

Dishing it out with Java Server Pages



Anything ASP can do, JSP can do cross-platform. Over the next months, David Griffiths shows you how to build threaded discussions

There is no denying that one of the Internet Information Server's most impressive features is its use of Active Server Pages. ASPs, for those who don't know, are a clever way of creating Web applications by adding extra tags into an ordinary Web page.

Before ASP, designers accomplished the same thing with languages such as Perl. It uses 'print' statements and embedded HTML to generate Web pages.

Another method used Java Servlets. All right-thinking individuals (that is, Java programmers) used them, but again they had to write out line after line of 'out.print' statements.

What ASPs did was turn the whole process inside out. Rather than writing a program that printed out a Web page, they enabled you to create a Web page that included the output from small programs.

As you would expect, Java Server Pages are very like Active Server Pages. The difference is that JSPs are turned into Java classes by the Web server. As a consequence they are platform independent, secure and multi-threading.

A pain in the ASP

Although ASPs are a clever idea – and a lot of corporate Web sites use them – most public Web providers don't run them. Why?

First, most Web servers don't support ASP software. Second, Web providers may have security concerns about the type of direct access that ASPs provide.

JSPs can run on even the most ancient Web server. They run within a Java virtual machine, so a Web service provider can control security by simply removing any library they fear is a

potential threat, and because it's Java, it doesn't matter if its running on NT, Unix or Linux.

Several ISPs in the States now support JSPs so that ordinary people can turn Web sites into full-blown Web applications. Just like motion pictures, mass-production and Legionnaire's Disease, whatever takes off in North America quickly finds its way over here.

JSP tags

Java Server Pages work by inserting tags into an HTML page. There are three main types of tag: compile time, standard run-time and user-defined run-time.

Compile-time tags provide information on how the JSP should be converted into a Java class (for more details on how this happens, see *How JSP pages work*). The usual format is:

```
<% tagName attributes... %>
```

Standard run-time tags generate predefined chunks of JSP code in your compiled page. They come in single-tag form:

```
<jsp:name attributes... />
```

or double-tag form:

```
<jsp:name attributes...>other
information</jsp:name>
```

The standard tags are so-called because they use the standard 'jsp' library. The third type of tag calls a user-defined library, for example:

```
<yourlib:name attributes... />
```



SEE OUR **SUPERDISC**

PATH: \handson\files\java

The Scriptlet tag

Scriptlets embody the JSP design concept, which is that you should be able to design a page using a GUI or text editor, and then insert pieces of code in afterwards. Here's an example:

```
<%
SimpleDateFormat f = new
SimpleDateFormat (
"E, dd/MM/yyyy 'at' hh:mm:ss a
zzz"
);
Date d = new Date();
out.print(f.format(d));
%>
```

This prints out the current date and time using java.util.SimpleDateFormat class on the server. Unlike JavaScripts, scriptlets don't have to be complete pieces of Java code, but can be broken into several parts, with normal HTML code sitting in between, for example:

```
<%
SimpleDateFormat f = new
SimpleDateFormat ("a");
Date d = new Date();
String ampm = f.format(d);
if (ampm.equals("AM")) {
%>
<h1>Good morning!</h1>
<% } else { %>
<h1>Good afternoon!</h1>
<% } %>
```

In this example, the opening and closing brackets of the [if-else] are in separate scriptlets. The JSP can access all of the usual methods and objects, including the request and response information:

```
<P>Parameter fred is &quot;
<% out.print(request.getParameter
("fred")); %>
&quot;;
```

Here the predefined request object enables us to look at the value of a parameter called 'fred' that has been posted to the page, or passed to it

ON THE WEB

Getting JSP software

The Web is currently awash with JSP software. Thankfully most of it's free

The reference version of Java Server Pages is available from Sun at <http://java.sun.com/products/jsp>. This is now known as the Apache Tomcat server (<http://jakarta.apache.org/>). It comes supplied with a small Web server and because it is written entirely in Java it's only about 1Mb in size.

If you want an open source JSP engine then there's a Gnu one at www.klomp.org/gnujsp/ and another called PolyJSP at www.plenix.org/polyjsp which claims it can produce JSPs written in non-Java languages.

If you have some Web server software and would like to add JSPs to it, we'd recommend

JRUN from Allaire: www.allaire.com. It's excellent, although as it includes its own Java Runtime Environment it's 18Mb.

