Mapping URLs to Filesystem Locations

This document explains how the Apache HTTP Server uses the URL of a request to determine the filesystem location from which to serve a file.

Related Modules and Directives

Related Modules	Related Directives
mod_actions	Alias
mod_alias	AliasMatch
mod_autoindex	CheckSpelling
mod_dir	DirectoryIndex
mod_imagemap	DocumentRoot
mod_negotiation	ErrorDocument
mod_proxy	Options
mod_rewrite	ProxyPass
mod_speling	ProxyPassReverse
mod_userdir	ProxyPassReverseCookieDomain
mod_vhost_alias	ProxyPassReverseCookiePath
	Redirect
	RedirectMatch
	RewriteCond
	RewriteRule
	ScriptAlias
	ScriptAliasMatch
	UserDir

DocumentRoot

In deciding what file to serve for a given request, httpd's default behavior is to take the URL-Path for the request (the part of the URL following the hostname and port) and add it to the end of the DocumentRoot specified in your configuration files. Therefore, the files and directories underneath the DocumentRoot make up the basic document tree which will be visible from the web.

For example, if DocumentRoot were set to `/var/www/html` then a request for `http://www.example.com/fish/guppies.html` would result in the file `/var/www/html/fish/guppies.html` being served to the requesting client.

If a directory is requested (i.e. a path ending with `/`), the file served from that directory is defined by the DirectoryIndex directive. For example, if DocumentRoot were set as above, and you were to set:

```
DirectoryIndex index.html index.php
```

Then a request for `http://www.example.com/fish/` will cause httpd to attempt to serve the file `/var/www/html/fish/index.html`. In the event that that file does not exist, it will next attempt to serve the file `/var/www/html/fish/index.php`.

If neither of these files existed, the next step is to attempt to provide a directory index, if mod_autoindex is loaded and configured to permit that.

httpd is also capable of Virtual Hosting, where the server receives requests for more than one host. In this case, a different DocumentRoot can be specified for each virtual host, or alternatively, the directives provided by the module

mod_vhost_alias can be used to dynamically determine the appropriate place from which to serve content based on the requested IP address or hostname.

The DocumentRoot directive is set in your main server configuration file (httpd.conf) and, possibly, once per additional Virtual Host you create.

Files Outside the DocumentRoot

There are frequently circumstances where it is necessary to allow web access to parts of the filesystem that are not strictly underneath the DocumentRoot. httpd offers several different ways to accomplish this. On Unix systems, symbolic links can bring other parts of the filesystem under the DocumentRoot. For security reasons, httpd will follow symbolic links only if the Options setting for the relevant directory includes FollowSymLinks or SymLinksIfOwnerMatch.

Alternatively, the Alias directive will map any part of the filesystem into the web space. For example, with

```
Alias "/docs" "/var/web"
```

the URL http://www.example.com/docs/dir/file.html will be served from /var/web/dir/file.html. The ScriptAlias directive works the same way, with the additional effect that all content located at the target path is treated as CGI scripts.

For situations where you require additional flexibility, you can use the AliasMatch and ScriptAliasMatch directives to do powerful regular expression based matching and substitution. For example,

```
ScriptAliasMatch "^/~([a-zA-Z0-9]+)/cgi-bin/(.+)"    "/h
```

will map a request to http://example.com/~user/cgi-bin/script.cgi to the path /home/user/cgi-bin/script.cgi and will treat the resulting file as a CGI script.

User Directories

Traditionally on Unix systems, the home directory of a particular *user* can be referred to as ~user/. The module mod_userdir extends this idea to the web by allowing files under each user's home directory to be accessed using URLs such as the following.

```
http://www.example.com/~user/file.html
```

For security reasons, it is inappropriate to give direct access to a user's home directory from the web. Therefore, the UserDir directive specifies a directory underneath the user's home directory where web files are located. Using the default setting of Userdir public_html, the above URL maps to a file at a directory like /home/user/public_html/file.html where /home/user/ is the user's home directory as specified in /etc/passwd.

There are also several other forms of the Userdir directive which you can use on systems where /etc/passwd does not contain the location of the home directory.

