CHAPTER 12

Directory Listings

The default Apache HTTP Server package includes a module, *mod_autoindex*, for displaying a directory listing as a Web page. The default display is simple and informative, but the module provides all sorts of controls to let you tweak and customise the output.

12.1 Generating Directory/Folder Listings

Problem

You want to see a directory listing when a directory is requested.

Solution

Turn on *Options Indexes* for the directory in question.

```
<Directory /www/htdocs/images>
    Options +Indexes
</Directory>
```

Discussion

When a URL maps to a directory or folder in the filesystem, Apache will respond to the request in one of three ways:

1. If *mod_dir* is part of the server configuration, *and* the mapped directory is within the scope of a *DirectoryIndex* directive, *and* the server can find one of the files identified in that directive, then the file will be used to generate the response. *OR*
2. If *mod_autoindex* is part of the server configuration and the mapped directory is within the scope of an *Options* directive that has enabled the *Indexes* keyword, then the server will construct a directory listing at runtime and supply it as the response. *OR*
3. The server will return a 404 ("Resource Not Found") status.

monospace

Enabling directory listings

The real keys to enabling the server's ability to automatically generate a listing of files in a directory are the inclusion of *mod_autoindex* in the configuration and the Indexes keyword to the *Options* directive. This can be done either as an absolute form, as in:

```
Options FollowSymLinks Indexes
```

or in a selective or relative form such as:

```
Options -ExecCGI +Indexes
```

Enabling directory listings should be done with caution. Because of the scope inheritance mechanism (see *http://httpd.apache.org/docs/2.2/sections.html#mergin* for more details), directories farther down the tree also will be affected; and because the server will apply the sequence of rules listed at the beginning of this section in an effort to provide some sort of response, a single missing file can result in inadvertent exposure of your filesystem's contents.

Disabling Directory Indexing Below an Enabled Directory

There are essentially two ways to work around this issue and ensure that the indexing applies only to the single directory:

- Add an "*Options* -Indexes" to *.htaccess* files in each subdirectory, or
- Add an "*Options* -Indexes" to a <*Directory*> container that matches all the subdirectories.

For example, to permit directory indexes for directory */usr/local/htdocs/archives* but not any subdirectories thereof,

```
<Directory /usr/local/htdocs/archives>
    Options +Indexes
</Directory>

<Directory /usr/local/htdocs/archives/*>
    Options -Indexes
</Directory>
```

If this needs to apply only to certain subdirectories, the task becomes a little more complex. You may be able to accomplish it with a <*DirectoryMatch*> directive, if the list of subdirectories is reasonably small:

```
<Directory /usr/local/htdocs/archives>
    Options +Indexes
</Directory>

<DirectoryMatch /usr/local/htdocs/archives/(images|video|audio)>
```

```
    Options -Indexes
</DirectoryMatch>
```

See Also

- *http://httpd.apache.org/docs/2.2/mod/core.html#options*
- *http://httpd.apache.org/docs/2.2/mod/mod_dir.html*
- *http://httpd.apache.org/docs/2.2/mod/mod_autoindex.html*

12.2 Display a Standard Header and Footer on Directory Listings

Problem

You want to display a header above and footer below your directory listing.

Solution

```
# Remove the standard HTML header, if desired
IndexOptions +SuppressHTMLPreamble
HeaderName /includes/header.html
ReadmeName /includes/footer.html
```

Discussion

The directives *HeaderName* and *ReadmeName* specify the URI of files to be used as a header and footer, respectively, for directory listings.

If your *HeaderName* file contains an HTML <head> tag, <title> tag, or other things associated with the start of an HTML document, you will want to use the *IndexOptions +SuppressHTMLPreamble* directive to disable *mod_autoindex*'s automatically generated HTML heading. Failure to do so will result in an HTML document with two heading elements, with the result that any heading attributes set in your header will probably be ignored by the browser.

The argument to both *HeaderName* and *ReadmeName* is a URI relative to the current directory. That is, if there is no leading slash, it is interpreted as a path relative to the current directory, but if there is a leading slash, it is interpreted as a URI path—that is, relative to the *DocumentRoot*.

