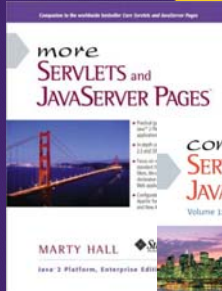




# Creating Custom JSP Tag Libraries: The Basics

Originals of Slides and Source Code for Examples:  
<http://www.coreservlets.com/JSF-Tutorial/>

**Customized J2EE Training:** <http://courses.coreservlets.com/>  
Servlets, JSP, Struts, JSF/MyFaces, Hibernate, Ajax, GWT, Java 5, Java 6, etc. Ruby/Rails coming soon.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



For live JSF training, please see training courses 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.

- Courses developed and taught by Marty Hall
  - Java 5, Java 6, intermediate/beginning servlets/JSP, advanced servlets/JSP, Struts, JSF, Ajax, GWT, custom courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Spring, Hibernate, EJB3, Ruby/Rails


# Agenda

- **Java-based tags**
  - Components of a tag library
  - Basic tags
  - Tags that use attributes
  - Tags that use body content
  - Tags that optionally use body content
- **JSP-based tags (tag files)**
  - Components of a tag library
  - Basic tags
  - Tags that use attributes
  - Tags that use body content

5

J2EE training: <http://courses.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
    - Servlet/JSP combo (MVC)
    - MVC with JSP expression language
  - **Custom tags**
  - **MVC with beans, custom tags, and a framework like Struts or JSF**

6

J2EE training: <http://courses.coreservlets.com>

# Components That Make Up a Tag Library

- **The Tag Handler Class**
  - Java code that says what to output
  - Must implement `javax.servlet.jsp.tagext.SimpleTag`
  - Usually extends `SimpleTagSupport`
  - 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 under `WEB-INF`
- **The JSP File**
  - Imports a tag library (referencing URL of descriptor file)
  - Defines tag prefix
  - Uses tags

7

J2EE training: <http://courses.coreservlets.com>

# Defining a Simple Tag Handler Class

- **Extend the `SimpleTagSupport` class**
- **Import needed packages**
  - `import javax.servlet.jsp.*;`
  - `import javax.servlet.jsp.tagext.*;`
  - `import java.io.*;`
- **Override `doTag`**
  - Obtain the `JspWriter` with `getJspContext().getOut()`
  - Use the `JspWriter` to generate output
  - Code gets called at *request* time
    - Tag instances are not reused like servlet instances, so no worry about race conditions, even if you have instance variables

8

J2EE training: <http://courses.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.Primes;

public class SimplePrimeTag extends SimpleTagSupport {
    protected int length = 50;

    public void doTag() throws JspException, IOException {
        JspWriter out = getJspContext().getOut();
        BigInteger prime =
            Primes.nextPrime(Primes.random(length));
        out.print(prime);
    }
}
```

9

J2EE training: <http://courses.coreservlets.com>

## Defining a Simple Tag Library Descriptor

- Start with XML header
- Top-level element is `taglib`
  - Just use `tlib-version` and `short-name` as in example
- Each tag defined by `tag` element with:
  - **description**, which gives short info. Optional.
  - **name**, which defines the base tag name.
  - **tag-class**, which gives the fully qualified class name of the tag handler.
  - **body-content**, which specifies if tag is standalone or contains content between start and end tag.
- You can have multiple `tag` entries in each TLD file
- Put TLD file somewhere under WEB-INF

10

J2EE training: <http://courses.coreservlets.com>

# TLD File for SimplePrimeTag

```
<?xml version="1.0" encoding="UTF-8" ?>
<taglib 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
  http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
  version="2.0">
  <tlib-version>1.0</tlib-version>
  <short-name>csajsp-taglib</short-name>

  <tag>
    <description>Outputs 50-digit primes</description>
    <name>simplePrime</name>
    <tag-class>coreservlets.tags.SimplePrimeTag</tag-class>
    <body-content>empty</body-content>
  </tag>
  ..
</taglib>
```

- **Don't memorize XML header and standard part; download and modify online version**
  - The important thing is to know how to write **tag** entries
  - Place TLD file somewhere under WEB-INF

11

J2EE training: <http://courses.coreservlets.com>

# Accessing Custom Tags From JSP Files

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

12

J2EE training: <http://courses.coreservlets.com>

# Using simplePrime Tag

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

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

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

</BODY>
</HTML>
```

13

J2EE training: <http://courses.coreservlets.com>

# Using simplePrime Tag: Result



14

J2EE training: <http://courses.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

15

J2EE training: <http://courses.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);  
}
```

16

J2EE training: <http://courses.coreservlets.com>

## Attributes: PrimeTag.java

```
package coreservlets.tags;

public class PrimeTag extends SimplePrimeTag {
    public void setLength(String length) {
        try {
            this.length = Integer.parseInt(length);
        } catch (NumberFormatException nfe) {
            this.length = 50;
        }
    }
}
```

17

J2EE training: <http://courses.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.
  - **required**, a required element that stipulates whether the attribute must always be supplied (true) or is optional (false).
  - **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.

18

J2EE training: <http://courses.coreservlets.com>

# TLD File for PrimeTag

