Custom Tags

When JSTL and HTML aren't enough

EHSDI

eBuzima

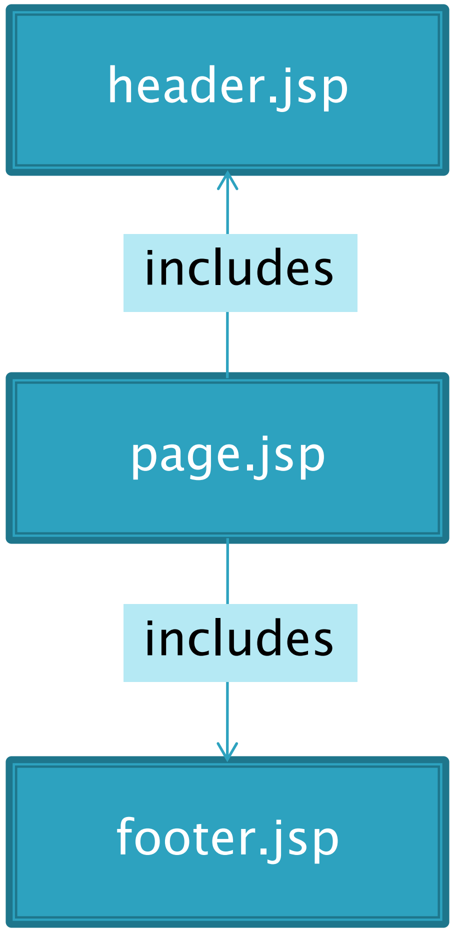
Reusable components

- We can do a lot with EL and JSTL
  - EL provides the access to data and attributes
  - JSTL provides some logic and flow control
- What if we need a web component with complex Java code just to display it?
- What if we have a web component that we want to include many times on page?
- What if we want a web designer to be able to customize such a component with having to edit any Java?

The include directive

- We've already used this to include JSP files within another JSP file
- The content from the included files is added to the JSP at *translation* time

```
<%@ include file="header.jsp" %>
```



The include directive

JSP:
example.jsp

```
<div>The content to include</div>
```

JSP:
page.jsp

```
<p>Before the include</p>
<%@ include file="example.jsp" %>
<p>After the include</p>
```

Servlet:
page_jsp.java

```
out.write("<p>Before the include<p>");

out.write("<div>The content to
include<div>");

out.write("<p>After the include</p>");
```

EHSDI
eBuzima

<jsp:include>

- This standard action also allows us to include content from other JSP files
- Does the including at *runtime,* i.e. inside the servlet it calls a method to load and include the JSP
  - This means we can modify the included JSP and see the changes in the parent JSP
  - Tomcat is smart enough to know when an included JSP has been changed, and will retranslate the parent JSP
  - You might not always be using Tomcat…

<jsp:include>

JSP:
page.jsp

```
<p>Before the include<p>

<jsp:include page="example.jsp" />

<p>After the include</p>
```

Servlet:
page_jsp.java

```
out.write("<p>Before the include<p>");

JspRuntimeLibrary.include(request,
  response, "example.jsp", out, false);

out.write("<p>After the include</p>");
```

EHSDI
eBuzima

Adding parameters

- The include action allows specifying parameters so that included JSPs can be customized from the parent JSP

page.jsp

```
<jsp:include page="header.jsp">
  <jsp:param name="title" value="Home" />
</jsp:include>
```