Your *HeaderName* and *ReadmeName* can be arbitrarily complex, to produce whatever page layout you like, wrapped around the auto-generated directory listing. You could, for example, open `<table>` or `<div>` sections in the header, which you then close in the footer, in order to produce page layout effects.

See Also

- *http://httpd.apache.org/docs/2.2/mod/mod_autoindex.html*

12.3 Applying a Style Sheet

Problem

You want to apply a CSS style sheet to a directory listing without supplying a whole HeaderName document.

Solution

```
IndexStyleSheet /styles/listing.css
```

Discussion

- *http://httpd.apache.org/docs/2.2/mod/mod_autoindex.html*

12.4 Hiding Things from the Listing

Problem

You want to omit certain things from the directory listing.

Solution

```
IndexIgnore *.tmp *.swp .svn secret.txt
```

Discussion

Certain files should be ommitted from directory listings. Temporary files, swap files, and various other generated files don't need to be shown to users visiting your Web site. Revision control directories, such as the *CVS* directory created by CVS, or the *.svn* directory created by Subversion, also should not be displayed, as it is unlikely to contain any information that would be of use to your visitors.

Although this technique can be used to hide private or secret documents, it must be understood that these files can still be accessed by someone who knows, or guesses, the filename. The files are hidden from the directory listing, but they are still accessible. Do not use this technique with an expectation of security.

Files that are password-protected are automatically omitted from directory listings.

See Also

- Recipe 12.19

- *http://httpd.apache.org/docs/2.2/mod/mod_autoindex.html*

12.5 Searching for Certain Files in a Directory Listing

Problem

You want to provide a way to filter the listing by filename.

Solution

Use a P (pattern) argument in the *QUERY_STRING* of the URL:

*http://servername/directory/?P=a**

Or place the following HTML form in a *HeaderName* file to provide a search feature on a directory listing.

```
<form action="" method="get">
Show files matching <input type="text" name="P" value="*" />
<input type="submit" value="Go" />
<form>
```

Discussion

In Apache 2.0.23, a number of new options were added to *mod_autoindex*, which allowed for more client control over the output of directory listings. By inserting options in the *QUERY_STRING* of the URL, changes can be made to the sort order, output formatting, and, as shown in this recipe, the files which are shown in the listing.

Using the *?P= QUERY_STRING*, the file listing is filtered by the supplied argument. For example, with a URL of: *http://servname/directory/?P=a**, any file starting with `a` will be listed.

Because this functionality is new with 2.0, there is not a way to accomplish the same thing with earlier versions of Apache.

See Also

- Recipe 12.16
- Recipe 12.2

12.6 Sorting the List

Problem

You want to sort the directory listing by something other than the defaults.

Solution

```
IndexOrderDefault Descending Date
```

Discussion

The *IndexOrderDefault* directive allows you to specify, in your configuration file, or *.htaccess* file, the order in which the directory listing will be displayed, by default. If, for example, you wish to have file displayed with the most recent one first, you could use the directive shown in the Solution above.

The possible arguments to *IndexOrderDefault* are:

- Name—The file or directory name.
- Date—The date and time that the file was most recently modified.
- Size—The size of the file, in bytes.
- Description—The file description, if any, set with the *AddDescription* directive.

Any of these may be ordered *Ascending* or *Descending*.

The value of *IndexOrderDefault* may be overridden by the end user by supplying *QUERY_STRING* arguments, unless you explicity forbid it using *IndexOptions Ignor-eClient*.

See Also

- Recipe 12.17
- Recipe 12.7

12.7 Allowing a Client-Specified Sort Order

Problem

You want to allow the end user to specify the order in which the listing should be ordered.