```
...
<taglib ...>
  ...
  <tag>
    <description>Outputs an N-digit prime</description>
    <name>prime</name>
    <tag-class>coreservlets.tags.PrimeTag</tag-class>
    <body-content>empty</body-content>
    <attribute>
      <name>length</name>
      <required>false</required>
    </attribute>
  </tag>
  ...
</taglib>
```

19

J2EE training: <http://courses.coreservlets.com>

# Using prime Tag

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

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

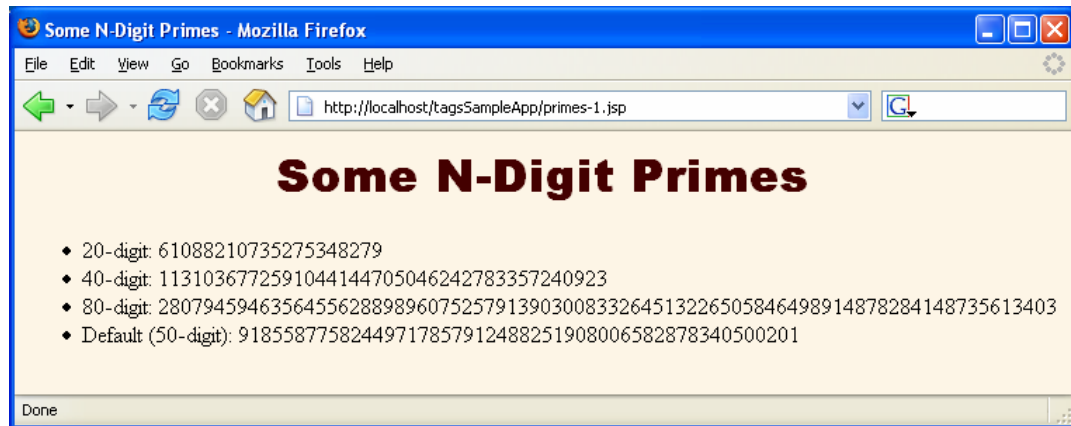
<UL>
  <LI>20-digit: <csajsp:prime length="20" />
  <LI>40-digit: <csajsp:prime length="40" />
  <LI>80-digit: <csajsp:prime length="80" />
  <LI>Default (50-digit): <csajsp:prime />
</UL>

</BODY>
</HTML>
```

20

J2EE training: <http://courses.coreservlets.com>

## Using prime Tag: Result



21

J2EE training: <http://courses.coreservlets.com>

## Including the Tag Body

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

22

J2EE training: <http://courses.coreservlets.com>

## Including Tag Body: The Tag Handler Class

- Call `getJspBody().invoke(null);`

```
public void doTag() throws ... {
    JspWriter out = getJspContext().getOut();
    out.print("...");
    getJspBody().invoke(null);
    out.print("...");
}
```

## Including Tag Body: HeadingTag.java

```
public class HeadingTag extends SimpleTagSupport {
    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 void doTag() throws JspException, IOException {
    JspWriter out = getJspContext().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");
    // Output content of the body
    getJspBody().invoke(null);
    out.println("</SPAN></TH></TR></TABLE>" +
            "<BR CLEAR=\""ALL\"><BR>");
}
```

25

J2EE training: <http://courses.coreservlets.com>

## Using Tag Body: The Tag Library Descriptor File

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

```
<tag>
    <name>...</name>
    <tag-class>...</tag-class>
    <body-content>scriptless</body-content>
</tag>
```
- **Legal values for body-content**
  - **empty**: no body content
    - Body content is ignored even if supplied
  - **scriptless**: body content is included
    - Can contain plain text, EL elements, and page directives only
    - No explicit scripting allowed (<%= ... %>)
  - **tagdependent**: body content is processed by tag

