10.4 Misconceptions About JSP

In this section, we address some of the most common misunderstandings about JSP.

Forgetting JSP Is Server-Side Technology

The book's Web site lists the lead author's email address: hall@coreservlets.com. Furthermore, Marty teaches JSP and servlet training courses for various companies and at public venues. Consequently, he gets a lot of email with servlet and JSP questions. Here are some typical questions he has received (most of them repeatedly).

- Our server is running JDK 1.4. So, how do I put a Swing component in a JSP page?
- How do I put an image into a JSP page? I do not know the proper Java I/O commands to read image files.
- Since Tomcat does not support JavaScript, how do I make images that are highlighted when the user moves the mouse over them?
- Our clients use older browsers that do not understand JSP. What should we do?
- When our clients use "View Source" in a browser, how can I prevent them from seeing the JSP tags?

All of these questions are based upon the assumption that browsers know something about the server-side process. But they do not. Thus:

- For putting applets with Swing components into Web pages, what matters is the browser's Java version—the server's version is irrelevant. If the browser supports the Java 2 platform, you use the normal `APPLET` (or Java plug-in) tag and would do so even if you were using non-Java technology on the server.
- You do not need Java I/O to read image files; you just put the image in the directory for Web resources (i.e., two levels up from `WEB-INF/classes`) and output a normal `IMG` tag.
- You create images that change under the mouse by using client-side JavaScript, referenced with the `SCRIPT` tag; this does not change just because the server is using JSP.
- Browsers do not "support" JSP at all—they merely see the output of the JSP page. So, make sure your JSP outputs HTML compatible with the browser, just as you would do with static HTML pages.
- And, of course you need not do anything to prevent clients from seeing JSP tags; those tags are processed on the server and are not part of the output that is sent to the client.

Confusing Translation Time with Request Time

A JSP page is converted into a servlet. The servlet is compiled, loaded into the server's memory, initialized, and executed. But which step happens when? To answer that question, remember two points:

- The JSP page is translated into a servlet and compiled only the first time it is accessed after having been modified.
- Loading into memory, initialization, and execution follow the normal rules for servlets.

Table 10.1 gives some common scenarios and tells whether or not each step occurs in that scenario. The most frequently misunderstood entries are highlighted. When referring to the table, note that servlets resulting from JSP pages use the `_jspService` method (called for both `GET` and `POST` requests), not `doGet` or `doPost`. Also, for initialization, they use the `jspInit` method, not the `init` method.

Table 10.1. JSP Operations in Various Scenarios

	JSP page translated into servlet	**Servlet compiled**	**Servlet loaded into server's memory**	**`jspInit` called**	**`_jspService` called**
			Page first written		
Request 1	Yes	Yes	Yes	Yes	Yes
Request 2	**No**	**No**	**No**	**No**	Yes
			Server restarted		
Request 3	**No**	**No**	Yes	Yes	Yes
Request 4	No	No	No	No	Yes
			Page modified		
Request 5	Yes	Yes	Yes	Yes	Yes
Request 6	No	No	No	No	Yes

Thinking JSP Alone Is Sufficient

There is a small community of developers that are so enamored with JSP that they use it for practically everything. Most of these developers *never* use servlets; many never even use auxiliary helper classes—they just build large complex JSP pages for each and every task.

This is a mistake. JSP is an excellent tool. But the fundamental problem it addresses is *presentation*: the difficulty of creating and maintaining HTML to represent the result of a request. JSP is a good choice for pages with relatively fixed formats and lots of static text. JSP, by itself, is less good for applications that have a variable structure, is poor for applications that have mostly dynamic data, and is totally unsuitable for applications that output binary data or manipulate HTTP without generating explicit output (as with the search engine servlet of Section 6.4). Still other applications are best solved with neither servlets alone nor JSP alone, but with a combination of the two (see Chapter 15).

JSP is a powerful and widely applicable tool. Nevertheless, other tools are sometimes better. Choose the right tool for the job.

Thinking Servlets Alone Are Sufficient

At the other end of the spectrum from the JSP-only camp is the servlets-only camp. Adherents of this camp state, quite rightly, that JSP pages are really just dressed up servlets, so JSP pages cannot accomplish anything that could not also be done with servlets. From this, they conclude that you should stick with servlets, where you have access to the full underlying power, have complete control, and can see exactly what is happening. Hmm, have you heard this argument before? It sounds a lot like the position of the "don't be a wimp; write all your code in assembly language" crowd.

Yes, you could use servlets for any task for which JSP is used. But it is not always equally convenient to do so. For tasks that involve a lot of static HTML content, use of JSP technology (or a combination of JSP and servlets) simplifies the creation and maintenance of the HTML, permits you to use industry standard Web site creation tools, and lets you "divide and conquer" by splitting your effort between the Java developers and the Web developers.

Servlets are powerful and widely applicable tools. Nevertheless, other tools are sometimes better. Choose the right tool for the job.

[Team LiB] PREVIOUS NEXT