Solution

Users can supply *QUERY_STRING* arguments to modify the sort order:

http://servername/directory/?C=D&O=D

Or you can provide a form allowing the user to select the sort order, by placing the following form in a *HeaderName* file:

```
<form action="" method="get">
Order by by <select name="C">
<option value="N" selected="selected"> Name</option>
```

```
<option value="M"> Date Modified</option>
<option value="S"> Size</option>
<option value="D"> Description</option>
</select>
<select name="O">
<option value="A" selected="selected"> Ascending</option>
<option value="D"> Descending</option>
</select>
<input type="submit" value="Go" />
</form>
```

Discussion

Allowing the end user to control their user experience is a powerful way to make your Web content more useful.

Unless this feature has been explicitly disabled using the *IgnoreClient* argument to *IndexOptions*, you will always be able to reorder the directory listing using the *?C=* and *?O= QUERY_STRING* options.

O (order) can be set to either *A*, for Ascending, or *D*, for Descending, and *C* (column) may be set to one of the following:

- N—Name of the file or directory
- M—The last modified date of the file or directory
- S—The size of the file in bytes
- D—The description of the file, set with the *AddDescription* directive

The argument parsing routine will quit if it encounters an invalid argument.

With Apache 1.3, the syntax is, instead:

http://servername/directory/?X=Y

where `X` is one of **`n`**, **`m`**, **`s`**, or **`d`**, as described above, and `Y` is either **`a`**, for ascending, or **`d`**, for descending.

See Also

- Recipe 12.16
- Recipe 12.17
- Recipe 12.2

12.8 Specifying How the List Will Be Formatted

Problem

You want to specify different levels of formatting on the listing.

Solution

There are three levels of formatting that can be set. The list may be unformatted, or can be formatted, or can be rendered in an HTML table.

To enable fancy indexing, do the following:

```
IndexOptions FancyIndexing
IndexOptions FancyIndexing HTMLTables
```

Discussion

The "fancy" formatting is the one that you're most used to seeing, Because it's the default setting in most configurations of Apache.

The *HTMLTable* formtting is rather less common, and gives a slightly less plain look to the listing.

12.9 Allowing the Client to Specify the Formatting

Problem

You want to allow the end user to specify how the list will be formatted.

Solution

The user may specify which of the formatting options they wish to use by adding an `F` argument to the query string.

To specify a plain bulleted list:

```
http://www.example.com/icons/?F=0
```

To specify a formatted list:

```
http://www.example.com/icons/?F=1
```

To specify a list arranged in an HTML table:

```
http://www.example.com/icons/?F=2
```

Discussion

Unless *IndexOptions IgnoreClient* is in effect, the end user may apply a number of layout customizations by adding query string arguments. The `F` argument controls the formatting of the list.

See Also

- Recipe 12.16

12.10 Adding Descriptions to Files

Problem

You want to put a brief description of files in the listing.

Solution

Use the *AddDescription* directive to add a description to certain files or groups of files.

```
AddDescription "GIF image" .gif
```

Discussion

You may set a description for a particular file, or for any file that matches a particular pattern. The first argument to *AddDescription* is the description that you want to use, and the second is a substring that will be compared file names. Any file that matches the pattern will have the description used for it.

By default, you have 23 characters available for this description. That space can be altered explicitly by setting *IndexOptions DescriptionWidth*, or by suppressing one of the other columns.

You should ensure that the description isn't too long, or it will be truncated when it reaches the width limit. This can be annoying, when the description is truncated and therefore unreadable. Also, because you're permitted to use HTML in the description, it's possible that the HTML could be truncated, leaving unclosed HTML tags.

12.11 AutoGenerated Document Titles

Problem

You want to have the description of HTML files autogenerated.

Solution

Place the following in the <*Directory*> scope where you want to have descriptions automatically loaded from the <*Title*> tags of HTML files:

```
IndexOptions ScanHTMLTitles
```

Discussion

If generating a directory listing of a directory full of HTML files, it is often convenient to have the titles of those documents automatically displayed in the description column.

The *ScanHTMLTitles* option has *mod_autoindex* look in each HTML file for the contents of the <*Title*> tag, and use that value for the description.

This process is, of course, rather file-access intensive, and so will cause a significant performance degradation, proportional to the number of HTML files that are in the directory.

