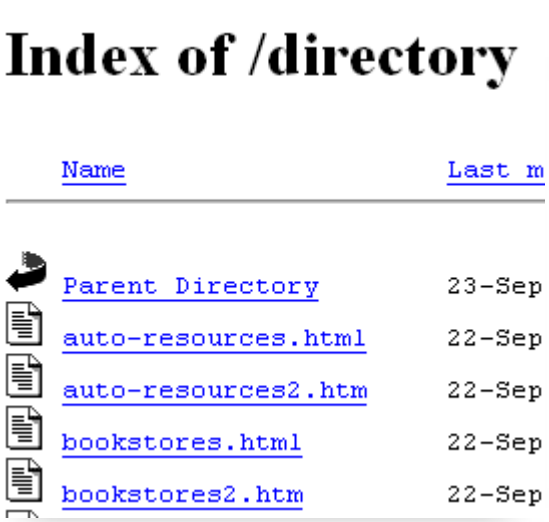
Better Default Directory Views with HTAccess

♦ Posted by Jeff Starr in .htaccess

UPDATED JULY 2, 2018 • 55 COMMENTS



Beautify your default directory listings! Displaying index-less file views is a great way to share files, but the drab, bare-bones interface is difficult to integrate into existing designs. While there are many scripts available to customize the appearance and functionality of default directory navigation, most of these methods are either too complicated, too invasive, or otherwise insufficient for expedient directory styling. In this comprehensive tutorial, you will learn how to use the built-in functionality of Apache's `mod_autoindex` module to style and enhance your default directory views with a smorgasbord of stylistic and functional improvements.

Before diving in..

Default directory views are very common on the Web. Any directory that does not contain a default index file, such as `index.html`, `index.php`, `default.asp`, or something similar, *may* present its contents via default directory view. Whether or not default directory views are enabled depends on several factors.

First, your server configuration must allow default directory listings. For shared hosting, this option may be specified via the account's control panel. Once directory listings are allowed at the server level, the directory itself needs to have its file permissions set to allow access (this is generally the case). Also, there must be no interference from other scripts, software, or other apps.

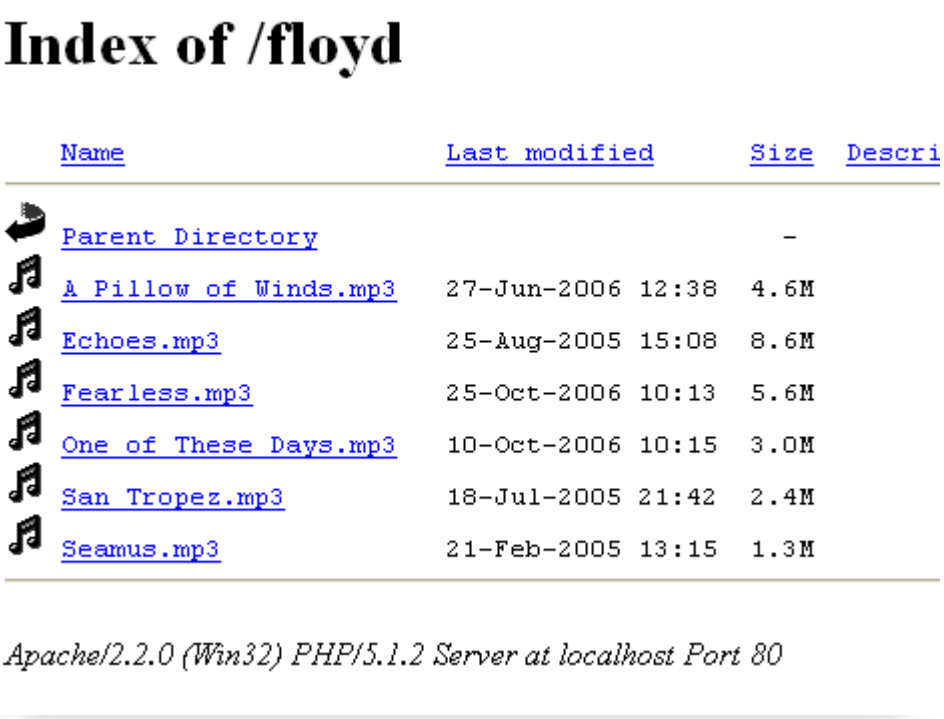
With all of these conditions met, you may gain granular control over which directories list their contents and which do not by using per-directory HTAccess files. Importantly, default directory views should be implemented *intentionally*, according to the specific content delivery requirements of the site. Unintentional display of directory contents may jeopardize site security if loopholes or potential exploits are discovered within unintentionally exposed scripts. For this reason, it is always a good idea to disable default directory views sitewide by default. On Apache-powered servers, this is easily accomplished with a single line of HTAccess:

```
# DISABLE DIRECTORY VIEWS
Options -Indexes
```

With that directive placed in the root HTAccess file of your site, directory listings will be disabled throughout your entire site. Then, once you have implemented this security measure, you may (re)enable directory views for any specific directory by creating an HTAccess file for the directory and adding the following directive to it:

```
# ENABLE DIRECTORY VIEWS
Options +Indexes
```

