

Java Lab:

Developing Web-based Applications

Using Java Server Pages(JSP) and
Tomcat

Speaker Contact Information

- Steve Ryder sryder@jsrsys.com
www.jsrsys.com

Advantages of JSP

- . **vs. Active Server Pages (ASP).**
 - Technology from Microsoft, requires MS Server.
 - Dynamic part written in Java, not an MS-specific language
 - It is portable to other operating systems and non-Microsoft Web servers
- . **vs. Server-Side Includes (SSI).**
 - Widely-supported technology for including externally-defined pieces into a static Web page
 - JSP lets you use servlets instead of a separate program to generate the dynamic part
 - SSI is really only intended for simple inclusions, not for "real" programs that use form data or make database connections

- **vs. JavaScript.**

- Generates HTML dynamically on the client
- Only handles situations where the dynamic information is based on the client's environment
- HTTP and form submission data is not available to JavaScript (only exception is cookies)
- Can't access server-side resources like databases, catalogs, or pricing information

- **vs. Static HTML.**

- Cannot contain dynamic information.
- Feasible to augment HTML pages with small amounts of dynamic data using JSP
- The cost of using dynamic data, in HTML, would preclude its use in all but the most valuable instances

- **vs. Pure Servlets.**

- It is more convenient to write (and to modify!) regular HTML than println statements that generate the HTML
- By separating the look from the content you can put different people on different tasks
 - Web page design experts can build the HTML
 - Java Class programmers to insert the dynamic content

Introduction

- Separate dynamic content from static content
- Write static content in HTML
- Enclose dynamic parts in special tags

Example: User clicks on “Super JSP Book” on a web page to order a book.

HTTP call looks like this:

`http://host/OrderConfirmation.jsp?title=Super+JSP+Book`

JSP code looks like this:

`You ordered: <I><%= request.getParameter("title") %> </I>`

Web page looks like this:

`You ordered: Super JSP Book`

- Normally the source code file is given '.jsp' extension
- Used like a normal web page
- Looks like HTML but *is a servlet* behind the scenes

Servlets

- Servlets are Java technology's answer to CGI programming
- Programs that run on a Web server and build Web pages
- Building Web pages on the fly is useful (and commonly done) for a number of reasons:
 - The Web page is based on data submitted by the user.
 - The data changes frequently.
 - The Web page uses information from corporate databases or other such sources.

–

Servlets Advantages

- **It's Efficient** - the Java Virtual Machine stays up, and each request is handled by a lightweight Java thread, not a heavyweight operating system process.
- **It's Convenient** - you are able to use Java rather than learn Perl too
- **It's Powerful** - you can easily do several things that are difficult or impossible with regular CGI
- **It's Portable** - follows a well-standardized API
- **It's Inexpensive** - there are a number of free or very inexpensive Web servers available that are good for "personal" use or low-volume Web sites

Since Java Server Pages (JSP) are Servlets, all the benefits of Servlets pertain to JSP

JSP Template Text

- Static HTML portion of the JSP page
- Follows same HTML syntax rules
- Passed straight through to the client
 - exception is **JSP tag** `<%`
- Can be generated using any tool for creating web pages

JSP Syntax Summary

JSP Element	Syntax	Interpretation
JSP Expression	<code><%= expression %></code>	Expression is evaluated and placed in output. <code>out.println(expression);</code>
JSP Scriptlet	<code><% code %></code>	Java Code is inserted in service method.
JSP Declaration	<code><%! code %></code>	inserted in body of servlet class, outside of service method.
JSP page Directive	<code><%@ page att="val" %></code>	Directions to the servlet engine about general setup.
JSP Comment	<code><%-- comment --%></code>	Comment; ignored when JSP page is translated into servlet.
JSP include Directive	<code><%@ include file="url" %></code>	A file on the local system to be included when the JSP page is translated into a servlet.

JSP Constructs

- Aside from HTML, three main types of constructs
- Scripting elements
 - Let you specify Java code that will become part of the resulting servlet
 - A number of predefined variables such as 'request'
- Directives
 - Let you control the overall structure of the servlet
- Actions
 - Let you control the behavior of the JSP engine

–

JSP Constructs

Scripting Elements

- Java code inserted into the servlet that results from the JSP page.
- Three forms:
 - Expressions `<%= someJavaExpression %>`
 - Scriptlets `<% if (something) ... else ... %>`
 - Declarations `<%! inserted in body of servlet class %>`

JSP Scripting Elements

Expressions

- Used to insert Java value directly into output
- The Java expression is evaluated, converted to a String, and inserted into the web page

Example: Use the `Java.util.Date` method to display the current time.