26

J2EE training: <http://courses.coreservlets.com>

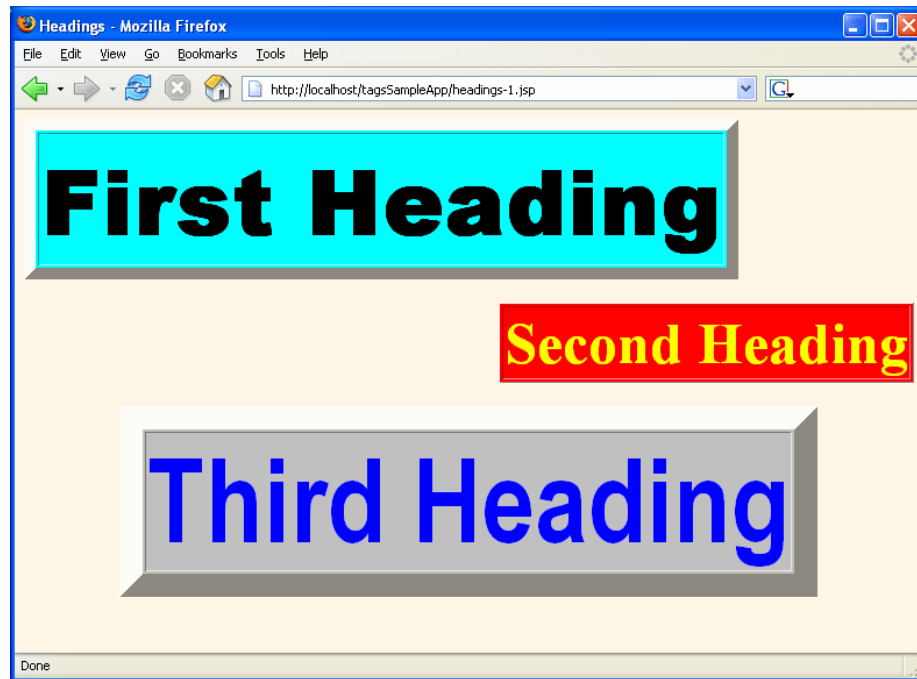
# heading Tag: TLD File

```
<tag>
  <description>Formats enclosed heading</description>
  <name>heading</name>
  <tag-class>coreservlets.tags.HeadingTag</tag-class>
  <body-content>scriptless</body-content>
  <attribute>
    <name>align</name>
    <required>true</required>
  </attribute>
  <attribute>
    <name>bgColor</name>
    <required>true</required>
  </attribute>
  ...
</tag>
```

# Using heading Tag

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

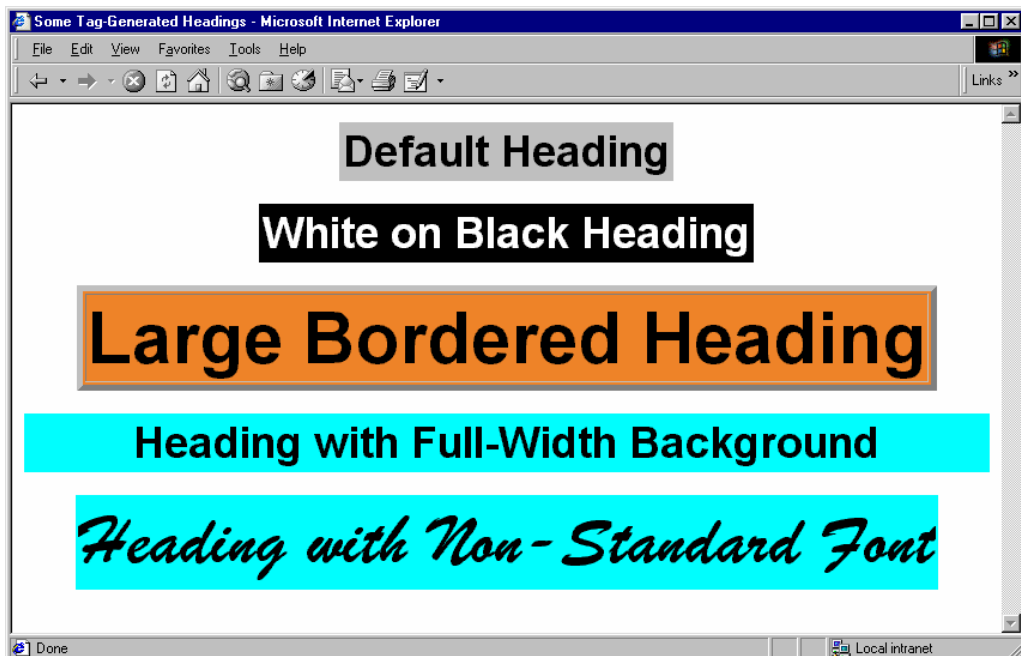
# Using heading Tag: Results



29

J2EE training: <http://courses.coreservlets.com>

# Using heading Tag: More Results



30

J2EE training: <http://courses.coreservlets.com>

## Optional Tag Bodies

- **First examples had no tag bodies**
  - body-content: empty
  - doTag does not call invoke(null)
- **Most recent examples always included tag bodies**
  - body-content: scriptless
  - doTag calls invoke(null)
- **Now: decide at request time whether or not to include tag body**
  - body-content: scriptless
  - doTag conditionally calls invoke(null)
    - Depending on run-time information

31

J2EE training: <http://courses.coreservlets.com>

## Optional Tag Bodies: DebugTag.java

```
public class DebugTag extends SimpleTagSupport {
    public void doTag() throws JspException, IOException {
        PageContext context = (PageContext)getJspContext();
        HttpServletRequest request =
            (HttpServletRequest)context.getRequest();
        // Output body of tag only if debug param is present.
        if (request.getParameter("debug") != null) {
            getJspBody().invoke(null);
        }
    }
}
```