At this point, your site is protected against *unwanted* directory listings and ready to display directory contents via default view for a specifically chosen directory. For the sake of this tutorial, let's assume we are ~~sharing~~ displaying some choice Pink Floyd MP3s from within a directory called (creatively enough) "`floyd`":



Index of /floyd

	Name	Last modified	Size	Descri
	Parent Directory		-	
	A Pillow of Winds.mp3	27-Jun-2006 12:38	4.6M	
	Echoes.mp3	25-Aug-2005 15:08	8.6M	
	Fearless.mp3	25-Oct-2006 10:13	5.6M	
	One of These Days.mp3	10-Oct-2006 10:15	3.0M	
	San Tropez.mp3	18-Jul-2005 21:42	2.4M	
	Seamus.mp3	21-Feb-2005 13:15	1.3M	

Apache/2.2.0 (Win32) PHP/5.1.2 Server at localhost Port 80

Default directory view for the "floyd" directory

Now let's have a little fun styling, tweaking, and enhancing that boring default directory view with some HTAccess magic. For the sake of maintaining order throughout this tutorial, we will walk through the construction of a completely functional HTAccess file capable of transforming any default directory listing into a thing of functional and aesthetic beauty!

Step 1: Preliminary Directives

The first thing we want to do is secure our directory's HTAccess file against attacks. We add the following code to the HTAccess file of our target directory (the one for which we have enabled default directory views):

```
# STRONG HTACCESS PROTECTION
<Files ~ "^.*\.([Hh][Tt][Aa])">
 order allow,deny
 deny from all
</Files>
```

For more information on this technique, check out my article, Improve Site Security by Protecting HTAccess Files. Let's move on..

Step 2: Precautionary Measures

When implementing HTAccess directives, it is always a good idea to test for the presence of the required Apache module *before* its invocation. Apache has built-in module "testing containers" (or whatever they're called) that are designed for this purpose:

```
# DIRECTORY CUSTOMIZATION
<IfModule mod_autoindex.c>
 .
 .
 .
 [ HTAccess directives relying on mod_autoindex go here ]
 .
 .
 .
</IfModule>
```

Add that code to your now-expanding HTAccess file (make sure you remove the six dots and the bracketed information!). We will be inserting all subsequent code in between these opening and closing module containers.

Step 3: Configurational Tricks

Apache's `IndexOptions` directive provides many useful configurational options. Rather than list them all here, we will choose a couple of key options to get us started and then explore additional options as needed. For a complete list, check out the official mod_autoindex manual at the venerable Apache website.

Returning to our Pink Floyd MP3 directory, let's assume that we want to enable the following features:

- Fancy indexing — display of additional file information, including extra sorting and configurational features, file icons and descriptions, etc.; without this option enabled, directory contents are returned in a

simple unordered list format

- Folders first — always display any subfolders before individual files, regardless of specified sort order
- Auto-width Name column — we want the name column to automatically resize according to the length of the longest file/folder name; without this option enabled, long file and folder names will be truncated unless we specify an accommodating column width.
- Auto-width Description column — we want the description column to automatically resize according to the length of the longest file/folder name; without this option enabled, long file and folder descriptions will be truncated unless we specify an accommodating column width.
- Suppress the HTML preamble — by default, Apache automatically includes the opening HTML elements (e.g., `<html>`, `<head>`) for the markup comprising the directory listing; however, we want to disable this behavior because we will be using our own custom header (and footer) files.

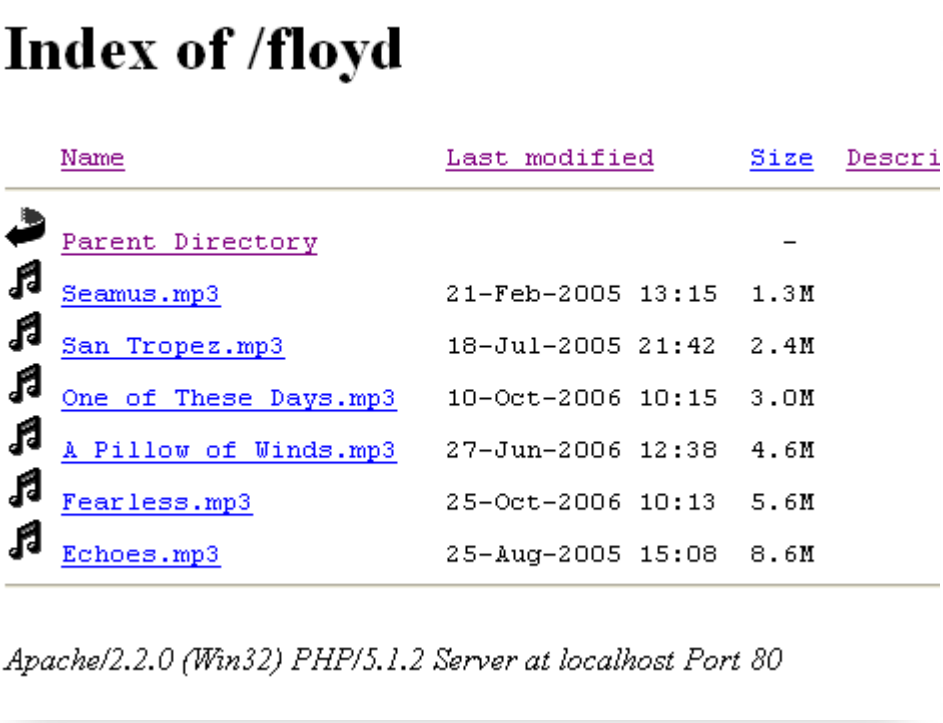
Amazingly enough, all of this functionality is invoked with a single line of code:

```
# SET INDEX OPTIONS
IndexOptions IgnoreCase FancyIndexing FoldersFirst NameWidth=* DescriptionWidth=*
SuppressHTMLPreamble
```