12.12 Changing the Listing Icons

Problem

You want to use different icons in the directory listing.

Solution

Use *AddIcon* and its variants to specify which icons are to be used by different kinds of files.

```
AddIcon /icons/image.gif .gif .jpg .png
```

Discussion

There are a number of variants of the *AddIcon* directive that allow you to associate certain icons with various files, groups of files, or types of files.

The *AddIcon* directive sets an icon to be used for files that match a particular pattern. The first argument is the URI of the icon file to be used. The argument or arguments following this are file extensions, partial file names, or complete file names, with which this icon should be used.

You also can specify the argument `^^DIRECTORY^^` for directories, or `^^BLANKICON^^` to be used for blank lines, to ensure correct spacing.

To specify an icon to be used for the parent directory link, use an argument of `".."`

```
AddIcon /icons/up_one.gif ".."
```

You also may use *AddIconByEncoding* to specify an icon to be used for files with a particular encoding such as, for example, `x-gzip`.

```
AddIconByEncoding /icons/gzip.gif x-gzip
```

And theres *AddIconByType* for associating an icon with a particular MIME type.

```
AddIconByType /icons/text.gif text/*
AddIconByType /icons/html.gif text/html
```

Finally, you can specify the default icon to be used if nothing else matches:

```
DefaultIcon /icons/unknown.png
```

With any of these directives, you also may specify an alternate text to be displayed for clients that have image loading turned off. The syntax for this is to include the alt text in parentheses, before the image path:

```
AddIcon (IMAGE,/icons/image.gif) .gif .png .jpg
```

See Also

- *http://httpd.apache.org/docs/2.2/mod/mod_autoindex.html#addicon*

12.13 Listing the Directories First

Problem

You want to have the folders (directories) listed at the top of the directory listing.

Solution

To have the directories displayed first in the directory listing, rather than in alphabetical order with the rest of the files, place the following in your configuration file:

```
IndexOptions FoldersFirst
```

Discussion

By default, directory listings are displayed in alphabetical order, including the directories. However, some people are used to having the directories at the top, followed by the files. This allows for faster navigation through deep directory structures.

Adding the *FoldersFirst* option puts the folders at the top of the listing, followed by the files in alphabetical order.

12.14 Ordering by Version Number

Problem

You want to have files ordered by version number, so that 1.10 comes after 1.9 rather than before 1.2.

Solution

To have files sorted in version number order, add the following to your configuration file:

```
IndexOptions VersionSort
```

Discussion

Sites that distribute software will often have multiple versions of the software in the directory, and it is useful to have them ordered by version number rather than alphabetically. In this way, *httpd-1.10.tar.gz* will be listed after *httpd-1.9.tar.gz*, rather than between *httpd-1.1.tar.gz* and *httpd-1.2.tar.gz*, as it would be in alphabetical order.

12.15 Allowing the End User to Specify Version Sorting

Problem

You want to let the end user enable or disable version sorting.

Solution

The user may specify whether to enable or disable the version ordering by adding a V query string argument to the URL:

To enable version ordering:

```
http://www.example.com/download/?V=1
```

To disable it:

```
http://www.example.com/download/?V=0
```

Discussion

Like the F argument, the V argument allows the user to impose their own custom formatting on a directory listing.

12.16 Complete User Control of Output

Problem

You want to combine some of the above techniques to give the end user full control of the output of a directory listing.

Solution

Place the following HTML in a file, and use it as the header for your directory listing:

```
<form action="" method="get">
Show me a <select name="F">
<option value="0"> Plain list</option>
```

```
<option value="1" selected="selected"> Fancy list</option>
<option value="2"> Table list</option>
<select>
Sorted by <select name="C">
<option value="N" selected="selected"> Name</option>
<option value="M"> Date Modified</option>
<option value="S"> Size</option>
<option value="D"> Description</option>
<select>
<select name="O">
<option value="A" selected="selected"> Ascending</option>
<option value="D"> Descending</option>
<select>
<select name="V">
<option value="0" selected="selected"> in Normal
order</option>
<option value="1"> in Version order</option>
<select>
Matching <input type="text" name="P" value="*" />
<input type="submit" value="Go" />
<form>
```

Discussion

Several of these recipes show how to let the end user specify formatting options in the query string. However, they're likely not going to know about this.