32

J2EE training: <http://courses.coreservlets.com>

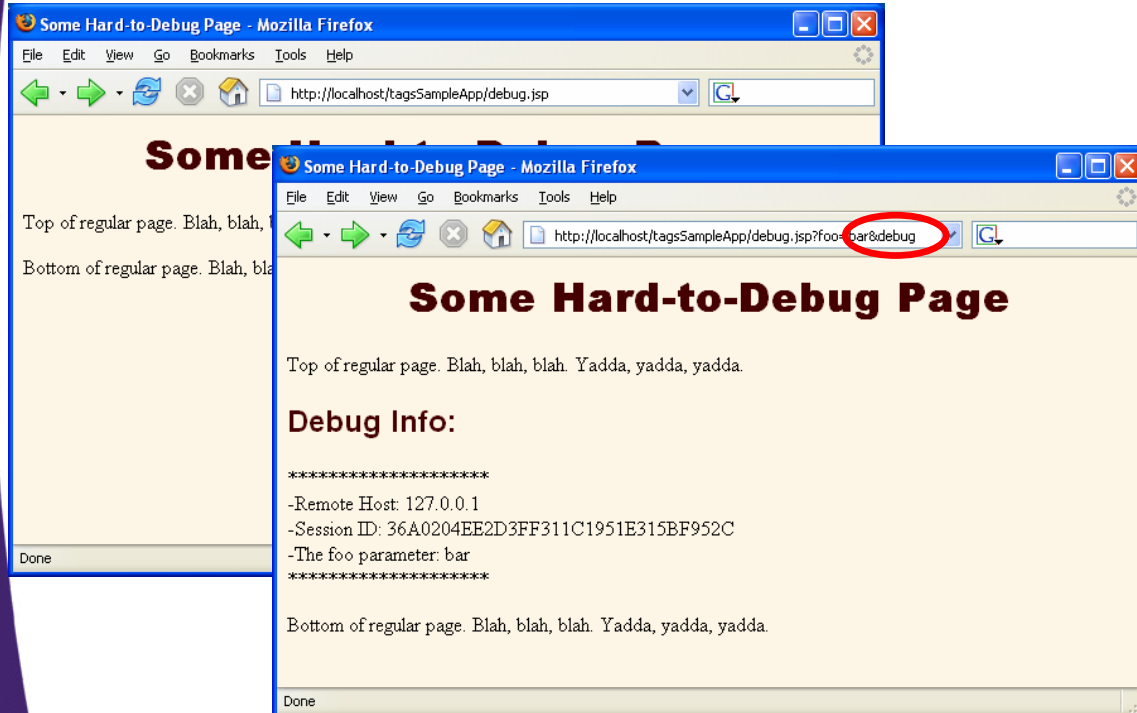
# TLD File for DebugTag

```
...
<tag>
  <description>
    Conditionally outputs enclosed body
  </description>
  <name>debug</name>
  <tag-class>coreservlets.tags.DebugTag</tag-class>
  <body-content>scriptless</body-content>
</tag>
...
```

# Using debug Tag

```
<%@ taglib uri="/WEB-INF/tlds/csajsp-taglib.tld"
      prefix="csajsp" %>
Top of regular page. Blah, blah, blah.
Yadda, yadda, yadda.
<csajsp:debug>
<H2>Debug Info:</H2>
*****<BR>
-Remote Host: ${pageContext.request.remoteHost}<BR>
-Session ID: ${pageContext.session.id}<BR>
-The foo parameter: ${param.foo}<BR>
*****<BR>
</csajsp:debug>
<P>
Bottom of regular page. Blah, blah, blah.
Yadda, yadda, yadda.
```

# Using debug Tag: Results



35

# The <uri> Tag in TLD Files

- **TLD files can define fake addresses**

```
<?xml version="1.0" encoding="UTF-8" ?>
<taglib ...>
...
<uri>http://anything/you/want</uri>
```
- **JSP pages import TLD using fake address**

```
<%@ taglib uri="http://anything/you/want" ...%>
```

  - Note: this address is totally made up: it just matches what is in the TLD file. JSP page does not connect to the Web to look for this address.
- **Advantages**
  - Can move/rename TLD file with no JSP code changes
    - You can even bundle tag libraries in JAR files under WEB-INF/lib and put TLD files in META-INF in the JAR files
- **Disadvantages**
  - Confusing: JSP authors don't know where TLD file is

36

# Tag Files: Custom Tags Using JSP Syntax

- **Two Approaches**
  - When there is lots of logic, use Java to create output
    - Analagous to when you use servlets
  - When there is lots of formatting, use JSP to create output
    - Analagous to when you use JSP pages
- **Pros**
  - Very good for complex text formatting
  - Very concise
- **Cons**
  - Not good for complicated logic
  - Runs only in JSP 2.0
    - Java-based versions had "classic" syntax that worked in older servers (e.g., BEA WebLogic 8.1, Oracle 9i AS)

37

J2EE training: <http://courses.coreservlets.com>

# Simple Standalone Tags

- **Java-based approach requires three pieces**
  - Java code that overrides doTag to generate output
    - Strengths and weaknesses generally similar to those of servlets, but more cumbersome
  - Tag Library Descriptor (TLD) file that maps Java class name to tag name
  - JSP page that refers to specific location of TLD file
- **JSP-based approach requires two pieces**
  - JSP code (tag file) that shows result
    - `/WEB-INF/tags/someName.tag`
  - No TLD file: tag name taken from tag-file name
  - JSP page that refers to directory containing tag file
    - `/WEB-INF/tags` or a subdirectory thereof