Notice the two options for specifying the column widths, `NameWidth` and `DescriptionWidth`. Note also the use of the wildcard operator (`*`) for the keen auto-width columns. Of course, any value (in pixels) may be specified here, depending on your specific needs. And, while we're at it, let's specify the default directory display order:

```
# SET DISPLAY ORDER
IndexOrderDefault Descending Name
```

After placing these directives within the previously specified `IfModule` container, our Pink Floyd directory view looks like this:



Index of /floyd

Name	Last modified	Size	Descri
Parent Directory		-	
Seamus.mp3	21-Feb-2005 13:15	1.3M	
San Tropez.mp3	18-Jul-2005 21:42	2.4M	
One of These Days.mp3	10-Oct-2006 10:15	3.0M	
A Pillow of Winds.mp3	27-Jun-2006 12:38	4.6M	
Fearless.mp3	25-Oct-2006 10:13	5.6M	
Echoes.mp3	25-Aug-2005 15:08	8.6M	

Apache/2.2.0 (Win32) PHP/5.1.2 Server at localhost Port 80

Options-modified directory view for the "floyd" directory

As you can see, not much has changed in terms of *appearance*, but don't worry, we'll get to that next..

Step 4: Accessing and Modifying Markup

The key to customizing the look and feel of the default directory listing is the ability to modify and customize the `<head>` region of the document markup. Even though they look plain and boring, default directory views are displayed via good 'ol fashioned HTML. Sadly formatted HTML perhaps, but HTML nonetheless. Each directory view consists of three different "chunks" of HTML:

- The Header — by default automatically generated by Apache
- The Directory Listing — necessarily generated by Apache
- The Footer — referred to as the "Readme" file

Fortunately, the Apache Wizards have provided a way to override the default header and footer files, thereby enabling us to include our own elements within the document `<head>`, like CSS styles, JavaScript, or anything else we desire. Likewise with the footer, we may add any sort of custom content, links, information, notes, or whatever. To specify custom header and footer files, we add this to our HTAccess file:

```
# SPECIFY HEADER FILE
HeaderName header.html
```

```
# SPECIFY FOOTER FILE
ReadmeName footer.html
```

Each of these two files, `header.html` and `footer.html`, must be created and properly located according to the specified path. In this case, we are simply placing the files in the same directory as the HTAccess file, so no other path information is required. These files may be placed anywhere on the server, however, so long as they are accessible via correct file paths. Given our preconfigured options, here is the minimum amount of markup that must be included within the custom header file:

```
<html>
	<head>
		<title>Directory Title</title>
	</head>
	<body>
		<h1>Directory Title</h1>
```

Technically, we only need the opening `<html>` and `<body>` elements, but it is nice to include a title for the document as well. Note that by omitting the previously defined `SuppressHTMLPreamble` option, Apache will generate the required opening markup, regardless of whether or not a custom header file is included, and regardless of whether or not the custom header file already includes it. Now, to style the entire directory listing, we simply add our desired CSS styles to the `<head>` region. Before applying CSS styles, however, it helps to know which HTML elements we have at our disposal. With `FancyIndexing` enabled, directory listing markup includes the following elements:

- `body`
- `img`
- `pre`
- `h1`
- `hr`
- `a`

All default directory listings generated by Apache (with `FancyIndexing` enabled) employ these elements to construct the page. The challenging part of styling the default markup involves Apache's use of `<pre>` tags to enclose all markup for the directory listing itself (i.e., the middle part of the document which cannot be customized). Here is the markup used for our exemplary `/floyd/` directory:

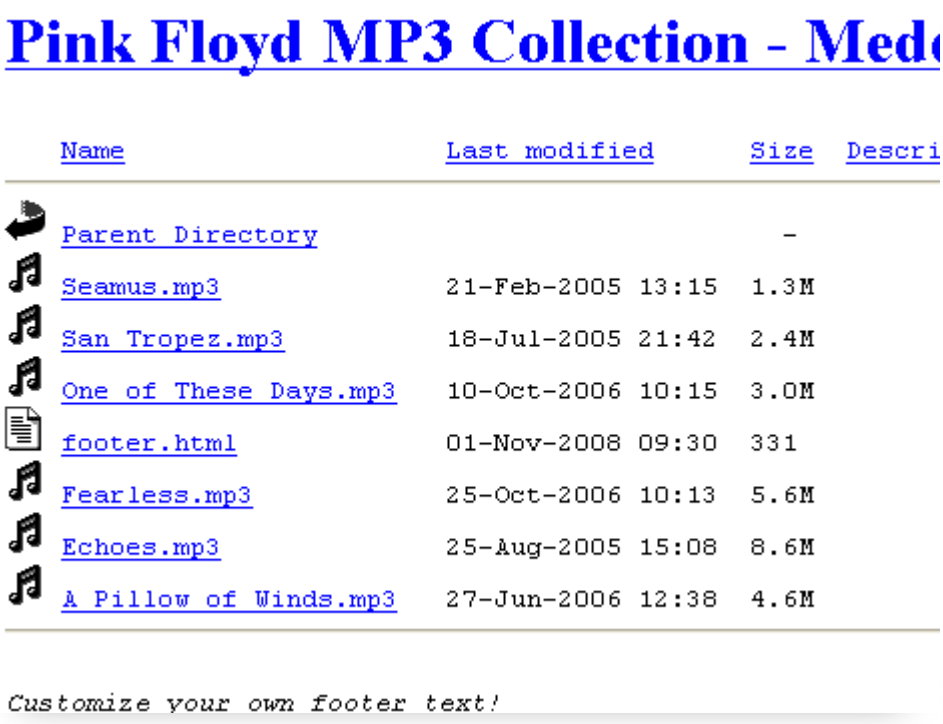
```
<pre><img src="/icons/blank.gif" alt="Icon "> <a href="?C=N;O=A">Name</a>                    <a
href="?C=M;O=A">Last modified</a>      <a href="?C=S;O=A">Size</a>  <a href="?
C=D;O=A">Description</a><hr><img src="/icons/back.gif" alt="[DIR]"> <a href="/">Parent
Directory</a>                             -
<img src="/icons/sound2.gif" alt="[SND]"> <a href="Seamus.mp3">Seamus.mp3</a>              21-
Feb-2005 13:15  1.3M
<img src="/icons/sound2.gif" alt="[SND]"> <a href="San%20Tropez.mp3">San Tropez.mp3</a>
18-Jul-2005 21:42  2.4M
<img src="/icons/sound2.gif" alt="[SND]"> <a href="One%20of%20These%20Days.mp3">One of These
Days.mp3</a>   10-Oct-2006 10:15  3.0M
<img src="/icons/sound2.gif" alt="[SND]"> <a href="Fearless.mp3">Fearless.mp3</a>            25-
Oct-2006 10:13  5.6M
<img src="/icons/sound2.gif" alt="[SND]"> <a href="Echoes.mp3">Echoes.mp3</a>              25-
Aug-2005 15:08  8.6M
<img src="/icons/sound2.gif" alt="[SND]"> <a href="A%20Pillow%20of%20Winds.mp3">A Pillow of
Winds.mp3</a>   27-Jun-2006 12:38  4.6M
<hr></pre>
```

As you can see, the designers of this portion of the markup are depending heavily upon the `<pre>` element for the layout of the page. As empty/white/blank spaces are preserved when presented as `pre`formatted text, the different columns are established via varying amounts of white space. This layout technique makes it difficult to style things like `line-height`s, `margin`s and `padding` for the individual lines of text. Fortunately, there are workarounds for this, such as styling the image elements to affect line height, etc..

Before styling our document, let's have a quick look at the custom footer file we will be using:

```
<pre><em>Customize your own footer text!<br />Add information and links!</em></pre>
<pre><a href="http://domain.tld/" title="Home">Return to Home Page</a>
<a href="http://domain.tld/search/" title="Search">Search this Website</a>
<a href="http://domain.tld/contact/" title="Contact">Contact Webmaster</a></pre>
</body>
</html>
```

Here, we could include any information we desire. For the sake of this tutorial, I have simply added a few generic links as an example. Note that you must close the `<body>` and `<html>` at the end of the document. Here is a screenshot showing our directory listing at this point:



"floyd" directory with custom header and footer [full view]

Step 5: Styling the Document with CSS

Before we begin styling the page, let's use HTAccess to hide the `footer.html` file from appearing in the directory list. Also, if we were using a name other than "header" for the header file, we would need to hide it as well. Regardless of what you decide to name these files, preventing their listing in the directory is as easy as adding the following line to your HTAccess file (beneath the custom header and footer directives):

```
# IGNORE THESE FILES
IndexIgnore header.html footer.html
```

Apache's `IndexIgnore` directives specify a space-delimited list of files and directories to ignore and exclude from the listing. Groups of files or file types may be selected via regular expressions and/or the use of wildcards. While we're here, let's add a few more items to our ignore list:

```
# IGNORE THESE FILES
IndexIgnore header.html footer.html favicon.ico .htaccess .ftpquota .DS_Store icons *.log *,v *,t
.??* *~ *#
```

Most of these items are commonly seen in various server directories and should not be displayed along with ordinary content. Feel free to add, delete, or modify this list of ignored items to suit your specific needs. For now, let's get on with the styling of the page.

