13.2 Including Files at Page Translation Time: The include Directive

You use the `include` directive to include a file in the main JSP document at the time the document is translated into a servlet (which is typically the first time it is accessed). The syntax is as follows:

```
<%@ include file="Relative URL" %>
```

Think of the `include` directive as a preprocessor: the included file is inserted character for character into the main page, then the resultant page is treated as a single JSP page. So, the fundamental difference between `jsp:include` and the `include` directive is the time at which they are invoked: `jsp:include` is invoked at request time, whereas the `include` directive is invoked at page translation time. However, there are more implications of this difference than you might first think. We summarize them in Table 13.1.

Table 13.1. Differences Between `jsp:include` and the `include` Directive

	`jsp:include` Action	`include` Directive
What does basic syntax look like?	`<jsp:include page="..." />`	`<%@ include file="..." %>`
When does inclusion occur?	Request time	Page translation time
What is included?	Output of page	Actual content of file
How many servlets result?	Two (main page and included page each become a separate servlet)	One (included file is inserted into main page, then that page is translated into a servlet)
Can included page set response headers that affect the main page?	No	Yes
Can included page define fields or methods that main page uses?	No	Yes
Does main page need to be updated when included page changes?	No	Yes
What is the equivalent servlet code?	`include` method of `RequestDispatcher`	None
Where is it discussed?	Section 13.1	Section 13.2 (this section)

There are many ramifications of the fact that the included file is inserted at page translation time with the `include` directive (`<%@ include ... %>`), not at request time as with `jsp:include`. However, there are two really important implications: maintenance and power. We discuss these two items in the following two subsections.

Maintenance Problems with the include Directive

The first ramification of the inclusion occurring at page translation time is that it is much more

difficult to maintain pages that use the `include` directive than is the case with `jsp:include`. With the `include` directive (`<%@ include ... %>`), if the included file changes, all the JSP files that use it may need to be updated. Servers are required to detect when a JSP page changes and to translate it into a new servlet before handling the next request. Unfortunately, however, they are not required to detect when the included file changes, only when the main page changes. Servers are *allowed* to support a mechanism for detecting that an included file has changed (and then recompiling the servlet), but they are not *required* to do so. In practice, few do. So, with most servers, whenever an included file changes, *you* have to update the modification dates of each JSP page that uses the file.

This is a significant inconvenience; it results in such serious maintenance problems that the `include` directive should be used only in situations in which `jsp:include` would not suffice. Some developers have argued that using the `include` directive results in code that executes faster than it would with the `jsp:include` action. Although this may be true in principle, the performance difference is so small that it is difficult to measure, and the maintenance advantages of `jsp:include` are so great that it is virtually always preferred when both options are available. In fact, some developers find the maintenance burden of the `include` directive so onerous that they avoid it altogether. Perhaps this is an overreaction, but, at the very least, reserve the `include` directive for situations for which you really need the extra power it affords.

Core Approach

For file inclusion, use `jsp:include` whenever possible. Reserve the `include` directive (`<%@ include ... %>`) for cases in which the included file defines fields or methods that the main page uses or when the included file sets response headers of the main page.

Additional Power from the include Directive

If the `include` directive results in hard-to-maintain code, why would anyone want to use it? Well, that brings up the second difference between `jsp:include` and the `include` directive. The `include` directive is more powerful. With the `include` directive, the included file is permitted to contain JSP code such as response header settings and field definitions *that affect the main page*. For example, suppose `snippet.jsp` contained the following line of code:

```
<%! int accessCount = 0; %>
```

In such a case, you could do the following in the main page:

```
<%@ include file="snippet.jsp" %> <%-- Defines accessCount --%>
<%= accessCount++ %>              <%-- Uses accessCount --%>
```

With `jsp:include`, of course, this would be impossible because of the undefined `accessCount` variable; the main page would not translate successfully into a servlet. Besides, even if it could be translated without error, there would be no point; `jsp:include` includes the output of the auxiliary page, and `snippet.jsp` has no output.

Updating the Main Page

With most servers, if you use the `include` directive and change the included file, you also have to update the modification date of the main page. Some operating systems have commands that update the modification date without your actually editing the file (e.g., the Unix `touch` command), but a simple portable alternative is to include a JSP comment in the top-level page. Update the comment whenever the included file changes. For example, you might put the

modification date of the included file in the comment, as below.

```
<%-- Navbar.jsp modified 9/1/03 --%>
<%@ include file="Navbar.jsp" %>
```

Core Warning

If you change an included JSP file, you may have to update the modification dates of all JSP files that use it.

XML Syntax for the include Directive

The XML-compatible equivalent of

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

is

```
<jsp:directive.include file="..." />
```

When this form is used, both the main page and the included file must use XML-compatible syntax throughout.

Example: Reusing Footers

As an example of a situation in which you would use the `include` directive instead of `jsp:include`, suppose that you have a JSP page that generates an HTML snippet containing a small footer that includes access counts and information about the most recent accesses to the current page. Listing 13.5 shows just such a page.

Now suppose you have several pages that want to have footers of that type. You could put the footer in `WEB-INF` (where it is protected from direct client access) and then have the pages that want to use it do so with the following.

```
<%@ include file="/WEB-INF/ContactSection.jsp" %>
```

Listing 13.6 shows a page that uses this approach; Figure 13-2 shows the result. "Hold on!" you say, "Yes, `ContactSection.jsp` defines some instance variables (fields). And, if the main page *used* those instance variables, I would agree that you would have to use the `include` directive. But, in this particular case, the main page does not use the instance variables, so `jsp:include` should be used instead. Right?" Wrong. If you used `jsp:include` here, then all the pages that used the footer would see the same access count. You want each page that uses the footer to maintain a different access count. You do not want `ContactSection.jsp` to be its own servlet, you want `ContactSection.jsp` to provide code that will be part of each *separate* servlet that results from a JSP page that uses `ContactSection.jsp`. You need the `include` directive.

Listing 13.5 ContactSection.jsp

```
<%@ page import="java.util.Date" %>
<%-- The following become fields in each servlet that
     results from a JSP page that includes this file. --%>
<%!
private int accessCount = 0;
```

```
private Date accessDate = new Date();
private String accessHost = "<I>No previous access</I>";
%>
<P>
<HR>
This page &copy; 2003
<A HREF="http//www.my-company.com/">my-company.com</A>.
This page has been accessed <%= ++accessCount %>
times since server reboot. It was most recently accessed from
<%= accessHost %> at <%= accessDate %>.
<% accessHost = request.getRemoteHost(); %>
<% accessDate = new Date(); %>
```

Listing 13.6 SomeRandomPage.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Some Random Page</TITLE>
<META NAME="author" CONTENT="J. Random Hacker">
<META NAME="keywords"
      CONTENT="foo,bar,baz,quux">


<META NAME="description"
      CONTENT="Some random Web page.">
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
      Some Random Page</TABLE>
<P>
Information about our products and services.
<P>
Blah, blah, blah.
<P>
Yadda, yadda, yadda.
<%@ include file="/WEB-INF/ContactSection.jsp" %>
</BODY></HTML>
```

Figure 13-2. Result of `SomeRandomPage.jsp`. It uses the `include` directive so that it maintains access counts and most-recent-hosts entries separately from any other pages that use `ContactSection.jsp`.

[Team LiB]

PREVIOUS NEXT