38

J2EE training: <http://courses.coreservlets.com>

# Tag Files

- **Look just like regular JSP files, except**
  - Must be located in (or under) WEB-INF/tags
  - Must be named blah.tag, not blah.jsp
  - You use `<%@ tag ... %>` instead of `<%@ page ... %>`
  - You use predefined variable `jspContext` instead of `pageContext`
    - But you can cast it to `PageContext`
    - Other variables (`request`, `response`, etc.) are the same
- **Example**

WEB-INF/tags/ <b>date</b> .tag	some-page.jsp
<pre>&lt;%@ tag import="java.util.*" %&gt; Date is &lt;%= new Date() %&gt;</pre>	<pre>&lt;%@ taglib tagdir="/WEB-INF/tags" prefix="test" %&gt; ... &lt;test:<b>date</b>/&gt;</pre>

39

J2EE training: <http://courses.coreservlets.com>

# Java-Based Tags: Code (Simple Standalone Tag)

```
package coreservlets.tags;

import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
import java.math.*;
import coreservlets.Primes;

public class SimplePrimeTag extends SimpleTagSupport {
    protected int length = 50;

    public void doTag() throws JspException, IOException {
        JspWriter out = getJspContext().getOut();
        BigInteger prime =
            Primes.nextPrime(Primes.random(length));
        out.print(prime);
    }
}
```

## Java-Based Tags: TLD File (Simple Standalone Tag)

```
<?xml version="1.0" encoding="UTF-8" ?>
<taglib 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
  http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
  version="2.0">
  <tlib-version>1.0</tlib-version>
  <short-name>csajsp-taglib</short-name>

  <tag>
    <description>Outputs 50-digit primes</description>
    <name>simplePrime</name>
    <tag-class>coreservlets.tags.SimplePrimeTag</tag-class>
    <body-content>empty</body-content>
  </tag>
  ...
</taglib>
```

## Java-Based Tags: Usage in JSP (Simple Standalone Tag)

```
...
<BODY>
<H1>Some 50-Digit Primes</H1>
<%@ taglib uri="/WEB-INF/tlds/csajsp-taglib.tld"
  prefix="csajsp" %>
<UL>
  <LI><csajsp:simplePrime />
  <LI><csajsp:simplePrime />
  <LI><csajsp:simplePrime />
  <LI><csajsp:simplePrime />
</UL>
</BODY></HTML>
```

## Java-Based Tags: Result (Simple Standalone Tag)



43

J2EE training: <http://courses.coreservlets.com>

## Tag Files: Tag Code (Simple Standalone Tag)

- **WEB-INF/tags/simplePrime2.tag**
  - Directory name is not arbitrary; must be in /WEB-INF/tags or a subdirectory thereof

```
<%= coreservlets.Primes.nextPrime  
    (coreservlets.Primes.random(50)) %>
```

44

J2EE training: <http://courses.coreservlets.com>

# Tag Files: Usage in JSP (Simple Standalone Tag)

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

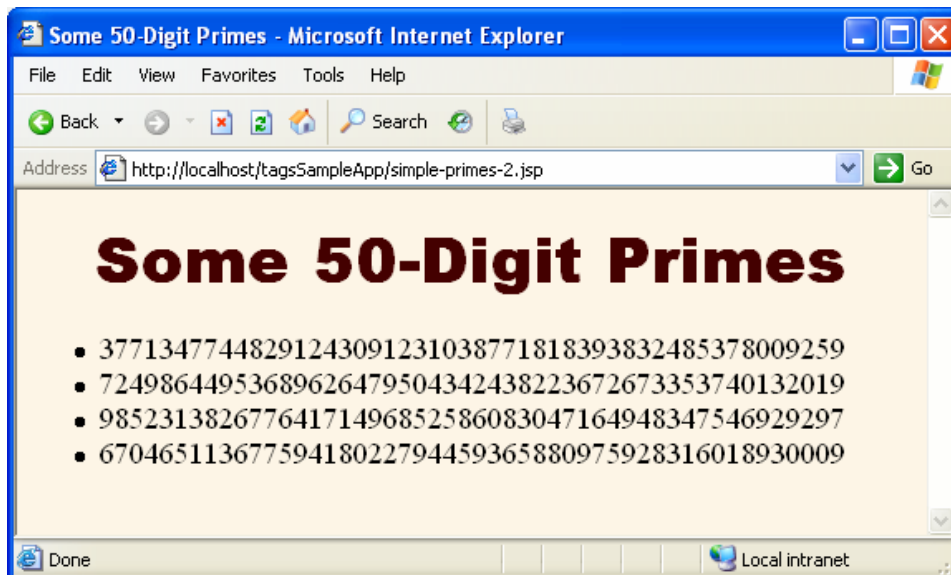
<%@ taglib tagdir="/WEB-INF/tags"
           prefix="csajsp" %>

