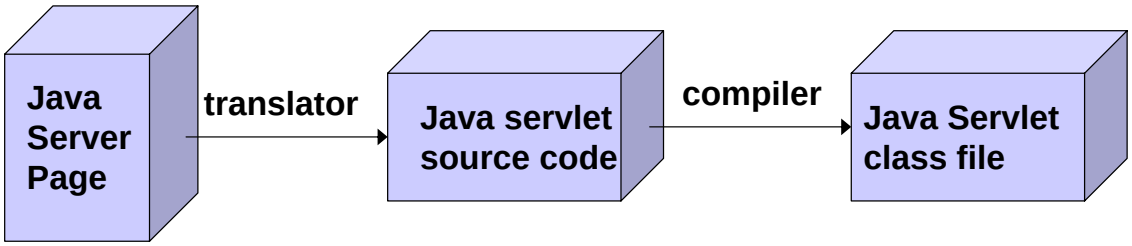
Java Server Pages (JSP)

What is JSP

- JSP simply puts Java inside HTML pages. You can take any existing HTML page and change its extension to ".jsp" instead of ".html".
- Scripting elements are used to provide dynamic pages

JSP and Servlets

- Each JSP page is turned into a Java servlet, compiled and loaded. This compilation happens on the first request. After the first request, the file doesn't take long to load anymore.
- Every time you change the JSP file, it will be re-compiled again.



Generated servlets

- You can examine the source code produced by the JSP translation process.
- There is a directory called *generated* in Sun Java J2EE Application Server where you can find the source code.
- Note that the **`_jspService`** method corresponds to the servlet service method (which is called by **`doGet`** or **`doPost`**)

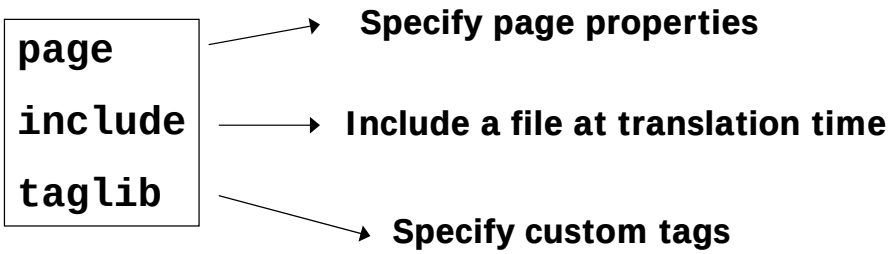
JSP elements (overview)

- Directives of the form **<%@ ... %>**
- Scripting elements
  - Expressions of the form **<%= expr %>**
  - Scriptlets of the form **<% code %>**
  - Declarations of the form **<%! code %>**
  - JSP Comments **<%-- ... --%>**
- Standard actions
  - Example: **<jsp:useBean>** ... **</jsp:useBean>**
- Implicit variables like request, response, out

JSP Directives

- They have the form

```
<%@ name attribute1="...", attribute2="..." ... %>
```



JSP Directive Examples

- Import java packages
  - `<%@ page import="java.util.*,java.sql.*" %>`
- Multiple import statements
  - `<%@ page import="java.util.*" %>`
    `<%@ page import="java.sql.*" %>`
- Including file at ***translation*** time
  - `<%@ include file="header.html" %>`
- For include the path is relative to the JSP file

JSP Scripting Elements: Expressions

- For an expression scripting element like **<%= expr %>**, **expr** is evaluated and the result is converted to a string and placed into the JSP's servlet output stream. In a Java servlet this would be equivalent to

```
PrintWriter out = response.getWriter();
...
out.print(expr);
```

JSP Expression Examples

- Displaying request parameters (request is an implicit object available in a JSP)

Your name is `<%= request.getParameter("name") %>`
and your age is `<%= request.getParameter("age") %>`

- Doing calculations

The value of pi is `<%= Math.PI %>` and the square root of two is `<%= Math.sqrt(2.0) %>` and today's date is `<%= new java.util.Date() %>`.