header.jsp

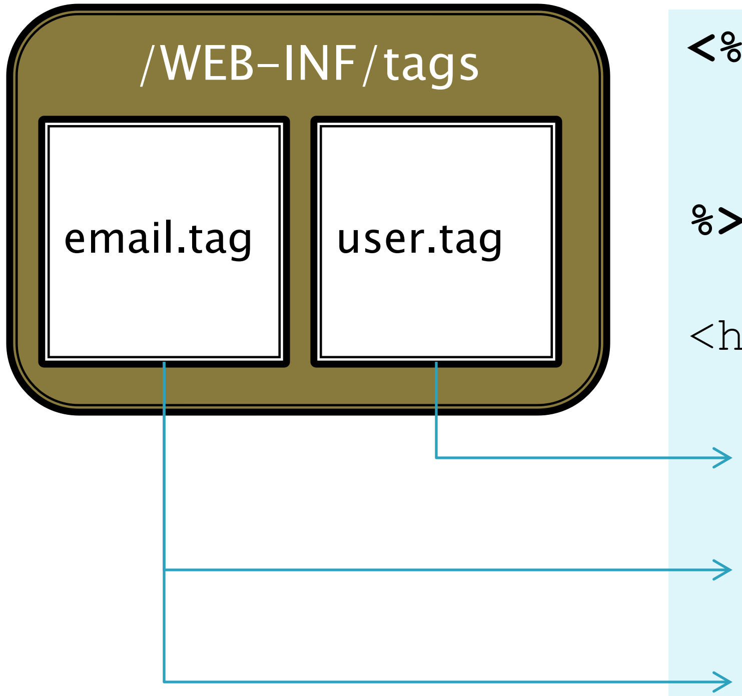
```
<html>
<head>
  <title>${param.title}</title>
</head>
```

EHSDI
eBuzima

Tag files

- These are like JSP includes, but they give us more flexibility, and are more suitable for constructing reusable components
- The tag file becomes a custom tag
- A directory of tag files becomes a tag library which is imported using a taglib directive

Taglibs



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

<html>
  <body>
    <p><ehsdi:email /></p>

    <ehsdi:user />

    <ehsdi:user />

  </body>
</html>
```

EHSDI
eBuzima

Creating tags

- At its simplest a tag file can be just some HTML to be included, e.g.

copyright.tag

```
<%@ tag language="java" %>

<small>Copyright EHSDI 2009</small>
```

page.jsp

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

<ehsdi:copyright />
```

Adding parameters

- JSP includes are customized using request parameters, using `<jsp:param>` tags
- This makes it easy to confuse tag parameters and GET/POST parameters

- Tag files use attributes instead:
  - Use the `attribute` directive inside the tag file
  - This creates attributes whose scope is the tag file
  - The attribute values are set on the tag itself

Tag attributes

copyright.tag

```
<%@ tag language="java" %>
<%@ attribute name="holder" %>

<small>Copyright ${holder}</small>
```

page.jsp

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

<ehsdi:copyright holder="EHSDI 2009" />
```

output...

```
<small>Copyright EHSDI 2009<small>
```

EHSDI eBuzima

Attribute directive

- This allows us to specify the properties of a tag attribute:
  - **name**: the name of the attribute
  - **required**: whether or not attribute is required for this tag
  - **rtexprvalue**: whether or not attribute value can be an EL expression

```
<%@ attribute name="holder" required="true" %>
```

Tags with bodies

- A tag body is anything between the start and end tag, e.g.

page.jsp

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

<ehsdi:msg>This is the body</ehsdi:msg>
```

- The body content is displayed using `<jsp:doBody/>`, e.g.

msg.tag

```
<%@ tag language="java" %>

<i>MSG: <jsp:doBody/></i>
```

EHSDI
eBuzima

Objects as parameters

- Attribute values don't have to be strings – they can be any Java object
- We can specify the object type in the attribute directive, to prevent the wrong type of object being passed, e.g.

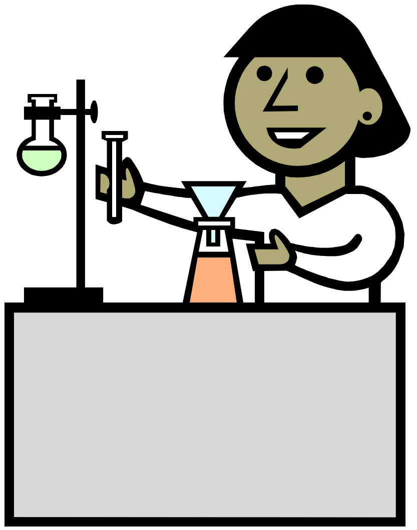
```
<%@ attribute name="items" type="java.util.Collection" %>

Collection size is: ${fn:length(items)}
```