<UL>
  <LI><csajsp:simplePrime2 />
  <LI><csajsp:simplePrime2 />
  <LI><csajsp:simplePrime2 />
  <LI><csajsp:simplePrime2 />
</UL>
</BODY></HTML>
```

45

J2EE training: <http://courses.coreservlets.com>

# Tag Files: Result (Simple Standalone Tag)



46

J2EE training: <http://courses.coreservlets.com>

# Tags with Attributes

- **Java-based tags**
  - For each attribute, add `setAttributeName` method to the Java class
  - Attribute value usually explicitly stored in instance variable (field) of Java class
  - List each attribute explicitly in TLD file
- **JSP-based tags (tag files)**
  - Each attribute listed in tag file with `@attribute`
    - Can also list `required` and `rtexprvalue` (default false)
  - Attribute value automatically stored in scoped variable (for access from expression language) and in local variable (for access from Java code)
  - No TLD file

47

J2EE training: <http://courses.coreservlets.com>

# Java-Based Tags: Code (Tag with Attributes)

```
package coreservlets.tags;

public class PrimeTag extends SimplePrimeTag {
    public void setLength(String length) {
        try {
            this.length = Integer.parseInt(length);
        } catch (NumberFormatException nfe) {
            this.length = 50;
        }
    }
}
```

48

J2EE training: <http://courses.coreservlets.com>

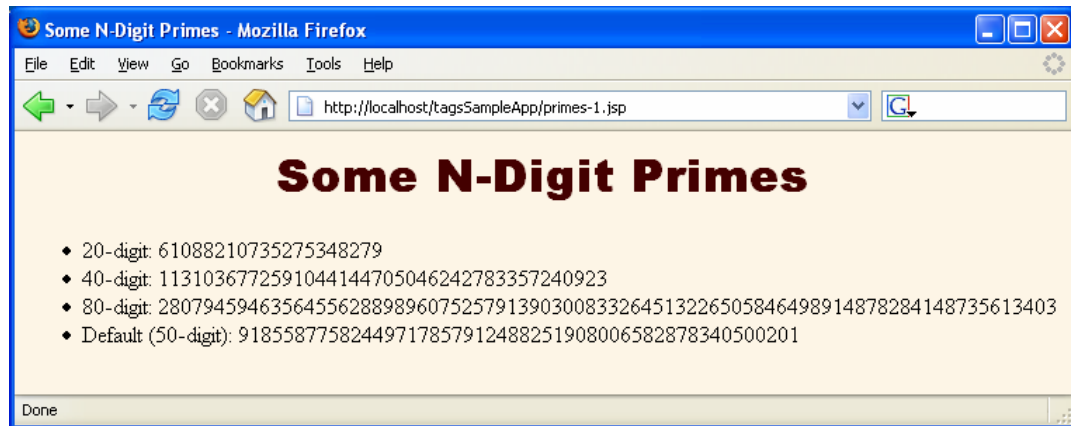
## Java-Based Tags: TLD File (Tag with Attributes)

```
...  
<tag>  
  <description>Outputs an N-digit prime</description>  
  <name>prime</name>  
  <tag-class>coreservlets.tags.PrimeTag</tag-class>  
  <body-content>empty</body-content>  
  <attribute>  
    <name>length</name>  
    <required>>false</required>  
  </attribute>  
</tag>  
...
```

## Java-Based Tags: Usage in JSP (Tag with Attributes)

```
...  
<BODY>  
<H1>Some N-Digit Primes</H1>  
<%@ taglib uri="/WEB-INF/tlds/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>
```

# Java-Based Tags: Results (Tag with Attributes)



51

J2EE training: <http://courses.coreservlets.com>

# Tag Files: Tag Code (Tag with Attributes)

```
<%@ attribute name="length" required="false" %>
<%
int len = 50;
try {
    len = Integer.parseInt(length);
} catch(NumberFormatException nfe) {}
%>
<%= coreservlets.Primes.nextPrime
    (coreservlets.Primes.random(len)) %>
```

52

J2EE training: <http://courses.coreservlets.com>

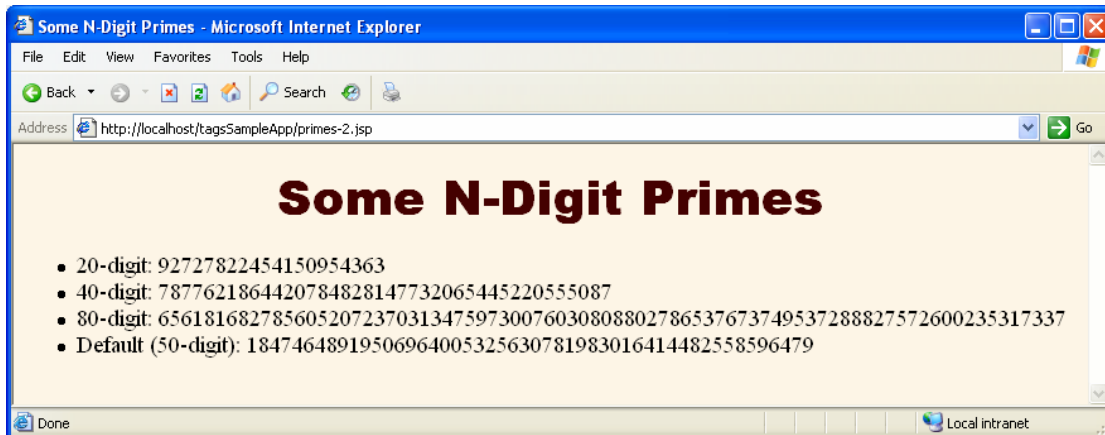
# Tag Files: Usage in JSP (Tag with Attributes)

