11.7 Writing Scriptlets

If you want to do something more complex than output the value of a simple expression, JSP scriptlets let you insert arbitrary code into the servlet's `_jspService` method (which is called by `service`). Scriptlets have the following form:

```
<% Java Code %>
```

Scriptlets have access to the same automatically defined variables as do expressions (`request`, `response`, `session`, `out`, etc.). So, for example, if you want to explicitly send output to the resultant page, you could use the `out` variable, as in the following example.

```
<%
String queryData = request.getQueryString();
out.println("Attached GET data: " + queryData);
%>
```

In this particular instance, you could have accomplished the same effect more easily by using a combination of a scriptlet and a JSP expression, as below.

```
<% String queryData = request.getQueryString(); %>
Attached GET data: <%= queryData %>
```

Or, you could have used a single JSP expression, as here.

```
Attached GET data: <%= request.getQueryString() %>
```

In general, however, scriptlets can perform a number of tasks that cannot be accomplished with expressions alone. These tasks include setting response headers and status codes, invoking side effects such as writing to the server log or updating a database, or executing code that contains loops, conditionals, or other complex constructs. For instance, the following snippet specifies that the current page is sent to the client as Microsoft Word, not as HTML (which is the default). Since Microsoft Word can import HTML documents, this technique is actually quite useful in real applications.

```
<% response.setContentType("application/msword"); %>
```

It is important to note that you need not set response headers or status codes at the *very* top of a JSP page, even though this capability appears to violate the rule that this type of response data needs to be specified before any document content is sent to the client. It is legal to set headers and status codes after a small amount of document content because servlets that result from JSP pages use a special variety of `Writer` (of type `JspWriter`) that partially buffers the document. This buffering behavior can be changed, however; see Chapter 12 (Controlling the Structure of Generated Servlets: The JSP page Directive) for a discussion of the `buffer` and `autoflush` attributes of the `page` directive.

JSP/Servlet Correspondence

It is easy to understand how JSP scriptlets correspond to servlet code: the scriptlet code is just directly inserted into the `_jspService` method: no strings, no `print` statements, no changes whatsoever. For instance, Listing 11.6 shows a small JSP sample that includes some static HTML, a JSP expression, and a JSP scriptlet. Listing 11.7 shows a `_jspService` method that might result. Note that the call to `bar` (the JSP expression) is not followed by a semicolon, but the call to `baz` (the JSP scriptlet) is. Remember that JSP expressions contain Java *values* (which do not end in semicolons), whereas most JSP scriptlets contain Java *statements* (which are

terminated by semicolons). To make it even easier to remember when to use a semicolon, simply remember that expressions get placed inside `print` or `write` statements, and `out.print(blah;);` is clearly illegal.

Again, different vendors will produce this code in slightly different ways, and we oversimplified the `out` variable (which is a `JspWriter`, not the slightly simpler `PrintWriter` that results from a call to `getWriter`). So, don't expect the code your server generates to look *exactly* like this.

Listing 11.6 Sample JSP Expression/Scriptlet

```
<H2>foo</H2>
<%= bar() %>
<% baz(); %>
```

Listing 11.7 Representative Resulting Servlet Code: Expression/Scriptlet

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
    throws ServletException, IOException {
  response.setContentType("text/html");
  HttpSession session = request.getSession();
  JspWriter out = response.getWriter();
  out.println("<H2>foo</H2>");
  out.println(bar());
  baz();
  ...
}
```

XML Syntax for Scriptlets

The XML equivalent of `<% Java Code %>` is

```
<jsp:scriptlet>Java Code</jsp:scriptlet>
```

In JSP 1.2 and later, servers are required to support this syntax as long as authors don't mix the XML version (`<jsp:scriptlet> ... </jsp:scriptlet>`) and the ASP-like version (`<% ... %>`) in the same page; if you use the XML version you must use XML syntax consistently for the entire page. Remember that XML elements are case sensitive; be sure to use `jsp:scriptlet` in lower case.