11.4 Using JSP Expressions

A JSP expression is used to insert values directly into the output. It has the following form:

```
<%= Java Expression %>
```

The expression is evaluated, converted to a string, and inserted in the page. This evaluation is performed at runtime (when the page is requested) and thus has full access to information about the request. For example, the following shows the date/time that the page was requested.

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

Predefined Variables

To simplify these expressions, you can use a number of predefined variables (or "implicit objects"). There is nothing magic about these variables; the system simply tells you what names it will use for the local variables in `_jspService` (the method that replaces `doGet` in servlets that result from JSP pages). These implicit objects are discussed in more detail in Section 11.12, but for the purpose of expressions, the most important ones are these:

- **`request`**, the `HttpServletRequest`.
- **`response`**, the `HttpServletResponse`.
- **`session`**, the `HttpSession` associated with the request (unless disabled with the `session` attribute of the `page` directive—see Section 12.4).
- **`out`**, the `Writer` (a buffered version of type `JspWriter`) used to send output to the client.
- **`application`**, the `ServletContext`. This is a data structure shared by all servlets and JSP pages in the Web application and is good for storing shared data. We discuss it further in the chapters on beans (Chapter 14) and MVC (Chapter 15).

Here is an example:

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

JSP/Servlet Correspondence

Now, we just stated that a JSP expression is evaluated and inserted into the page output. Although this is true, it is sometimes helpful to understand what is going on behind the scenes.

It is actually quite simple: JSP expressions basically become `print` (or `write`) statements in the servlet that results from the JSP page. Whereas regular HTML becomes `print` statements with double quotes around the text, JSP expressions become `print` statements with no double quotes. Instead of being placed in the `doGet` method, these `print` statements are placed in a new method called `_jspService` that is called by `service` for both `GET` and `POST` requests. For instance, Listing 11.1 shows a small JSP sample that includes some static HTML and a JSP expression. Listing 11.2 shows a `_jspService` method that might result. Of course, different vendors will produce code in slightly different ways, and optimizations such as reading the HTML from a static byte array are quite common.

Also, we oversimplified the definition of the `out` variable; `out` in a JSP page is a `JspWriter`, so you have to modify the slightly simpler `PrintWriter` that directly results from a call to `getWriter`. So, don't expect the code your server generates to look *exactly* like this.

Listing 11.1 Sample JSP Expression: Random Number

```
<H1>A Random Number</H1>
<%= Math.random() %>
```

Listing 11.2 Representative Resulting Servlet Code: Random Number

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
    throws ServletException, IOException {
  response.setContentType("text/html");
  HttpSession session = request.getSession();
  JspWriter out = response.getWriter();
  out.println("<H1>A Random Number</H1>");
  out.println(Math.random());
  ...
}
```

If you want to see the exact code that your server generates, you'll have to dig around a bit to find it. In fact, some servers delete the source code files once they are successfully compiled. But here is a summary of the locations used by three common, free development servers.

Tomcat Autogenerated Servlet Source Code

install_dir`/work/Standalone/localhost/_`

(The final directory is an underscore. More generally, in *install_dir*`/work/Standalone/localhost/`*webAppName*.

The location varies slightly among various Tomcat versions.)

JRun Autogenerated Servlet Source Code

install_dir`/servers/default/default-ear/default-war/WEB-INF/jsp`

(More generally, in the `WEB-INF/jsp` directory of the Web application to which the JSP page belongs. However, note that JRun does not save the `.java` files unless you change the `keepGenerated` element from `false` to `true` in *install_dir*`/servers/default/SERVER-INF/default-web.xml`.)

Resin Autogenerated Servlet Source Code

install_dir`/doc/WEB-INF/work`

(More generally, in the `WEB-INF/work` directory of the Web application to which the JSP page belongs.)

XML Syntax for Expressions

XML authors can use the following alternative syntax for JSP expressions:

`<jsp:expression>`*Java Expression*`</jsp:expression>`

In JSP 1.2 and later, servers are required to support this syntax as long as authors don't mix the XML version and the standard JSP version (`<%= ... %>`) in the same page. This means that, to use the XML version, you must use XML syntax in the *entire* page. In JSP 1.2 (but not 2.0), this requirement means that you have to enclose the entire page in a `jsp:root` element. As a result, most developers stick with the classic syntax except when they are either generating XML documents (e.g., xhtml or SOAP) or when the JSP page is itself the output of some XML process (e.g., XSLT).

Note that XML elements, unlike HTML ones, are case sensitive. So, be sure to use `jsp:expression` in lower case.

[Team LiB]

PREVIOUS NEXT