JSP Scripting Elements: Scriptlet

- For a scriplet `<% statements %>` the Java statements are placed in the translated servlet's **`_jspService`** method body (which is called by either **`doGet`** or **`doPost`**)

```
public void _jspService(HttpServletRequest
   request, HttpServletResponse response)
   throws java.io.IOException, ServletException
{...
   statements
 ...
}
```

JSP Scriplet Examples

- Check a request parameter

```
<% String name = request.getParameter("name");
   if (name == null)
   { %>
      <h3>Please supply a name</h3>
<% }
   else
   { %>
      <h3>Hello <%= name %></h3>
<% } %>
```

In this example, there are 3 scriptlets elements and one expression element

JSP Scripting Elements: Declaration

- For a declaration **<%! declarations %>** the Java statements are placed in the class outside the **_jspService** method.
- Declarations can be Java instance variable declarations or Java methods

```
// declarations would go here
public void _jspService(...)
{
   ...
}
```

JSP Declaration examples

- Declaring instance variables

```
<%! private int count = 0; %>
...
The count is <%= count++ %>.
```

- Declaring methods

```
<%!
private int toInt(String s)
{
   return Integer.parseInt(s);
}
%>
```

Including Files

- Including files at translation time (when JSP is translated to a servlet)

```
<%@ include file="filename" %>
```

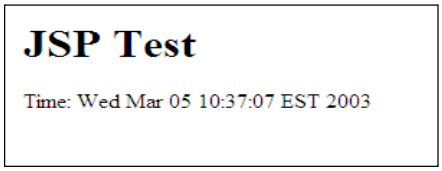
- Including files at request time

```
<jsp:include page="filename" flush = "true" />
```

A simple JSP

```
<html>
<head><title>JSP Test</title></head>
<body>
<h1>JSP Test</h1>
Time: <%= new java.util.Date() %>
</body>
</html>
```

The expression scripting element **<%= ... %>** is equivalent to the scriptlet **<% out.print(...); %>**



The Implicit *out* Object

- In a scriptlet **<% ... %>** you can use the ***out*** object to write to the output stream.
- Example:

```
<%
   out.print("The sum is ");
   out.print("1 + 2 = " + (1+2));
%>
```

The Implicit *request* Object

- In a scriptlet **<% ... %>** you can use the ***request*** object to access GET/POST parameters.

- Example:

```
<html>
<head><title>...</title></head>
<body>
<h1>...</h1>
<p><%= request.getParameter("greeting") %></p>
</body></html>
```

Example: HTML Form Submission Using GET

```
<html>
<head><title>JSP Processing ...</title></head>
<body>
<h1>JSP Processing form with GET</h1>
<form action="processForm.jsp" method="GET">
First name: <input type="text" name="firstName"><br />
Last name: <input type="text" name="lastName">
<p><input type="submit" name="button"
      value="SubmitName"></p>
</form>
</body>
</html>
```

Example: HTML Form Submission Using POST

```
<html>
<head><title>JSP Processing ...</title></head>
<body>
<h1>JSP Processing form with POST</h1>
<form action="processForm.jsp" method="POST">
First name: <input type="text" name="firstName"><br />
Last name: <input type="text" name="lastName">
<p><input type="submit" name="button"
      value="SubmitName"></p>
</form>
</body>
</html>
```

Processing Submitted HTML Forms

ProcessForm.jsp:

```
<html>
<head>
   <title>JSP Form Results</title>
</head>
<body>
<h1>JSP Form Results</h1>
Hello <%= request.getParameter("firstName") %>
<%= request.getParameter("lastName") %>
</body>
</html>
```

Temperature Conversion Example

```
<%@ page import="java.text.DecimalFormat" %>
<html>
<head><title>Fahrenheit ... Conversion</title></head>
<body>
<h1>Fahrenheit to Celsius Conversion</h1>
<% String self = request.getRequestURI();
   if (request.getParameter("convert") == null)
   {
%>
   <form action="<%= self %>" method="POST">
   Fahrenheit temperature:
   <input type="text" name="fahrenheit" />
   <p><input type="submit" name="convert"
         value="Convert to Celsius" /></p>
   </form>
```

Temperature Conversion Example (Contd.)

```
<%
   }
   else
   {
      double fahr = 0.0;
      try
      {
         fahr = Double.parseDouble(
            request.getParameter("fahrenheit"));
      }
      catch (NumberFormatException e)
      {
         // do nothing, accept default value
      }
```