Some people find the "~" symbol (which is often encoded on the web as %7e) to be awkward and prefer to use an alternate string to represent user directories. This functionality is not supported by mod_userdir. However, if users' home directories are structured in a regular way, then it is possible to use the AliasMatch directive to achieve the desired effect. For example, to make http://www.example.com/upages/user/file.html map to

/home/user/public_html/file.html, use the following AliasMatch directive:

```
AliasMatch "^/upages/([a-zA-Z0-9]+)(/(.*))?$"    "/home/
```

URL Redirection

The configuration directives discussed in the above sections tell httpd to get content from a specific place in the filesystem and return it to the client. Sometimes, it is desirable instead to inform the client that the requested content is located at a different URL, and instruct the client to make a new request with the new URL. This is called *redirection* and is implemented by the Redirect directive. For example, if the contents of the directory /foo/ under the DocumentRoot are moved to the new directory /bar/, you can instruct clients to request the content at the new location as follows:

```
Redirect permanent "/foo/"    "http://www.example.com/ba
```

This will redirect any URL-Path starting in /foo/ to the same URL path on the www.example.com server with /bar/ substituted for /foo/. You can redirect clients to any server, not only the origin server.

httpd also provides a RedirectMatch directive for more complicated rewriting problems. For example, to redirect requests for the site home page to a different site, but leave all other requests alone, use the following configuration:

```
RedirectMatch permanent "^/$"     "http://www.example.co
```

Alternatively, to temporarily redirect all pages on one site to a particular page on another site, use the following:

```
RedirectMatch temp ".*"  "http://othersite.example.com/
```

Reverse Proxy

httpd also allows you to bring remote documents into the URL space of the local server. This technique is called *reverse proxying* because the web server acts like a proxy server by fetching the documents from a remote server and returning them to the client. It is different from normal (forward) proxying because, to the client, it appears the documents originate at the reverse proxy server.

In the following example, when clients request documents under the /foo/ directory, the server fetches those documents from the /bar/ directory on internal.example.com and returns them to the client as if they were from the local server.

```
ProxyPass         "/foo/" "http://internal.example.com/b
ProxyPassReverse  "/foo/" "http://internal.example.com/b
ProxyPassReverseCookieDomain internal.example.com publi
ProxyPassReverseCookiePath "/foo/" "/bar/"
```

The ProxyPass configures the server to fetch the appropriate documents, while the ProxyPassReverse directive rewrites redirects originating at internal.example.com so that they target the appropriate directory on the local server. Similarly, the ProxyPassReverseCookieDomain and ProxyPassReverseCookiePath rewrite cookies set by the backend server.

It is important to note, however, that links inside the documents will not be rewritten. So any absolute links on internal.example.com will result in the client breaking out of the proxy server and requesting directly from

internal.example.com. You can modify these links (and other content) in a page as it is being served to the client using mod_substitute.

```
Substitute s/internal\.example\.com/www.example.com/i
```

For more sophisticated rewriting of links in HTML and XHTML, the mod_proxy_html module is also available. It allows you to create maps of URLs that need to be rewritten, so that complex proxying scenarios can be handled.

Rewriting Engine

When even more powerful substitution is required, the rewriting engine provided by mod_rewrite can be useful. The directives provided by this module can use characteristics of the request such as browser type or source IP address in deciding from where to serve content. In addition, mod_rewrite can use external database files or programs to determine how to handle a request. The rewriting engine is capable of performing all three types of mappings discussed above: internal redirects (aliases), external redirects, and proxying. Many practical examples employing mod_rewrite are discussed in the detailed mod_rewrite documentation.

File Not Found

Inevitably, URLs will be requested for which no matching file can be found in the filesystem. This can happen for several reasons. In some cases, it can be a result of moving documents from one location to another. In this case, it is best to use URL redirection to inform clients of the new location of the resource. In this way, you can assure that old bookmarks and links will continue to work, even though the resource is at a new location.

Another common cause of "File Not Found" errors is accidental mistyping of URLs, either directly in the browser, or in HTML links. httpd provides the module mod_speling (sic) to help with this problem. When this module is activated, it will intercept "File Not Found" errors and look for a resource with a similar filename. If one such file is found, mod_speling will send an HTTP redirect to the client informing it of the correct location. If several "close" files are found, a list of available alternatives will be presented to the client.

An especially useful feature of mod_speling, is that it will compare filenames without respect to case. This can help systems where users are unaware of the case-sensitive nature of URLs and the unix filesystem. But using mod_speling for anything more than the occasional URL correction can place additional load on the server, since each "incorrect" request is followed by a URL redirection and a new request from the client.

mod_dir provides FallbackResource, which can be used to map virtual URIs to a real resource, which then serves them. This is a very useful replacement for mod_rewrite when implementing a 'front controller'

If all attempts to locate the content fail, httpd returns an error page with HTTP status code 404 (file not found). The appearance of this page is controlled with the ErrorDocument directive and can be customized in a flexible manner as discussed in the Custom error responses document.

Other URL Mapping Modules

Other modules available for URL mapping include:

- mod_actions - Maps a request to a CGI script based on the request method, or resource MIME type.
- mod_dir - Provides basic mapping of a trailing slash into an index file such as index.html.

- mod_imagemap - Maps a request to a URL based on where a user clicks on an image embedded in a HTML document.
- mod_negotiation - Selects an appropriate document based on client preferences such as language or content compression.