColor = bgColor;
  }

  public void setBorder(String border) {
    this.border = border;
  }…
```

Including Tag Body: HeadingTag.java (Continued)

```
public int doStartTag() {
  try {
    JspWriter out = pageContext.getOut();
    out.print("<TABLE ALIGN=\"" + align + "\"\n" +
              "       BGCOLOR=\"" + bgColor + "\"\n" +
              "       BORDER=" + border + "\">\n");
    out.print("<TR><TH>");
    out.print("<SPAN STYLE=\"color: " + fgColor + ";\n" +
              "             font-family: " + font + ";\n"+
              "             font-size: " + size + "px; " +
              "\">\n"); // End of <SPAN ...>
  } catch(IOException ioe) {
    System.out.println("Error in HeadingTag: " + ioe);
  }
  return(EVAL_BODY_INCLUDE); // Include tag body
}
```


Including Tag Body: HeadingTag.java (Continued)

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

Using Tag Body: The Tag Library Descriptor File

- **Only difference is body-content element**
  - Should be **JSP** instead of **empty**:

```
<tag>
  <name>…</name>
  <tag-class>…</tag-class>
  <body-content>JSP</body-content>
  <description>…</description>
</tag>
```

- **Purpose is primarily for development environments**
  - Advanced IDEs may have drag-and-drop support for custom tags
  - Defined as optional in JSP 1.2 specification
    - But *not* optional in JSP 2

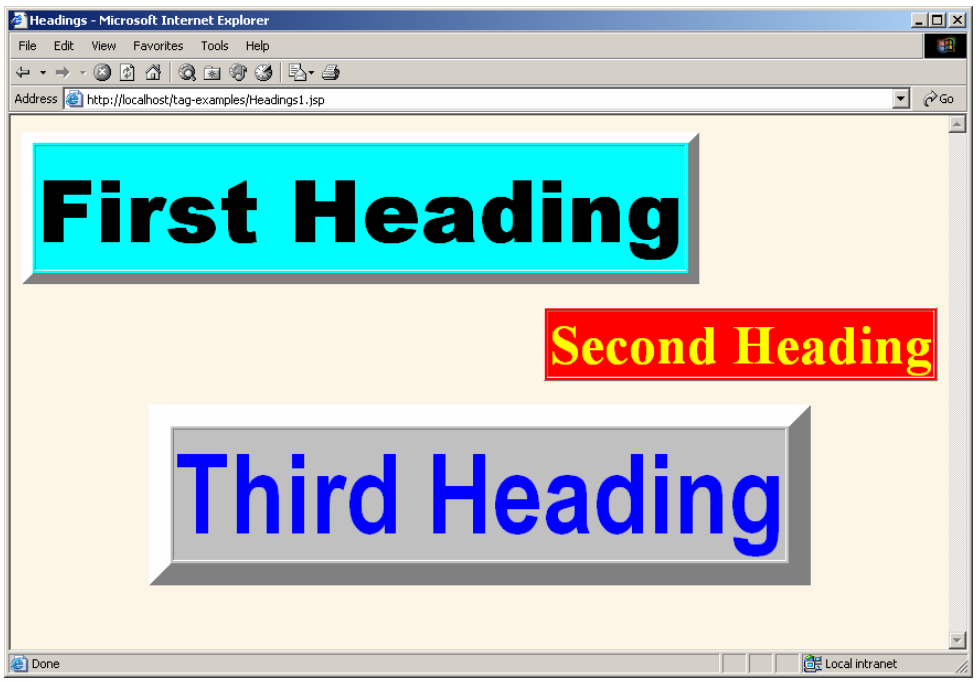
heading Tag: TLD File

```
<tag>
  <name>heading</name>
  <tag-class>coreservlets.tags.HeadingTag</tag-class>
  <body-content>JSP</body-content>
  <description>
    Outputs a 1-cell table used as a heading.
  </description>
  <attribute>
    <name>align</name>
    <required>true</required>
  </attribute>
  <attribute>
    <name>bgColor</name>
    <required>true</required>
  </attribute>
  <attribute>
    <name>border</name>
    <required>true</required>
  </attribute>
  …
</tag>
```

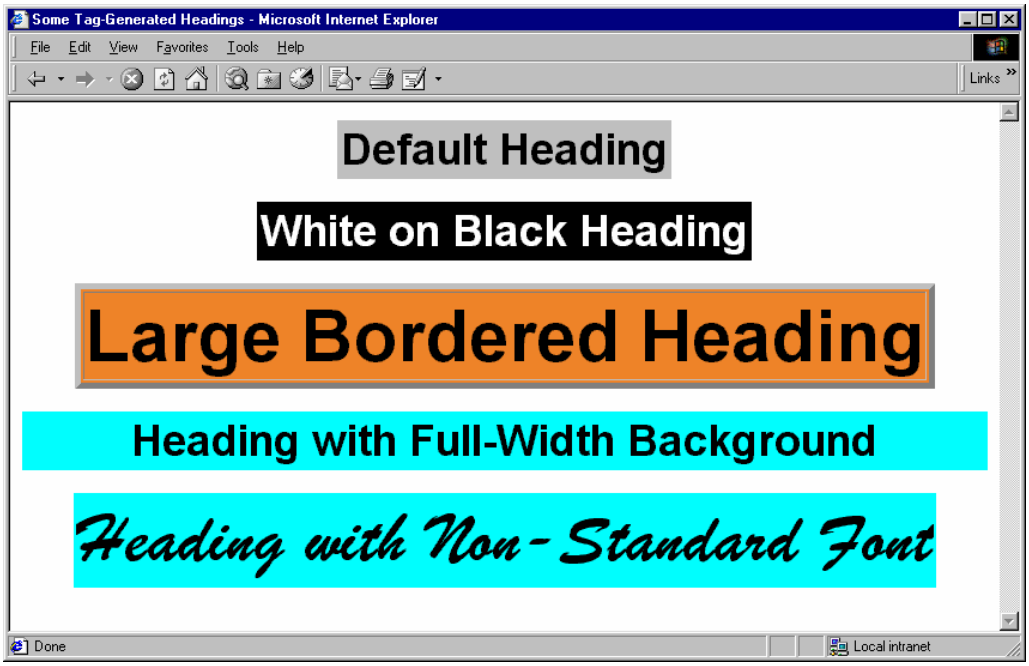
Using heading Tag