Finally for a list of public Web providers who currently support servlets and JSPs see www.adrenalinegroup.com/jwsisp.html.

in the URL. Conversely it is also possible to access the response object. Among other things, this enables us to re-define the MIME-type of the Web page. The MIME-type is the piece of information that tells the Web browser how it should be displayed. By default, JSPs set the MIME-type to 'text/html' because most of them will be ordinary HTML Web pages. However, the following will change the MIME-type to VRML:

```
#VRML V2.0 utf8
<% response.setContentType
("model/vrml"); %>
Shape {geometry Box {}}
```

If you change the MIME-type of a page, remember to call it using an extra dummy parameter, for example:

```
http://host_machine/vrml_page
.jsp?dummy=.wrl
```

The dummy parameter in the URL ensures that it ends with the characters '.wrl' which will avoid a problem in Internet Explorer that will prevent it loading the correct plug-in.

The scriptlet tag alone will enable you to create complex interactive pages, but it is most commonly used in conjunction with the <jsp:useBean...> tag.

The useBean tag

Java Server Pages often need to send page parameters to a piece of code on a server. For this reason, JSPs can be bound to JavaBeans classes using the <jsp:useBean...> tag. Consider the following example:

```
<jsp:useBean id="msg" class="com.
spigots.jsp.MessageStore"
scope="session"/>
<jsp:setProperty name="msg"
property="*" />
```

The <jsp:useBean...> tag will cause a

MessageStore object to be created and given the name 'msg'.

The <jsp:setProperty...> tag will then sift its way through all of the variables currently defined and will send them to the properties of the bean that have the same name.

Imagine our 'MessageStore' class has a method called 'setMessage(String...)'. If our page has been called with the following URL:

```
http://127.0.0.1/bean1.jsp?message=
Hello+World
```

the page will first of all create a new instance of the MessageStore object called msg and, because the 'message' parameter from the URL matches the 'setMessage(...)' method of object, the compiled page will effectively call:

```
msg.setMessage("Hello World");
```

If we now enter the following lines into the body of the page:

```
Message is: <% out.print(msg.get
Message()); %>
```

we will display the contents of the 'message' variable that was passed to it.

There is a <jsp:getProperty...> tag, which simplifies the use of property values within the page. In our example, we could have used:

```
<jsp:getProperty name=msg property=
message />
```

In the compiled version of the page, the two would be identical.

Our MessageStore bean just stores and returns a String value, but it could have been a commercial program to return the results of a site search, or add an entry to guestbook.

The Page tag

Our page will be converted dynamically

into a Java source file, and the Page tag enables us to modify the way the conversion process works. For instance, just as we would normally import a libraries using Java's 'import'. We can do the same within a JSP using:

```
<%@ page import="java.net.*,
java.util.zip.*" %>
```

Remember how we changed the MIME-header using a scriptlet above? It's easier to use:

```
<%@ page contentType="model
/vrml" %>
```

The page tag can also specify whether multiple copies of the page can be run at once, whether it should output a special page if an error occurs and how much of the output should be buffered before being sent back to the browser.

Including Library code

Library code can also be inserted into a page in one of two ways. The first looks up code from a text file when a page is compiled:

```
<%@ include file="static.txt" %>
```

This means that if the contents of the 'static.txt' file are later changed, the updated version will not appear in our JSP page. The second way is:

```
<jsp:include page="dynamic.txt"
flush="true" />
```

Here, the contents of 'dynamic.txt' are looked up each time the page is run. **PCP**

NEXT MONTH

Put your JSP knowledge to work creating threaded discussions on your site.

JSP ARCHITECTURE

How JSP pages work

Behind the scenes the server brokers each page request

1. A browser makes a request to a JSP enabled Web site. If the page you are requesting has a '.jsp' extension, the Web server will pass the request over to the JSP/Servlet engine.

2. The JSP engine checks to see if the page needs to be compiled. It will do this by comparing the last compilation time (if there was one) with the last time the page was updated. Because the page will only be compiled if it has changed, JSP pages generally run slightly slower on the first access, but very quickly from then on. If the page needs to be compiled, the JSP engine will pass it over to the JSP compiler.

3. The JSP compiler. The page will be split up into two parts: the sections of HTML text and the in-lined Java code. The HTML text will be stored as an array of text strings in a data file. The

in-lined code will be stitched together to form a new dynamically created Java Servlet. The compiled page can then use the sections of HTML text to re-create the main layout of the page as well as insert the dynamic elements from embedded Java code.

4. Because the compiled page becomes part of the class structure in the Web server's Java machine, it can access any other class that has been made available to it. In the same way that some Web service providers remove particular Perl libraries, you might find that some of the standard Java classes

(such as those in the java.net.* package) will not be available for security reasons.

5. The compiled class uses a JspWriter class to output HTML text back to the browser. A JSP page uses

a cookie called 'SESSIONID' to create a virtual session between a browser and the Web server. Data persists between individual requests, making it very easy to create Web applications such as shopping carts.