```
...  
<BODY>  
<H1>Some N-Digit Primes</H1>  
<%@ taglib tagdir="/WEB-INF/tags"  
           prefix="csajsp" %>  
  
<UL>  
  <LI>20-digit: <csajsp:prime2 length="20" />  
  <LI>40-digit: <csajsp:prime2 length="40" />  
  <LI>80-digit: <csajsp:prime2 length="80" />  
  <LI>Default (50-digit): <csajsp:prime2 />  
</UL>  
</BODY></HTML>
```

53

J2EE training: <http://courses.coreservlets.com>

# Tag Files: Results (Tag with Attributes)



54

J2EE training: <http://courses.coreservlets.com>

# Tags with Bodies

- **Java-based tags**
  - Change body-content from empty to scriptless (in TLD)
  - Call `getJspBody().invoke(null)`
  - Still need setter method and TLD entry for every attribute
- **JSP-based tags (tag files)**
  - Use `<jsp:doBody/>` to output tag body
  - No major syntax changes
  - Access to attributes still much simpler

55

J2EE training: <http://courses.coreservlets.com>

# Java-Based Tags: Code (Tag with Body)

```
public class HeadingTag extends SimpleTagSupport {
    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;
    }
    ...
}
```

56

J2EE training: <http://courses.coreservlets.com>

## Java-Based Tags: Code (Tag with Body -- Continued)

```
public void doTag() throws JspException, IOException {
    JspWriter out = getJspContext().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"); //
    // Output content of the body
    getJspBody().invoke(null);
    out.println("</SPAN></TH></TR></TABLE>" +
        "<BR CLEAR=\""ALL\"><BR>");
}
}
```

57

J2EE training: <http://courses.coreservlets.com>

## Java-Based Tags: TLD File (Tag with Body)

```
...
<tag>
  <description>Formats enclosed heading</description>
  <name>heading</name>
  <tag-class>coreservlets.tags.HeadingTag</tag-class>
  <body-content>scriptless</body-content>
  <attribute>
    <name>align</name>
    <required>true</required>
  </attribute>
  <attribute>
    <name>bgColor</name>
    <required>true</required>
  </attribute>
  ...
</tag>
```

58

J2EE training: <http://courses.coreservlets.com>

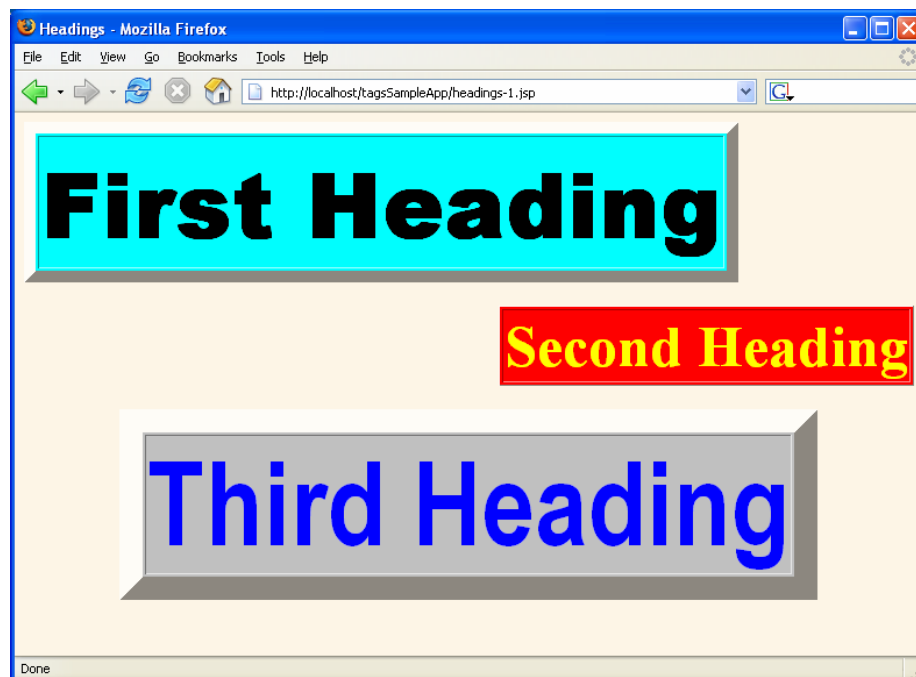
# Java-Based Tags: Usage in JSP (Tag with Body)

```
<BODY>
<%@ taglib uri="/WEB-INF/tlds/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>
```

59

J2EE training: <http://courses.coreservlets.com>

# Java-Based Tags: Results (Tag with Body)



60

J2EE training: <http://courses.coreservlets.com>

## Tag Files: Tag Code (Tag with Body)

