Moving things around with mod_rewrite (very rough)

By Cal Henderson

From time to time, you might want to change the way your weblog handles archives, or you might want to move it from one folder to another. Perhaps even move to a new domain. One of the big problems with moving is link-rot; Anyone linking to your posts will end up with broken links.

Fortunately, there's something you can do about this. If your webserver is Apache and has mod_rewrite installed (you'll need to ask your web host about this) then you can simply redirect old URLs to new URLs.

Let's start simple. Suppose your weblog is at www.domain.com/blog and your archives are in a folder called "archive". You realise that the term "blog" has become unfashionable and you really want to move it to www.domain.com/weblog so that people wont laugh at you.

All you'll need is a single rewriting rule. Rewriting rules live in a file called ".htaccess" in the folder in question, on your webserver. You may not be able to create a file called ".htaccess" if you're using windows - don't worry, just upload the file with a different name, then rename it over FTP. You might already have a ".htaccess" file, since it's used to configure lots of different server options.

Before using any rewrite rules, you need to switch on rewriting with the following line in your ".htaccess" file:

```
RewriteEngine on
```

After this line, you can put as many rewriting rules as you like. The one you'll need to switch from "blog" to "weblog" is this:

```
RewriteRule ^blog(/.*)?$ weblog$1 [R=permanent]
```

Each line is made up of four parts. The first part is the command, which will always be "RewriteRule" (or "RewriteCond", when we get onto conditions a bit later on). The second part specifies the path to match, and the third part specifies the path to replace it with. The fourth and final part specifies any special options for the rule.

In the example above, we match lines that start ("^" is the starting indicator) with the word "blog", optionally followed by ("()?" means an optional part) a slash ("/") and any other characters (".*" means any sequence of characters). The dollar sign ("$") at the end means "match all the way to the end of the line". These symbols are what are called regular expressions. Don't fret, you don't really need to understand how they work unless you need to write more complex rewriting rules.

The replacement clause then replaces the URL we had, with the URL "weblog", followed by anything that was matched inside the brackets of the matching clause. To see what's actually happening, we can look at some examples of URLs with their rewritten results:

```
http://www.domain.com/blog
http://www.domain.com/weblog

http://www.domain.com/blog/
http://www.domain.com/weblog/

http://www.domain.com/blog/hello.htm
http://www.domain.com/weblog/hello.htm
```

```
http://www.domain.com/blog/archive/hello.htm
http://www.domain.com/weblog/archive/hello.htm
```

That was pretty easy, but what if you're moving from one domain to another? All we need is a small change because rewrite rules allow us to specify a new domain in the replacement string. If we move a weblog from www.domain.com to www.newdomain.com then we can put a .htaccess file on the old server with the following rewrite line in:

```
RewriteEngine on
RewriteRule ^(/.*)?$ http://www.newdomain.com$1 [R=permanent]
```

Or we can combine the two, moving folders and domains:

```
RewriteEngine on
RewriteRule ^blog(/.*)?$ http://www.newdomain.com/weblog$1 [R=permanent]
```

TODO: something about changing archive url formats. talk to tom about this?

Note: This article is a work in progress and doesn't yet show you what mod_rewrite is really useful for. Everything above could be accomplished with the simpler Redirect directive.

(end of article)