Temperature Conversion Example (Contd.)

```
        double celsius = (fahr - 32.0) * (5.0/9.0);
        DecimalFormat f2 = new DecimalFormat("0.00");
%>
        <%= f2.format(fahr) %>F is
        <%= f2.format(celsius) %>C
        <p><a href="<%= self %>">Another conversion</a>
        </p>
<%
    }
%>
</body>
</html>
```

Java Beans

- Special classes that encapsulate some data
- They have a default constructor
- get and set methods for data fields (properties)
- A bean can be constructed in JSP using
  - `<jsp:useBean id = "..." class = "..." />`
- If the bean already exists this statement does nothing

Setting properties

- To set a property of a bean use

```
<jsp:setProperty name="..."
   property="..." value="..."
/>
```

- To set a property using the value of a request parameter use

```
<jsp:setProperty name="..."
   property="..." param="..."
/>
```

Getting properties

- To get a property of a bean use

```
<jsp:getProperty name="..."
   property="..." />
```

A Greeting Bean

```
package beans;
public class Greeting
{
   private String greeting; // the property

   public Greeting()
   { greeting = "Hello World"; }

   public String getGreeting()
   { return greeting; }

   public void setGreeting(String g)
   {  greeting = (g == null) ? "Hello World" : g;
   }
}
```

Java Beans Naming convention

If the property name is **`greeting`**

- The get method must have the name **`getGreeting`**
- The set method must have the name **`setGreeting`**

Creating a Bean

- Create a bean and use default property

```
<jsp:useBean id="hello" class="beans.Greeting" />
```

- Create a bean and set its property when it is constructed

```
<jsp:useBean id="hello" class="beans.Greeting" >
<jsp:setProperty name="hello" property="greeting"
   value="Hello JSP World" />
</jsp:useBean>
```

- Here **<jsp:setProperty>** is in the body of the **<jsp:useBean>** element.

Creating a Bean (Contd.)

- Create a bean and set its property after it has been constructed

```
<jsp:useBean id="hello" class="beans.Greeting" />
<jsp:setProperty name="hello" property="greeting"
   value="Hello JSP World" />
```

- The **<jsp:setProperty>** tag is now outside the **<jsp:useBean>** tag so it will always set the property, not just when the bean is constructed

Example 1: Using Java Beans in JSP Files

```
<jsp:useBean id="hello" class="beans.Greeting" />
<jsp:setProperty name="hello" property="greeting"
   value="Hello JSP World" />
<html>
<head>
<title>Greeting JSP that uses a Greeting bean</title>
</head>
<body>
<h1>Greeting JSP that uses a Greeting bean</h1>
<p><jsp.getProperty name="hello" property="greeting" />
</p>
</body>
</html>
```

Example 2: Using Java Beans in JSP Files

- Two Beans: One initialized explicitly and the other is initialized using a request parameter

```
<jsp:useBean id="greet1" class="beans.Greeting" />
<jsp:useBean id="greet2" class="beans.Greeting" />
<jsp:setProperty name="greet1" property="greeting"
   value="Hello JSP World" />
<jsp:setProperty name="greet2" property="greeting"
   param="greeting" />
<html><head><title>Greeting JSP using two Greeting
beans</title></head><body>
<h1>Greeting JSP using two Greeting beans</h1>
<p>1st bean: <jsp:getProperty name="greet1"
              property="greeting" /></p>
<p>2nd bean: <jsp:getProperty name="greet2"
              property="greeting" /></p></body></html>
```

Example 3: Using Java Beans in JSP Files

- Three Beans: One initialized explicitly, one is initialized using a request parameter, and one is initialized using getParameter

```
<jsp:useBean id="greet1" class="beans.Greeting" />
<jsp:useBean id="greet2" class="beans.Greeting" />
<jsp:useBean id="greet3" class="beans.Greeting" />
<jsp:setProperty name="greet1" property="greeting"
   value="Hello JSP World" />
<jsp:setProperty name="greet2" property="greeting"
   param="greeting" />

<%-- Following works but param method is better --%>

<jsp:setProperty name="greet3" property="greeting"
  value="<%= request.getParameter(\"greeting\") %>" />
```

Reference

- COS 2206 JSP, Barry G. Adams, Department of Mathematics and Computer Science, Laurentain University.