Hopefully, most of my readers are familiar enough with CSS that styling basic things like background colors, text sizes, and other basic properties is straightforward. Some fun things to do include styling the heading element, default text, and the various link states. For our evolving `/floyd/` directory, we want a nice, light-grey background, monospace font selection, and dark link color with discernible `:hover` and `:visited` states. To do this, we add the following code to the `<head>` region of our custom header document like so:

```
<html><head><title>Pink Floyd - Meddle</title>
<style type="text/css">
body {
        background: #eee;
        margin: 33px;
        color: #333;
        }
h1 {
        font: 2.0em Georgia, serif;
        }
h1 a:hover, h1 a:active {
        text-decoration: none;
        }
a:link {
        text-decoration: none;
        color: #555;
        }
a:visited {
        text-decoration: none;
        color: #777;
        }
a:hover, a:active {
        text-decoration: underline;
        color: maroon;
        }
pre {
        font: 0.9em/1.3em "Courier New", Courier;
        margin: 3px 0;
        color: #777;
        }
pre img {
        display: inline;
        }
img {
        margin: 3px 0;
        }
</style>
</head>
```

```
<body><h1><a href="http://domain.tld/mp3/floyd/" title="Pink Floyd - Meddle">Pink Floyd MP3
Collection - Meddle</a></h1>
```

With that in place, here is how our `/floyd/` directory looks now:

"floyd" directory with applied CSS styles [full view]

Now let's do something about those horrible default icons.

Step 6: Customizing File and Folder Icons

Apache's `mod_autoindex` module provides a great amount of flexibility and customization for the icons used in "fancy indexed" directory listings. The first thing we want to do when using default icons is to specify a default icon to be used for non-specified file types:

```
# DEFAULT ICON
DefaultIcon icons/generic.gif
```

For our example directory, we want to keep all custom icons located within the directory itself, so we create a folder called "`icons`" and place our `generic.gif` and all subsequent icons inside of it. Then, to hide the new `icons` folder from the directory listing, we add the following term to our `IndexIgnore` directive:

```
# IGNORE THESE FILES
IndexIgnore header.html footer.html icons
```

Looking at the default directory view, we see that there are many different icons that may be customized, including one that is disabled by default. This “blank icon” may be customized with any icon by adding the following directive to your HTAccess file:

```
AddIcon icons/purple.gif ^^BLANKICON^^
```

..which displays a purple icon in place of the previously unseen “blank” icon:

Default hidden icon replaced by a visible icon (note left-alignment of name link)

I prefer the default, blank icon, however, so I will disable the custom icon by “commenting out” the previous directive using a pound sign (`#`):

```
# AddIcon icons/purple.gif ^^BLANKICON^^
```

Moving on to other default icons, there are two key icons that you may want to customize: the “up a level” icon and the folder icon. Custom folder icons are specified using the following directive:

```
AddIcon icons/grey.gif ^^DIRECTORY^^
```

Similarly, the “up a level” icon is associated with two dots (`..`) in HTAccess, and may be customized like so:

```
AddIcon icons/green.gif ..
```

Likewise, individual icons may be specified in several ways. First, we may specify custom icons by matching any part of the file extension (using pattern matching). Here are some examples that we will include in our finished HTAccess file:

```
# SPECIFIC FILE ICONS
AddIcon icons/blue.gif .txt .pdf .zip .rar .jpg .jpeg .jpe .png .gif .mpg .ico .js .log .doc .css
.html
AddIcon icons/red.gif readme
AddIcon (MP3,icons/arrow.gif) .mp3
```

With the first line, any of the listed file types will sport a nifty blue icon. With the second line, any files containing the character string, “readme” will feature a nice red icon. And finally, in the third line, we are associating all MP3s with a small, grey arrow. Also, the format used in the third line allows us to define the `alt` text used by the image element displaying our image. If the image should become unavailable for some reason, the `alt` text will be displayed instead.

Another way to sepcify custom icons is by type. By targeting the MIME-type of the file, we may target, say, all image files and associate them with a specific type of icon:

```
# CUSTOM IMAGE ICONS
AddIconByType (IMG,icons/image.gif) image/*
```

And finally, we may specify custom icons by encoding:

```
# GZIP ENCODING ICON
AddIconByEncoding (CMP,icons/compress.gif) x-compress x-gzip
```

With these three options for specifying custom icons, the configurational possibilities are endless. It all depends on your specific directory-listing needs. Other kewl things you can do with your custom icon play include specifying a custom height and width for your icons, and also turning your icons (custom or not) into links! To do so, add the following parameters to your `IndexOptions` directive:

```
IconHeight=16 IconWidth=16 IconsAreLinks
```

Although for our example, we will comment out these parameters as they are not needed. (Still with me? — Your stamina is great!) Here is how our custom `/floyd/` directory looks with teh custom icons added:

"floyd" directory listing featuring custom icons [full view]

Ahh, looking much better.. Now let's wrap things up with some custom descriptions for the various file types and subdirectories.