```
…<BODY>
<%@ taglib uri="/WEB-INF/csajsp-taglib.tld"
           prefix="csajsp" %>
<csajsp:heading align="LEFT" bgColor="CYAN"
                border="10" fgColor="BLACK"
                font="Arial Black" size="78">
  First Heading
</csajsp:heading>
<csajsp:heading align="RIGHT" bgColor="RED"
                border="1" fgColor="YELLOW"
                font="Times New Roman" size="50">
  Second Heading
</csajsp:heading>
<csajsp:heading align="CENTER" bgColor="#C0C0C0"
                border="20" fgColor="BLUE"
                font="Arial Narrow" size="100">
  Third Heading
</csajsp:heading>
```


Using heading Tag: Results



Using heading Tag: More Results



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

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);
    }
  }}
```

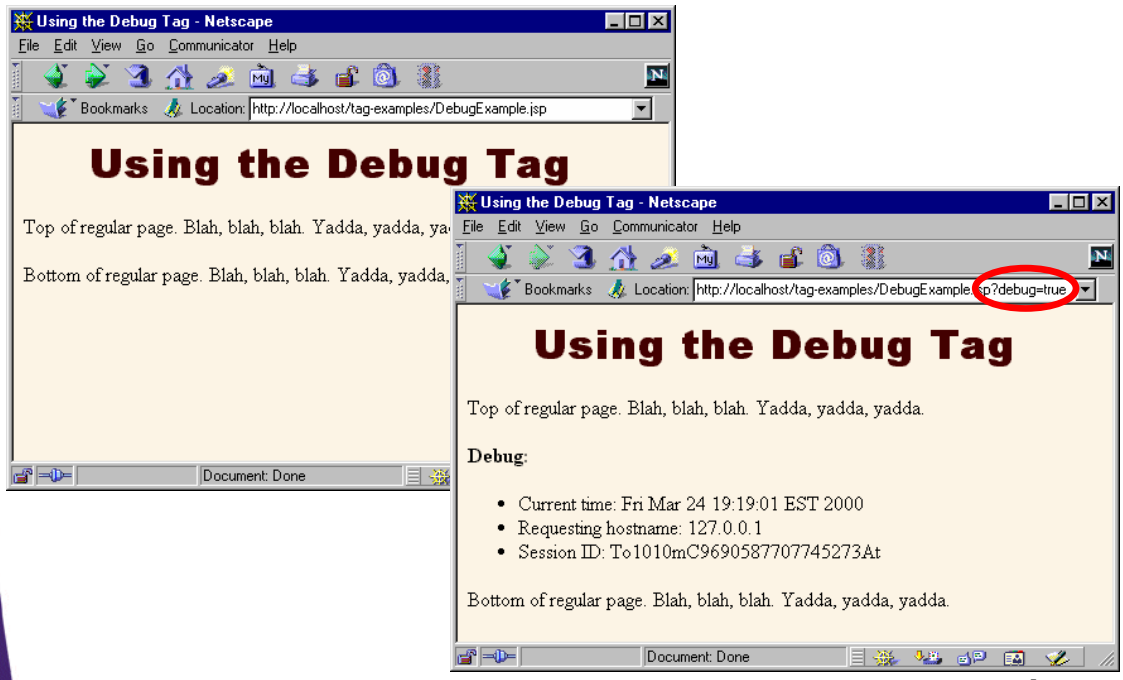

TLD File for DebugTag

```
...
<taglib>
  <tag>
    <name>debug</name>
    <tag-class>
      coreservlets.tags.DebugTag
    </tag-class>
    <description>
      Includes body only if debug param is set.
    </description>
  </tag>
</taglib>
```

Using debug Tag

```
<%@ taglib uri="/WEB-INF/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.
```


Using debug Tag: Results



Summary

- **For each custom (classic) tag, you need**
  - A tag handler class (usually extending TagSupport)
  - 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 set*AttributeName* method. Update TLD file.
- **Body content**
  - doStartTag returns EVAL_BODY_INCLUDE
  - Add doEndTag method

Questions?