13.1 Including Pages at Request Time: The jsp:include Action

Suppose you have a series of pages, all of which have the same navigation bar, contact information, or footer. What can you do? Well, one common "solution" is to cut and paste the same HTML snippets into all the pages. This is a bad idea because when you change the common piece, you have to change every page that uses it. Another common solution is to use some sort of server-side include mechanism whereby the common piece gets inserted as the page is requested. This general approach is a good one, but the typical mechanisms are server specific. Enter `jsp:include`, a portable mechanism that lets you insert any of the following into the JSP output:

- The content of an HTML page.
- The content of a plain text document.
- The output of JSP page.
- The output of a servlet.

The `jsp:include` action includes the output of a secondary page at the time the main page is requested. Although the *output* of the included pages cannot contain JSP, the pages can be the result of resources that use servlets or JSP to *create* the output. That is, the URL that refers to the included resource is interpreted in the normal manner by the server and thus can be a servlet or JSP page. The server runs the included page in the usual way and places the output into the main page. This is precisely the behavior of the `include` method of the `RequestDispatcher` class (see Chapter 15, "Integrating Servlets and JSP: The Model View Controller (MVC) Architecture"), which is what servlets use if they want to do this type of file inclusion.

The page Attribute: Specifying the Included Page

You designate the included page with the `page` attribute, as shown below. This attribute is required; it should be a relative URL referencing the resource whose output should be included.

```
<jsp:include page="relative-path-to-resource" />
```

Relative URLs that do not start with a slash are interpreted relative to the location of the main page. Relative URLs that start with a slash are interpreted relative to the base Web application directory, *not* relative to the server root. For example, consider a JSP page in the `headlines` Web application that is accessed by the URL `http://host/headlines/sports/table-tennis.jsp`. The `table-tennis.jsp` file is in the `sports` subdirectory of whatever directory is used by the `headlines` Web application. Now, consider the following two include statements.

```
<jsp:include page="bios/cheng-yinghua.jsp" />
<jsp:include page="/templates/footer.jsp" />
```

In the first case, the system would look for `cheng-yinghua.jsp` in the `bios` subdirectory of `sports` (i.e., in the `sports/bios` sub-subdirectory of the main directory of the `headlines` application). In the second case, the system would look for `footer.jsp` in the `templates` subdirectory of the `headlines` application, *not* in the `templates` subdirectory of the server root. The `jsp:include` action *never* causes the system to look at files outside of the current Web application. If you have trouble remembering how the system interprets URLs that begin with

slashes, remember this rule: they are interpreted relative to the current Web application whenever the server handles them; they are interpreted relative to the server root only when the client (browser) handles them. For example, the URL in

```
<jsp:include page="/path/file" />
```

is interpreted within the context of the current Web application because the server resolves the URL; the browser never sees it. But, the URL in

```
<IMG SRC="/path/file" ...>
```

is interpreted relative to the server's base directory because the browser resolves the URL; the browser knows nothing about Web applications. For information on Web applications, see Section 2.11.

Core Note

URLs that start with slashes are interpreted differently by the server than by the browser. The server always interprets them relative to the current Web application. The browser always interprets them relative to the server root.

Finally, note that you are permitted to place your pages in the `WEB-INF` directory. Although the client is prohibited from directly accessing files in this directory, it is the server, not the client, that accesses files referenced by the `page` attribute of `jsp:include`. In fact, placing the included pages in `WEB-INF` is a recommended practice; doing so will prevent them from being accidentally accessed by the client (which would be bad, since they are usually incomplete HTML documents).

Core Approach

To prevent the included files from being accessed separately, place them in `WEB-INF` or a subdirectory thereof.

XML Syntax and jsp:include

The `jsp:include` action is one of the first JSP constructs we have seen that has only XML syntax, with no equivalent "classic" syntax. If you are unfamiliar with XML, note three things:

- **XML element names can contain colons.** So, do not be thrown off by the fact that the element name is `jsp:include`. In fact, the XML-compatible version of all standard JSP elements starts with the `jsp` prefix (or namespace).
- **XML tags are case sensitive.** In standard HTML, it does not matter if you say `BODY`, `body`, or `Body`. In XML, it matters. So, be sure to use `jsp:include` in all lower case.
- **XML tags must be explicitly closed.** In HTML, there are container elements such as `H1` that have both start and end tags (`<H1> ... </H1>`) as well as standalone elements such as `IMG` or `HR` that have no end tags (`<HR>`). In addition, the HTML specification defines the end tags of some container elements (e.g., `TR`, `P`) to be optional. In XML, all elements are container elements, and end tags are never optional. However, as a convenience, you can

replace bodyless snippets such as `<blah></blah>` with `<blah/>`. So when using `jsp:include`, be sure to include that trailing slash.