Step 7: Adding Descriptions to Folders and Files

Custom descriptions may also be added for any specific file, file type, or folder. Depending on your needs, descriptions may be overkill, but they are useful for providing highlighting key files and folders, explaining various items, and providing extra information in general. This list of descriptions for common file types should help you understand how to include descriptions for your own directories:

```
AddDescription "MPEG Layer 3 Format" .mp3
AddDescription "GZIP compressed TAR archive" .tgz .tar.gz
AddDescription "GZIP compressed archive" .Z .z .gz .zip
AddDescription "RAR compressed archive" .rar
AddDescription "TAR compressed archive" .tar
AddDescription "ZIP compressed archive" .zip
AddDescription "Windows executable file" .exe
AddDescription "Common Gateway Interface" .cgi
AddDescription "Joint Photographics Experts Group" .jpg .jpeg .jpe
AddDescription "Graphic Interchange Format" .gif
```

```
AddDescription "Portable Network Graphic" .png
AddDescription "Vector graphic" .ps .ai .eps
AddDescription "Hypertext Markup Language" .html .shtml .htm
AddDescription "Cascading Style Sheet" .css
AddDescription "DocType Definition" .dtd
AddDescription "Extensible Markup Language" .xml
AddDescription "Win32 compressed HTML help" .chm
AddDescription "Adobe Portable Document Format" .pdf
AddDescription "Plain text file" .txt .nfo .faq .readme
AddDescription "Unix man page" .man
AddDescription "Email data" .eml .mbox
AddDescription "Microsoft Word document" .doc
AddDescription "PHP: Hypertext Preprocessor script" .php  .php3 .php4
AddDescription "PHP: Hypertext Preprocessor source code" .phps
AddDescription "Javascript" .js
AddDescription "Java code"  .java
AddDescription "Unix shell script" .sh .shar .csh .ksh .command
AddDescription "Mac OS X shell script" .command
AddDescription "Configuration file" .conf
AddDescription "Mac OS X terminal" .term
AddDescription "BitTorrent file" .torrent
AddDescription "Windows link" .lnk .url
```

See the pattern there? It's a lot of fun to play around with descriptions, as there are many different things you can do with them. For example, you can actually include HTML along with your descriptions:

```
# FILE DESCRIPTIONS
AddDescription "<span class='description'>MPEG Layer 3 Format</span>" .mp3
AddDescription "<span class='description'>GZIP compressed TAR archive</span>" .tgz .tar.gz
AddDescription "<span class='description'>GZIP compressed archive</span>" .Z .z .gz .zip
AddDescription "<span class='description'>RAR compressed archive</span>" .rar
AddDescription "<span class='description'>TAR compressed archive</span>" .tar
AddDescription "<span class='description'>ZIP compressed archive</span>" .zip
AddDescription "<span class='description'>Windows executable file</span>" .exe
AddDescription "<span class='description'>Common Gateway Interface</span>" .cgi
AddDescription "<span class='description'>Joint Photographics Experts Group</span>" .jpg .jpeg
.jpe
AddDescription "<span class='description'>Graphic Interchange Format</span>" .gif
AddDescription "<span class='description'>Portable Network Graphic</span>" .png
AddDescription "<span class='description'>Vector graphic</span>" .ps .ai .eps
AddDescription "<span class='description'>Hypertext Markup Language</span>" .html .shtml .htm
AddDescription "<span class='description'>Cascading Style Sheet</span>" .css
AddDescription "<span class='description'>DocType Definition</span>" .dtd
AddDescription "<span class='description'>Extensible Markup Language</span>" .xml
AddDescription "<span class='description'>Win32 compressed HTML help</span>" .chm
AddDescription "<span class='description'>Adobe Portable Document Format</span>" .pdf
AddDescription "<span class='description'>Plain text file</span>" .txt .nfo .faq .readme
AddDescription "<span class='description'>Unix man page</span>" .man
```

```
AddDescription "<span class='description'>Email data</span>" .eml .mbox
AddDescription "<span class='description'>Microsoft Word document</span>" .doc
AddDescription "<span class='description'>PHP: Hypertext Preprocessor script</span>" .php  .php3
.php4
AddDescription "<span class='description'>PHP: Hypertext Preprocessor source code</span>" .phps
AddDescription "<span class='description'>Javascript</span>" .js
AddDescription "<span class='description'>Java code</span>"  .java
AddDescription "<span class='description'>Unix shell script</span>" .sh .shar .csh .ksh .command
AddDescription "<span class='description'>Mac OS X shell script</span>" .command
AddDescription "<span class='description'>Configuration file</span>" .conf
AddDescription "<span class='description'>Mac OS X terminal</span>" .term
AddDescription "<span class='description'>BitTorrent file</span>" .torrent
AddDescription "<span class='description'>Windows link</span>" .lnk .url
```

