Web Publishing - Redirects

Web Publishing - Redirects

Working with files on Athena

The Following instructions assume you know how to access and work within your Athena locker. Detailed instructions are available for Using SecureFX on Windows and for using Fetch on macOS.

Summary

When you rename or move a page that results in a URL change, a redirect is required. Otherwise, users following links to the old URL will get a `404 (Page not found)` error.

Since you may have users who have bookmarked specific pages on your site or there may be other web sites linking to your pages, it's a good idea to set up a redirect that automatically re-routes visitors instantly to the correct location.

The Apache `mod_alias` module is installed on `web.mit.edu` and provides `Redirect` and `RedirectMatch` directives that can be configured for any directory and locker on `web.mit.edu`. The advantage to using these two directives is that setting a `301 (redirect permanent)` code will allow search engines to pick up on the new page and update its indexes to no longer reference the old location. It also does not require the old page to still exists or to be loaded.

Don't use meta-refresh
The old `meta-refresh` tag used in page headers is no longer recommended. This tag used to be embedded in the header of HTML pages, but it required the old pages to remain around, and creates more overhead for the visitor as the old page needs to be loaded before the browser is redirected to the new one. If you are still using `meta-refresh`, we recommend that you use `Redirect` or `RedirectMatch` instead.

`mod_alias`, `Redirect`, and `RedirectMatch`

`mod_alias` provides instantaneous redirection for pages that have moved. By providing the appropriate HTTP response codes, using these directives vs. the meta-refresh approach will improve your search engine listings (internally and externally), and reduce the work each of your visitors' browsers needs to do.

By customizing your `.htaccess.mit` file (in this case usually one at the root of your website, in the same directory as your home pages) you can indicate whether certain content has moved temporarily, permanently or has been removed altogether.

Redirecting single pages or directories

The below examples show how to redirect single locations to locations at a new site. Note that the original page location referenced in your redirect should begin with a `/` and the name of your locker on {[web.mit.edu}}. So if your old page URL was `http://web.mit.edu/old/about.html` then your redirect should reference `/old/about.html`.

The redirect entry in the `.htaccess.mit` file will have four components in the specified order:

The `Redirect` directive	Type of redirect	Old location	New location

Here are several examples of how to use `Redirect` to redirect specific locations to their new ones.

```
# Two ways to create a permanent redirect sending visitors
# going to web.mit.edu/old/page.html to newsite.mit.edu/page.html
Redirect permanent /old/page.html http://newsite.mit.edu/page.html
Redirect 301 /old/page.html http://newsite.mit.edu/page.html

# Two ways to create a temporary redirect sending visitors
# going to web.mit.edu/old/page.html to newsite.mit.edu/page.html
Redirect temp /old/page.html http://newsite.mit.edu/page.html
Redirect 302 /old/page.html http://newsite.mit.edu/page.html

# Two ways to create a "page removed" redirect sending visitors
# going to web.mit.edu/one to newsite.mit.edu/ and letting search engines
# know that an equivalent page does not exist on the new site
Redirect gone /old/page.html http://newsite.mit.edu/
Redirect 410 /old/page.html http://newsite.mit.edu/
```

Other HTTP result codes are supported as well, but the ebove three are the most commonly used ones for site redirects. Note that `Redirect` will do a prefix match, so if you do not specify each page fully, this can lead to unpredictable results. For example, this redirect:

```
# Create a permanent redirect for locations that begin with
# web.mit.edu/one
Redirect 301 /old/ http://newsite.mit.edu/
```

will only work if there is an equivalent page for every old page on the new site. It will redirect `web.mit.edu/old/index.html` to `newsite.mit.edu/index.html`, `web.mit.edu/old/about.html` to `newsite.mit.edu/about.html`, and so on.

Redirecting patterns and matching subsets of pages

The `RedirectMatch` directive can be used to redirect certain pages and locations based on matched patterns and wildcards. You can redirect many pages at once using pattern matching, or target a specific page. You can also reference portions of the original page name in the destination for your redirect. For example:

```
# Permanently redirect every page in the "web" directory at web.mit.edu/one
# to its matching page on the new site in the "two" directory. For example,
# the old page web.mit.edu/one/web/about.html would be redirected to
# example.com/two/about.html
RedirectMatch 301 ^/one/web/(.*)$ http://example.com/two/$1
```

The `RedirectMatch` directive is very powerful and can be used for complex pattern matching and redirects. For more information, see: Apache Group's documentation on RedirectMatch.

Perl Regular Expressions not supported
The `mod_alias` that runs on `web.mit.edu` is an older version that doesn't support Perl-Compatible Regular Expressions (PCREs), only POSIX Extended Regular Expressions (EREs). This means it will only provide the level of regular expression support typically available in the `egrep` command.