The journal of Apple technology.

Volume Number: 15 (1999)
Issue Number: 8
Column Tag: Web Development

Server-Side Includes

by Rich Morin and Vicki Brown

Generating A Dynamic Appearance in Static HTML

A Desire for Consistency

We realize that static web pages are passé, but we still need to generate them from time to time. We also like to have a certain consistency (and controlled variability) to our web pages. So, we use server-parsed HTML documents, more commonly known as server-side includes, to manage page headers and footers.

Server-side includes (SSIs) provide a convenient way to embed variables (such as the current time, date, URL, or size of the current page) into any HTML document "on the fly". SSIs can also be used to insert the contents of "boilerplate" files into the current document. This makes them an excellent choice for storing common information, such as Copyright notices, contact information, and navigation bars.

SSI support is provided by several of the common Web servers, including Apache http://www.apache.org for Unixish systems, WebTen http://www.tenon.com and WebSTAR http://www.starnine.com for Mac OS, and WebSite http://www.oreilly.com; for Microsoft Windows. If you use a different server, consult your documentation to see if it supports SSIs.

[Footnote: Tenon's WebTen 2.1.9 is a point-and-click Web server based on Apache 1.2.6].

A sample web page (as the user's browser receives it) might look as follows:

Sample web page as seen in the browser

```
<HTML>
  <HEAD>
    <TITLE>
      Sample Web Page
    </TITLE>
  </HEAD>

   <H1>Sample Web Page</H1>
  <HR>

  <H2>Welcome to a Sample Web Page!</H2>
  <P>
    With some typical sample text...
```

```
     <P><HR><P>
    Copyright 1999 Prime Time Freeware<BR>
       Send comments, inquiries, or trouble reports to
<A HREF="mailto:www@ptf.com?subject=http://www.ptf.com/sample.shtml">www@ptf.com</A>.
   </BODY>
 </HTML>
```

This is generated by a much smaller HTML file (sample.shtml):

sample.shtml

```
 <!-#set var="page_title" value="Typical Web Page"  ->
 <!-#set var="page_uri"   value="${DOCUMENT_URI}"   ->
 <!-#include virtual="/h.shtml"                     ->

   <H2>Welcome to a Typical Web Page!</H2>
   <P>
     With some typical sample text...
   <!-#include virtual="/t.shtml"->
```

The man behind the curtain...

We'll agree with anyone who says that the syntax is clunky. On the other hand, it works reasonably well and is clear, once you understand the ground rules.

The code sets two variables (page_title and page_uri). The first is just a text string ("Typical Web Page"). The second, however, is a bit more involved.

We want the "mailto" HREF at the end of the page to include a subject line; specifically, the URL for the original HTML file. We can't hard-code the URL into the trailer file, however, because we want to use the same trailer file for many pages. Worse, the calling page's URI (Uniform Resource Identifier; the "local" portion of the URL) is not available to the trailer file.

We work around the problem by saving the value of the calling file's DOCUMENT_URI in page_uri. Our trailer file (/t.shtml) can then pick up and use this "global" information.

Once the variables have been defined, sample.shtml can invoke the header file (/h.shtml), fill in some body text, and invoke the trailer file (/t.shtml).

Now for the underlying details. First, the header file (/h.shtml):

```
/h.shtml
<HTML>
   <HEAD>
     <TITLE>
       <!-#echo var="page_title"->
     </TITLE>
   </HEAD>

     <H1><!-#echo var="page_title"-></H1>
     <HR>
```

This file combines "constant" text with echos of an inherited variable (page_title). Conveniently, we can use page_title twice.

The trailer file (/t.shtml) applies the same trick to generate the desired URL in the mailto's Subject line:

```
/t.shtml
<P><HR><P>
    Copyright 1999 Prime Time Freeware<BR>
       Send comments, inquiries, or trouble reports to
<A HREF="mailto:www@ptf.com?subject=http://www.ptf.com<!-#echo  var="page_uri"-
>">www@ptf.com</A   </BODY>
 </HTML>
```

Server-side includes also allow conditional evaluation (roughly, flow control), which we won't try to cover here. The Apache web pages (http://www.apache.org, http://www.apache.org/docs-1.2/mod/mod_include.html) are a big help, even if they get a bit terse from time to time.

How Do I Make Them Work?

Depending on which server you use, server-side includes may not be enabled by default. Although we develop many of our Web pages on a Macintosh, we deploy them on Apache (running on a FreeBSD server). In Apache, setting up server-side includes requires a bit of administrative effort, mostly in editing configuration files. If you use another server, consult your documentation for how to configure the server-side include mechanism.

The configuration process for Apache can be a little tricky, as there are several steps you must go through. If you don't have an Apache wizard to call upon, you may want to use the following recipe. (Even if you don't use Apache, you may want to skim the following section, as it will give you some clues to what is going on "under the covers".)

In the srm.conf file, uncomment the AddHandler and Addtype lines for .shtml files:

```
# To use server-parsed HTML files
AddType text/html .shtml
AddHandler server-parsed .shtml
```

Note that neither the server nor the browser really care what suffix you use for these files, only that you specify it here. You could specify the conventional .html ending here, in which case all of your .html files would be scanned for include directives. This causes a lot of extra work for your server, however, and may hurt your response times. Besides, use of a special file name extension is a good way to remind yourself which files use SSIs!