Then, with the classed `spans` in place, you can add something like the following to the CSS in your custom `header.html` file:

```
.description {
        font-style: italic;
        font-size: 90%;
        color: #777;
        }
```

..which would then have the following effect on our default directory listing:

"floyd" directory listing featuring styled descriptions [full view]

You may also want to add a default description for anything that you may have missed (bonus points if you call me on this!):

```
# DEFAULT DESCRIPTION
AddDescription "[<span class='description'>unknown item..</span>]" *
```

While we're at it, let's see how subfolders and their respective descriptions might be included within our incredible `/floyd/` directory. After creating three folders, "`Images`", "`Lyrics`", and "`Notes`", we add the following directives:

```
# FOLDER DESCRIPTIONS
AddDescription "[<span class='description'>Pink Floyd Images</span>]" Images
AddDescription "[<span class='description'>Pink Floyd Lyrics</span>]" Lyrics
AddDescription "[<span class='description'>Pink Floyd Notes</span>]" Notes
```

..which gives us this final result:

"floyd" directory listing with three subfolders and their descriptions [full view]

Alright, I can't take any more! Let's close this thing!!

Bonus Tips!

Tip #1 You can add the following tag the header file to make links open in a new tab/window:

```
<base target="_blank">
```

Should be placed anywhere in the `head` element.

Tip #2 If you can't get the `header.html` file to display, try adding `AddType text/html .html` to your .htaccess file.

Thanks to Mike Epler for these tips!

En Closure

In this rather extensive tutorial, we have transformed a default directory listing into a completely customized, fully functional presentation of directory contents. Using nothing more than HTAccess and a little HTML & CSS, any Apache-powered directory listing is under your complete and utter control. No fancy-pants third party scripts or programming tricks required. It's all self-contained, self-sustaining, and wonderful. Have fun with this method, and use it to transform your boring default directories into beautifully styled works of well-designed bliss. Or whatever!! I don't care 'cuz I am finally done with article and it's time to chill!!!! Peace!!!

References

- Apache Module mod_autoindex — Official Manual

55 responses to "Better Default Directory Views with HTAccess"

Louis 2008/11/02 9:59 AM

Two little remarks about this post of yours Jeff:

> *Displaying index-less file views is a great way to share files*

This statement may be a little confusing as the result of not putting any index.html or index.php is the display of another *index*; the list of the files existing in the current directory is called an index too.

I totally understand the meaning of your phrase, but it might be a little confusing for someone who's not used to raw Apache directory listings.

> *I don't care 'cuz I am finally done with article and it's time to chill!!!! Peace!!!*

"with *this* article", I think you were in such hurry to finish it that you did this little typo at the end. Also, that was your most quick & dirty conclusion ever ! Very funny indeed :p

—

Now, more on the topic of this very post, I have a bottom question.

These default directory listings are what Apache provides when the webmaster hasn't even bothered putting an index.xxx file in the directory, or denying listings – as I always do.

Now, the whole point of this article of yours is to explain how to style these emergency listings, and what I don't understand is : if someone has the time/motivation to enhance the visual appearance of his data – that is to say, *build a proper interface* – why wouldn't he go for a real solution, like building a proper index.xxx file or a real PHP system, instead of enhancing something that is meant to be an emergency solution?

Also, I don't think that – except in some very specific situations – it's a good idea to allow directories listing.

When you build a site, you are building an application, with an interface. You build pages for displaying choosen content, and in the ideal situation, you know exactely what your visitors can see, and on which pages they can see it.

It seems to me that allowing visitors to have access to the coulisses is synonym with a broken interface : if there is a need to display listing, make these listings part of the interface, and make them have their own official space.

Don't you think?

Jeff Starr 2008/11/03 11:54 AM • POST AUTHOR

Yes, this article was a killer to write! I kept putting it off until, finally, I just had to get it done. After spending too many hours on it, I admit, I just didn't have the patience required to fashion a more robust conclusion. Hopefully, the number of readers who actually make it all the way through the article will be few! ;)

—

To address your other concerns, the reason I felt that this information is useful is that I have I used default directory listings on many occasions; they work great for sharing mp3s and other bulk files, and I also use them for various FTP accounts for myself and clients. In an effort to customize these listings, I have tried many different techniques, mostly in the form of third-party folder-navigation scripts that usually provide

waay more functionality than required. After discovering how easy it is to enhance the look, feel, and functionality of these default directories using a few lines of htaccess, I thought that others would find the information useful as well.

May not be for everyone, of course, but we now have a nice reference for those that would like to use it.

Louis 2008/11/03 2:48 PM

Yes I see, after all, it may be possible to build a quite sexy interface starting only with the Apache listing view :)

Jessi Hance 2008/11/04 8:09 AM

There's a good reason to prefer this method to manually creating an index page. Once you set this up, you can throw any files you like in the directory, and there's your beautiful customized index that uses the same stylesheet as the rest of your site. You can add or remove files any time you like without having to make any other changes. That's why it's called autoindex.

And I don't know why you'd prefer using a php script when .htaccess will do the job so nicely. Though I love php for the right job.