A JSTL function which returns the length of a collection

Tag classes

- Sometimes the logic for displaying a tag is too complex for JSTL and EL
- We can create a Java class which displays a tag by extending one of the *tag handler* classes in the API



EHSDI
eBuzima

Simple vs Classic

- There are two types of tag handler class in the API:
- **Simple**: these are the newer, JSP 2.0 way of creating custom tags
- **Classic**: the older way – sometimes necessary to resort to these, but mainly you only have to work with these in older code

EHSDI
eBuzima

Creating a 'Simple' custom tag

- Create a new class that extends `SimpleTagSupport` (requires *jsp-api.jar*)
- Override the `doTag()` method, to output some HTML, etc
- Create a *Tag Library Descriptor* (TLD) file and add a definition for your new tag
- Place the TLD in `/WEB-INF/`
- Import the tag library into a JSP page using the `taglib` directive

Example: the handler class

```
import javax.servlet.jsp.tagext.SimpleTagSupport;

public class FirstTag extends SimpleTagSupport {

  public void doTag() throws JspException, IOException {
    JspWriter out = getJspContext().getOut();

    out.write("<b>My First Tag</b>");
  }
}
```

Example: the TLD

```
<taglib ...>
  <description>EHSDI tag library</description>
  <tlib-version>1.0</tlib-version>
  <short-name>EHSDI</short-name>
  <uri>ehsdiTags</uri>

  <tag>
    <description>My first tag</description>
    <name>first</name>
    <tag-class>eh203.custtags.FirstTag</tag-class>
    <body-content>empty</body-content>
  </tag>
</taglib>
```

EHSDI
eBuzima

Example: the JSP

The tag library's **uri** in the TLD

```
<%@ taglib prefix="ehsdi" uri="ehsdiTags" %>

<html>
  <body>
    <ehsdi:first />
  </body>
</html>
```

The tag **name** in the TLD

Example: tag with body

- To create a tag that has a body, we can use `getJspBody()` to get the body

```
<%@ taglib prefix="ehsdi" uri="ehsdiTags" %>
<html>
  <body>
    <ehsdi:first>I've got a body!</ehsdi:first>
  </body>
</html>
```

```
public void doTag() throws JspException, IOException {
  JspWriter out = getJspContext().getOut();
  out.write("<b>");
  getJspBody().invoke(null);
  out.write("</b>");
}
```

Example: tag with body

- … and we must also change the value of `body-content` in the TLD

```
<taglib ...>
  <description>EHSDI tag library</description>
  <tlib-version>1.0</tlib-version>
  <short-name>EHSDI</short-name>
  <uri>ehsdiTags</uri>

  <tag>
    <description>My first tag</description>
    <name>first</name>
    <tag-class>eh203.custtags.FirstTag</tag-class>
    <body-content>scriptless</body-content>
  </tag>
</taglib>
```

Adding attributes

- Attributes are added to the tag as bean properties, e.g.

```
public class FirstTag extends SimpleTagSupport {
   ...

   public String getName() { ... }
   public void setName(String name) { ... }

   public int getValue() { ... }
   public void setValue(int value) { ... }
}
```

```
<ehsdi:myTag name="Hello" value="44" />
```

Example: adding attributes

- … but we must also add the attributes in the TLD

```
<taglib ...>
  ...

  <tag>
    <description>My first tag</description>
    <name>first</name>
    <tag-class>eh203.custtags.FirstTag</tag-class>
    <body-content>scriptless</body-content>
    <attribute>
      <name>value</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
    </attribute>
  </tag>
</taglib>
```

Means attribute is required for this tag

Means attribute value can be an EL expression

Example: adding attributes

```
<ehsdi:myTag value="Hello" />
```

Checks TLD for this attribute

```
<attribute>
  <name>value</name>
  <required>true</required>
  <rtexprvalue>true</rtexprvalue>
</attribute>
```

Sets attribute value as bean property

```
public class FirstTag extends SimpleTagSupport {
  ...
  public void setValue(int value) { ... }
}
```

References

- Books
  - Head First Servlets and JSP (O'Reilly)
- Websites
  - http://java.sun.com/javaee/reference/tutorials/

EHSDI

eBuzima