```
<%@ attribute name="align" required="true" %>
<%@ attribute name="bgColor" required="true" %>
<%@ attribute name="border" required="true" %>
<%@ attribute name="fgColor" required="true" %>
<%@ attribute name="font" required="true" %>
<%@ attribute name="size" required="true" %>
<TABLE ALIGN="${align}"
        BGCOLOR="${bgColor}"
        BORDER="${border}">
  <TR><TH>
    <SPAN STYLE="color: ${fgColor};
                font-family: ${font};
                font-size: ${size}px;">
      <jsp:doBody/></SPAN>
  </TR>
</TABLE><BR CLEAR="ALL"><BR>
```

61

J2EE training: <http://courses.coreservlets.com>

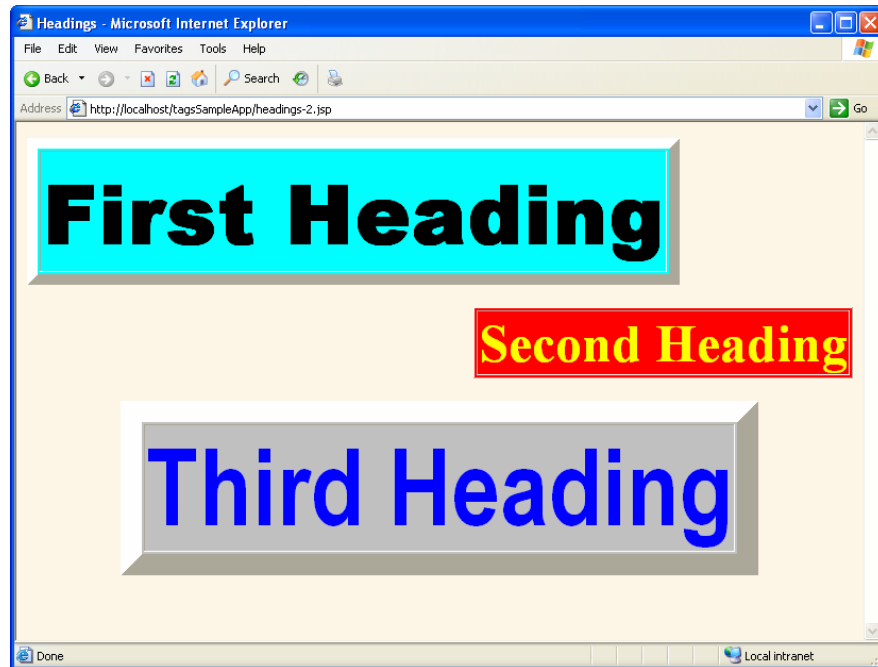
## Tag Files: Usage in JSP (Tag with Body)

```
<BODY>
<%@ taglib tagdir="/WEB-INF/tags"
           prefix="csajsp" %>
<csajsp:heading2 align="LEFT" bgColor="CYAN"
                 border="10" fgColor="BLACK"
                 font="Arial Black" size="78">
  First Heading
</csajsp:heading2>
<csajsp:heading2 align="RIGHT" bgColor="RED"
                 border="1" fgColor="YELLOW"
                 font="Times New Roman" size="50">
  Second Heading
</csajsp:heading2>
<csajsp:heading2 align="CENTER" bgColor="#C0C0C0"
                 border="20" fgColor="BLUE"
                 font="Arial Narrow" size="100">
  Third Heading
</csajsp:heading2>
```

62

J2EE training: <http://courses.coreservlets.com>

## Tag Files: Results (Tag with Body)



63

J2EE training: <http://courses.coreservlets.com>

## Open Source Tag Libraries

- **JSP Standard Tag Library (JSTL)**
  - Covered in later lecture
- **AjaxTags**
  - <http://ajaxtags.sourceforge.net/>
- **Jakarta Taglibs**
  - <http://jakarta.apache.org/taglibs/>
    - Benchmarking
    - Internationalization (I18N)
    - Database access
    - Sending email
    - Populating/validating form fields
    - Regular expressions
    - Extracting data from other Web pages
    - Logging
    - Etc.

64

J2EE training: <http://courses.coreservlets.com>

# Summary

- **For each custom (Java-based) tag, you need**
  - A tag handler class (usually extending SimpleTagSupport)
  - An entry in a Tag Library Descriptor file
  - A JSP file that imports it, specifies prefix, and uses it
- **Simple tags**
  - Generate output in doTag, body-content is empty
- **Attributes**
  - Define setAttribute*Name* method.
  - Add attribute entries to TLD file.
- **Body content**
  - body-content is scriptless
  - doTag calls getJspBody().invoke(null)
- **Tag Files**
  - In *tagName.tag*, outline what result should look like
    - Refer to attributes with `${attributeName}`
    - Refer to tag body with `<jsp:doBody/>`

65

J2EE training: <http://courses.coreservlets.com>

© 2007 Marty Hall



## Questions?

Customized J2EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, Struts, JSF/MyFaces, Hibernate, Ajax, GWT, Java 5, Java 6, etc. Ruby/Rails coming soon.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

66