Jessi Hance 2008/11/04 8:18 AM

Another follow-up. Some of the comments seem to imply that you'd be losing control over access to your site. That's wrong. This tutorial quite clearly instructs you to turn on indexing only in the desired directory. You're choosing to allow access to a specific directory in order to have the convenience of auto-indexing.

It's like a box of free stuff in your house or store, where visitors can look through and take anything they like. Sometimes you want to give things away.

Louis 2008/11/04 11:52 AM

Hi Jessi,

I think you misinterpreted what I said. My point is not that using HTAccess and HTML/CSS only makes you lose control over your content, no no; my point is that it is relatively limitated compared with what you can build in terms of interface, using an external langage such as PHP or Ruby.

If I can summup my thoughts, it would be: if you have the time and motivation to make something good looking and usable, why building up something from the Apache emergency listing, and not build a proper interface using a more advanced technic (PHP, ruby, etc)?

You two critics are, in order:

> *You can add or remove files any time you like without having to make any other changes. That's why it's called autoindex.*

PHP is meant for this too, I think you'd agree. Once again, if you are taking the time to build a nice experience for the visitor, you may want to consider going the whole way and use PHP (or another langage).

> *You're choosing to allow access to a specific directory in order to have the convenience of auto-indexing.*

(a) you want a quick & dirty solution to share files on the internet, and you have no time ahead to implement it. Then, I agree that it makes perfect sense to use Apache to list these files quickly – though it clearly may be confusing for users that are not familiar with this particular display.

(b) you want to build a great experience, so why limitate to custom Apache listing when you have the world in your hands with PHP, Ruby, Python, etc?

Bottom line:

I don't think it's a bad practice to use Apache default listings! I simply question myself about the purpose of tweaking it, when you can build a real interface, using a langage that is meant for doing such task.

Jessi Hance 2008/11/04 1:58 PM

Hi Louis,

Sorry I misunderstood. You seemed to imply that an Apache-generated auto-index is somehow less "real" or "proper." You called it an "emergency" solution. I don't really understand that perspective.

Perhaps what you're getting at is that with a script, you can generate the rest of the page, not just the index listing itself, dynamically? I would agree with that, unless you do an additional small mod to your .htaccess to enable php in your index header and footer to be parsed.

I haven't had a need for anything more than what Apache+xhtml+css can do, but I can certainly understand that others would want even more power and flexibility.

I also understand that everyone has their favorite tools. Nothing wrong with that. I just don't think that one tool is more "real" than another.

Louis 2008/11/04 2:33 PM

Yes, sorry if my words are inappropriate sometime. I have the worst difficulties expressing myself in english when it comes to complex explanations. In french, I use a lot of idiomatic expressions, and when I need to express that in english, I'm often uncapable of doing it properly, so I simplify me phrases.

I hope that I can go to Canada to study english seriously next year, with my studies.

So, to react to what you where saying: yes, indeed, I do understand your position. Though, I see this solution as quick answer to a problem. If it needs to be quick & dirty, okay then; but if there is a popular request from the visitors to have a way to get files, then, I'll want to build a rich interface, and that often mean, using a langage such as PHP (if I want a search engine, a sorting feature, etc).

In this former case, I don't see the point of doing "an additional small mod to your .htaccess to enable php in your index header and footer to be parsed." Why wouldn't we use PHP or another dynamic langage as you do for everything else in your site.

To me, it makes it like the list of the files is not a part of your application (site, blog, webapp). It's separated from the rest of the code of your application, and it's a different technology/approch.

What if my app requires a login, what if the other pages are using elements that require a specific technology (RoR for example)? It seems like a break to me, that you would tweak Apache to display what your need, instead of making the listing *part of the application*.

Jessi Hance 2008/11/04 5:02 PM

Aha, now I'm understanding a whole lot better, Louis! Thanks for supplying all those details. Yes, the features you mentioned sound like they warrant a different solution than this one. Your site sounds rather sophisticated – you refer to it as an application, not simply a site. So I can understand why you would take a different approach.

I have built a few humble sites with php, and for those, this method fit well and was a perfectly elegant solution and made handsome, useful pages that looked just like the rest of my site. And I just want to mention that Apache auto-indexes are indeed sortable by default. :-)

Maybe part of the reason I love this method is that it is easy enough for someone like me! I will leave the fancier techniques to the pros like you! :-D

Bleyder 2008/11/05 5:48 AM

Jeff, very interesting article. I have learned some aspect of the .htaccess files that I ignore.

I thank you your work for the web design community, with post like this.

August Klotz 2008/11/05 2:48 PM

What about using `IndexIgnore *` to prevent sitewide directory listings..?

Jeff Starr 2008/11/09 9:42 AM • POST AUTHOR

@Bleyder: my pleasure! Glad to be a part of the community :)

@August: good question! the main difference between using `IndexIgnore *` and `Options -Indexes` is that `IndexIgnore` is intended to hide specific files/types for enabled default directory listings, while `Options -Indexes` is used to disable directory listings entirely.

1 2 3 … 5 • Newer Comments »

Comments are closed for this post. Something to add? Let me know.