Next, create (or modify) the Directory Options directive in either httpd.conf or access.conf. (we recommend htpd.conf). In the Directory part, be sure to name the root of the directory tree for which server-side includes will be enabled. Note that this must be a full, OS (not WWW) path name; that is, it is rooted in the filesystem, not the DocumentRoot.

```
<Directory /usr/local/Server/WWW/web>
Options Indexes IncludesNoExec FollowSymLinks
</Directory>
```

It is very important to get the options right. Most Apache options are on (by default) if you don't explicitly set them; only a few are not. However, if you do explicitly set certain options, you must then explicitly set the all others you want. That is, the defaults only work as long as you use only the defaults.

Options, in detail...

To enable SSI, you must choose one of two possible inclusion options. To enable all forms of server-side includes, use the Includes option. This option turns on variables as well as allowing program (CGI) execution. If you just want variable substitution and file inclusion, but not program execution, use IncludesNoExec. This option only controls CGI execution from inside .shtml files; it has no connection to general CGI execution (e.g., from a URL).

You must also explicitly set several other options if you wish to use the associated features. If you want to be able to execute server-side CGI scripts from any directory (not just cgi-bin), you'll need to set the option ExecCGI. This option is related to the SSI options, but also controls execution of CGI scripts, exclusive of .html files. Consider wisely, as opening up CGI execution could lead to problems (e.g., if you have many users creating their own pages).

Set the Indexes option if you want to allow the web server to access directories that do not have an explicit index.html file. If you intend to use symbolic links (aliases, in Mac OS jargon) within your Web page hierarchy, be sure to set the FollowSymLinks option.

If you're unsure how the options will work, try them out. Create a few HTML directories with and without index.html files, add a few symbolic links, test out server-side includes with some variables, and try to execute a few CGI scripts. This is definitely a case where you can learn best by doing. You may also want to consult a book or two. How to Set Up and Maintain a Web Site, by Lincoln Stein (2nd. ed. with CD-ROM) is an excellent resource.

Try it Out

Now you're ready to include something! The format of a server side include (in your HTML code) is:

<!-#directive param1="value1" param2="value2" .. ->

Note that an include directive looks suspiciously like a comment; the difference is the hash mark and the directive immediately following the dashes at the front.

```
<!- this is a comment ->
<!-#include ... ->
```

The directive can be any of several choices. Apache supports the following directives; other servers may support more. Again, check your documentation.

config	control and configure various aspects of SSI parsing
echo	echo (print) the value of a variable
exec	execute a named CGI script; the IncludesNoExec option disables this
include	insert the text from another document at this point, unless including the document would cause a program to be executed and the IncludesNoExec option is set
fsize	include the size of the specified file
flastmod	include the last modification date of the specified file
printenv	print out a listing of all environment variables and their values (Apache 1.2 and later)
set	set the value of a variable (Apache 1.2 and later)

As you can see, SSIs provide a reasonable level of flexibility. With a simple SSI directive, you can easily include a dynamic "Last modified On..." line in your HTML documents. You can include any of the standard variables from the

server's environment, or you can set, and include, your own variables.

```
<!-#set var="page_title" value="Typical Web Page"  ->
<!-#set var="page_uri"   value="${DOCUMENT_URI}"   ->

<!-#echo var="page_title"->
```

As we saw in the examples at the beginning of this discussion, we make extensive use of the variable setting (and echoing) capabilities of SSIs. Note that the set directive was added more recently than the others; if it is not available in your server implementation, you should contact your vendor to determine if an upgrade is available.

Including Files

Including files is perhaps even more useful than including variables. We like to have a standard "boilerplate" footer at the bottom of our pages, but what do we do if something changes? For example, we probably want to update our Copyright notice with the new year!

There are two ways to include the contents of a file; the syntax depends upon the type of file you are including. To include a file named in relation to the DocumentRoot or by full URL, use the virtual parameter.

```
<!-#include virtual="/t.shtml"->
```

(It may seem odd to use the word "virtual" when you are specifying an absolute location in the file tree hierarchy, but that's the way it is). Alternatively, to include a file from the current directory, use the file parameter.

```
<!-#include file="righthere.html"->
```

In neither case can you include the contents of a file beyond the server root. In addition, the file parameter will accept only relative pathnames from the current point downwards, rejecting both absolute pathnames as well as any path beginning with "..".

If the file you are including also makes use of SSIs, its name must also end in the .shtml extension. When we include t.shtml, it resolves the page_uri variable to dynamically set the webmaster's mailto link at the bottom of the page - includes within includes!

t.shtml with variable-setting

```
    <P><HR><P>
    Copyright 1999 Prime Time Freeware<BR>
       Send comments, inquiries, or trouble reports to
 <A HREF="mailto:www@ptf.com?subject=http://www.ptf.com<!-#echo
   var="page_uri"->">www@ptf.com</A>.
   </BODY>
 </HTML>
```

Bibliography and References

Apache is the industry standard Open Source Web Server, used by more than half the hosts on the Internet. See http://www.apache.org for additional information and details. See http://www.apache.org/docs-1.2/mod/mod_include.html for details regarding server-side includes.

Rich Morin has been programming computers for thirty years. He has been a freelance computer programmer, consultant, technical writer and small-time entrepreneur since 1975. Rich currently operates Prime Time Freeware,

a publisher of mixed-media book/CDs relating to Open Source software. Rich lives in San Bruno, on the San Francisco peninsula.

Vicki Brown has been programming computers for about 20 years now. She discovered Unix in 1983 and the Macintosh in 1986. Unix is her favorite OS, but the Mac OS is her favorite user interface. Vicki is currently employed as a Perl programmer for a Silicon Valley Biotech firm. She is the co-author of MacPerl: Power and Ease. When she's not programming or writing, Vicki enjoys relaxing with her spouse and their two Maine Coon cats.