This recipe allows you to give the end user the full bag of tricks, and lets them select various formatting options right there in the page. If you save the above HTML as *header.html*, then you can use this in your directory listing with the *HeaderName* directive:

```
HeaderName /header.html
```

The user can than select various options, and reorder the listing, search for various strings, and alter the formatting of the output, to their heart's content.

12.17 Don't Allow the End User to Modify the Listing

Problem

You don't want the end user to be able to modify the output of the directory listing.

Solution

Place the following *IndexOptions* directive in the <Directory> scope where you wish this restriction to be in place.

```
IndexOptions +IgnoreClient
```

Discussion

Although it is generably preferable to allow the end user to have some control over their user experience, there may be times when you wish for a particularly directory listing to be presented in a particular way, without the option of a user to modify that display.

Although most users will probably be unaware of the ability to do so, by default any user can modify the output of the directory listing with a combination of *QUERY_STRING* arguments. With the recipe shown above, this feature is disabled.

When *IgnoreClient* is set, *SuppressColumnSorting* is also put into effect. That is, the clickable header at the top of each column is removed, so that the user isn't misled into thinking that they can alter the sort order by these links.

IgnoreClient can be used in conjunction with *IndexOrderDefault* to enforce a certain non-default directory listing order.

See Also

- *http://httpd.apache.org/docs/2.2/mod/mod_autoindex.html*

12.18 Suppressing Certain Columns

Problem

You don't want to show certain columns in the directory listing.

Solution

Various columns can be suppressed with one of the *Suppress** arguments to the *Index-Options* directive. For example, to suppress the last modified date column:

```
IndexOptions SuppressLastModified
```

Discussion

With the exception of the filename, all of the columns in a directory listing may be suppressed, using one of the following *IndexOptions* arguments:

- *SuppressDescription*—Hide the description column.
- *SuppressIcon*—Don't display the icon usually shown next to the filename.
- *SuppressLastModified*—Hide the column which lists the file datestamp.
- *SuppressSize*—Hide the column showing the file size.

See Also

- *http://httpd.apache.org/docs/2.2/mod/mod_autoindex.html*

12.19 Showing Forbidden Files

Problem

Password-protected files and directories don't show up in the directory listing.

Solution

If you're running Apache 2.2, place the following *IndexOptions* directive in a *<Directory>* block referring to the directory in question, or in a *.htaccess* file in that directory:

```
IndexOptions +ShowForbidden
```

If you're running Apache 2.0, there is no solution.

Discussion

Starting with Apache 2.0, directory listings attempt to protect protected documents. So if a file or directory requires password authentication, then it is not shown in a directory listing.

In Apache 2.0, this is simply how things work. There is no way to have forbidden files or directories shown in directory listings.

In Apache 2.2, the *ShowForbidden* argument was added for the *IndexOptions* directive, specifically to address this request.

See Also

- *http://httpd.apache.org/docs/2.2/mod/mod_autoindex.html*

12.20 Aliases in Directory Listings

Problem

Aliases don't show up in directory listings.

Solution

In the directory to be listed, put a file or directory named the same as the *Alias*. It will be displayed in the listing, but clicking on it will invoke the *Alias*.

Discussion

Aliases don't show up in directory listings, because *mod_autoindex* generates the listing by asking the filesystem for an actual directory listing. The filesystem doesn't know about the aliases.

There is no way to get *mod_autoindex* to pick up on these *Alias*es and list them.

You can, however, place items in the directory that act as placeholders for the *Alias*. Because an *Alias* is consulted before the filesystem, when you actually click on the file, the *Alias* will be invoked, and the file ignored.

See Also

- *http://httpd.apache.org/docs/2.2/mod/mod_autoindex.html*
- *http://httpd.apache.org/docs/2.2/mod/mod_alias.html*