JSP code looks like this:

```
Current time: <%= new java.util.Date() %>
```

JSP Scripting Elements

Expression Variables

- Predefined variables for simplification
- Four common examples are:
 - `request = HttpServletRequest`
 - `response = HttpServletResponse`
 - `session = HttpSession`
 - `out = PrintWriter`

Example: Use the `HttpServletRequest.getRemoteHost` method to display the user's hostname. JSP code looks like this:

Your hostname: `<%= request.getRemoteHost ()`

`%>` JSP Scripting Elements Scriptlets

- To do something more complex than insert a simple expression, use scriptlets
- Write your own code and have it executed in the JSP

Example: Calculate the tax amount and display it to the user.

JSP code looks like this:

```
<%
```

```
String totTax = subtotal * .075;
```

```
out.println("Total Tax: ", + totTax);
```

`<%>` JSP Scripting Elements

Declarations

- Define methods or fields to insert into the main body of the servlet class
 - Executed outside of the service method
 - Run only at compile or reboot of servlet

Example: Count number of accesses to web page and display. JSP code looks like this:

```
<%! static int accessCount = 0 %>  
Accesses since server reboot:  
<%= ++accessCount %>
```

JSP Constructs

Directives

- Affect the overall structure of the servlet class
- Two main types:
 - Page: set specific attributes
 - Include: files included at translation to servlet

```
<% directive attribute="value" %>
```

```
<%@ directive attribute1="value1"  
                attribute2="value2" %>
```


JSP Directives

page: Some Common Attributes

Attribute	Purpose
Import	specify packages to import <code>import="package.class"</code>
Extends	specifies the superclass of the servlet that will be generated <code>extends="package.class"</code>
Info	defines a string that can be retrieved via the <code>getServletInfo</code> method <code>info="message"</code>

JSP Directives

include

`<%@ include file="relative url" %>`

Example: Include the HTML file that has the Navigation bar.

`<HTML>`

`<BODY>`

`<%@ include file="/navbar.html" %>`

`<!-- HTML and Java Code specific to this page`

`... -->`

`</BODY>`

`</HTML>`

JSP Constructs: Actions

- Use constructs in XML to control the behavior of the servlet engine
 - `jsp:include` - Include a file at the time the page is requested
 - `jsp:forward` - Forward the requester to a new page.
 - `jsp:useBean` - Find or instantiate a `JavaBean`.
 - `jsp:setProperty` - Set the property of a `JavaBean`.
 - `jsp:getProperty` - Insert the property of a `JavaBean` into the output.

JSP Actions: `jsp:include`

- Inserts files into the page being generated

- Inserts the file at the time the page is requested
- Small decrease in efficiency
- Page can not contain general JSP code because it is not inserted at compile time, but rather run time!
- Adds significant flexibility

JSP code looks like this:

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

jsp:include: An Example

<P>

Here is a summary of our four most recent news stories:

<jsp:include page="news/Item1.html"/>

<jsp:include page="news/Item2.html"/>

<jsp:include page="news/Item3.html"/>

<jsp:include page="news/Item4.html"/>

Lab Exercises

- 1 (HelloWorld) and
- 2 (More JSP samples and Forms)

JSP Actions: jsp:forward

- Forward the request to another page
- One attribute, page, holds the URL
- Could be a static value or could be computed at request time

JSP code looks like this:

```
<jsp:forward page="/utils/errorReporter.jsp" />
```

```
<jsp:forward page="<%= someJavaExpression %>" />
```

JSP Actions

jsp:useBean

- Lets you load in a JavaBean to be used in the JSP page
- Very useful capability
- Exploit the reusability of Java classes without sacrificing the convenience of JSP

JSP code looks like this:

```
<jsp:useBean id="name" class="package.class"  
             scope="session"           />
```


**The following combines useBean & jsp:forward:
Authority is set by the Login.jsp script.**

```
<%@page import="com.jsrsys.web.*"%>
<jsp:useBean id="user" class="com.jsrsys.web.JsrUser"
scope="session" />
<% String access = user.getValue("Authority");
    if (access.length()==0 || !access.equals("administrator"))
    {%>
        <jsp:forward page="index.jsp" /><%
    }%>
```

Lab Exercise 3 (SQL):

That's all folks