The flush Attribute

In addition to the required `page` attribute, `jsp:include` has a second attribute: `flush`, as shown below. This attribute is optional; it specifies whether the output stream of the main page should flushed before the inclusion of the page (the default is `false`). Note, however, that in JSP 1.1, `flush` was a required attribute and the only legal value was `true`.

```
<jsp:include page="relative-path-to-resource" flush="true" />
```

A News Headline Page

As an example of a typical use of `jsp:include`, consider the simple news summary page shown in Listing 13.1. Page developers can change the news items in the files `Item1.html` through `Item3.html` (Listings 13.2 through 13.4) without having to update the main news page. Figure 13-1 shows the result.

Figure 13-1. Including files at request time lets you update the individual files without changing the main page.

Notice that the included pieces are not complete Web pages. The included pages can be HTML files, plain text files, JSP pages, or servlets (but with JSP pages and servlets, only the output of the page is included, not the actual code). In all cases, however, the client sees only the *composite* result. So, if both the main page and the included pieces contain tags such as `DOCTYPE`, `BODY`, etc., the result will be illegal HTML because these tags will appear twice in the result that the client sees. With servlets and JSP, it is always a good habit to view the HTML source and submit the URL to an HTML validator (see Section 3.5, "Simple HTML-Building Utilities"). When `jsp:include` is used, this advice is even more important because beginners often erroneously design both the main page and the included page as complete HTML documents.

Core Approach

Do not use complete HTML documents for your included pages. Include only the HTML tags appropriate to the place where the included files will be inserted.

Listing 13.1 WhatsNew.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>What's New at JspNews.com</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
      What's New at JspNews.com</TABLE>


<P>
Here is a summary of our three most recent news stories:
<OL>
  <LI><jsp:include page="/WEB-INF/Item1.html" />
  <LI><jsp:include page="/WEB-INF/Item2.html" />
  <LI><jsp:include page="/WEB-INF/Item3.html" />
</OL>
</BODY></HTML>
```

Listing 13.2 /WEB-INF/Item1.html

```
<B>Bill Gates acts humble.</B> In a startling and unexpected
development, Microsoft big wig Bill Gates put on an open act of
humility yesterday.
<A HREF="http://www.microsoft.com/Never.html">More details...</A>
```

Listing 13.3 /WEB-INF/Item2.html

```
<B>Scott McNealy acts serious.</B> In an unexpected twist,
wisecracking Sun head Scott McNealy was sober and subdued at
yesterday's meeting.
<A HREF="http://www.sun.com/Imposter.html">More details...</A>
```

Listing 13.4 /WEB-INF/Item3.html

```
<B>Larry Ellison acts conciliatory.</B> Catching his competitors
off guard yesterday, Oracle prez Larry Ellison referred to his
rivals in friendly and respectful terms.
<A HREF="http://www.oracle.com/Mistake.html">More details...</A>
```

The jsp:param Element: Augmenting Request Parameters

The included page uses the same request object as the originally requested page. As a result, the included page normally sees the same request parameters as the main page. If, however, you want to add to or replace those parameters, you can use the `jsp:param` element (which has `name` and `value` attributes) to do so. For example, consider the following snippet.

```
<jsp:include page="/fragments/StandardHeading.jsp">
  <jsp:param name="bgColor" value="YELLOW" />
</jsp:include>
```

Now, suppose that the main page is invoked by means of `http://host/path/MainPage.jsp?`

`fgColor=RED`. In such a case, the following list summarizes the results of various `getParameter` calls.

- **In main page (`MainPage.jsp`).** (Regardless of whether the `getParameter` calls are before or after the file inclusion.)
    - `request.getParameter("fgColor")` returns `"RED"`.
    - `request.getParameter("bgColor")` returns `null`.
- **In included page (`StandardHeading.jsp`).**
    - `request.getParameter("fgColor")` returns `"RED"`.
    - `request.getParameter("bgColor")` returns `"YELLOW"`.

If the main page receives a request parameter that is also specified with the `jsp:param` element, the value from `jsp:param` takes precedence only in the included page.

[Team LiB]

PREVIOUS NEXT