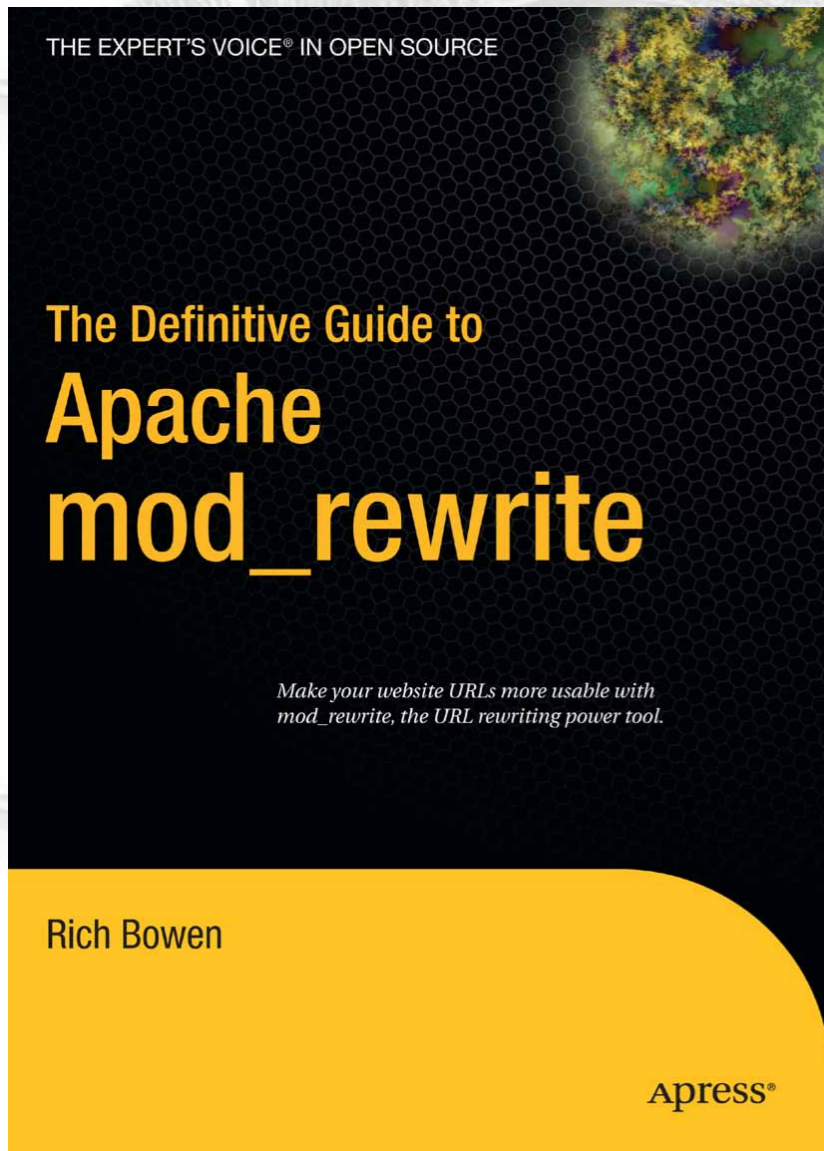


mod_rewrite

Introduction to mod_rewrite
Rich Bowen, Director of Development,
ClearMyRecord.com
rbowen@apache.org

<http://people.apache.org/~rbowen/>

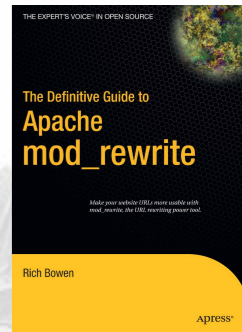


clearMYrecord.com



Outline

- Regex basics
- RewriteRule
- RewriteCond
- RewriteMap
- The evils of .htaccess files
- Recipes





mod_rewrite is not magic

- Fear, more than complexity, makes mod_rewrite difficult





Although, it is complex

“The great thing about mod_rewrite is it gives you all the configurability and flexibility of Sendmail. The downside to mod_rewrite is that it gives you all the configurability and flexibility of Sendmail.”

-- Brian Behlendorf

And let's not forget voodoo!

“ Despite the tons of
examples and docs,
mod_rewrite is voodoo.
Damned cool voodoo,
but still voodoo. ”
-- Brian Moore





Line noise

"Regular expressions
are just line noise.
I hate them!"
(Heard 20 times per
day on IRC)



When you hear it often
enough, you start to
believe it

Now that that's out of the way

- Regular expressions are not magic
- They are an algebraic expression of text

patterns

- Once you get over the mysticism, it can still be hard, but it's no longer mysterious

$$y_2 - y_1 = mx_2 + b - (mx_1 + b)$$

$$y_2 - y_1 = mx_2 + b - mx_1 - b$$

$$y_2 - y_1 = mx_2 - mx_1$$

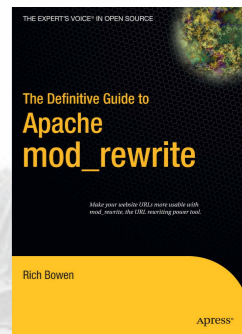
$$y_2 - y_1 = m(x_2 - x_1)$$

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$



Vocabulary

- We're going to start with a very small vocabulary, and work up from there
- Most of the time, this vocabulary is all that you'll need

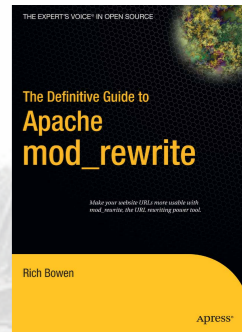




-
- . matches any character
- "a.b" matches acb, axb, a@b, and so on
- It also matches Decalb and Marbelized



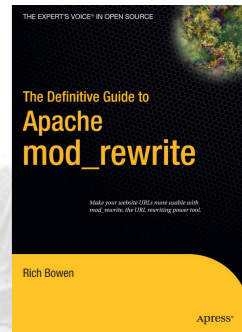
Repetition



- + means that something needs to appear one or more times: 'a+' matches 'a', 'aa', 'aaa', and 'Mondrian'
- * matches zero or more. 'a*' matches 'a', 'aa', and the empty string ('')
- ? means that the match is optional. 'colou?r' matches 'color' and 'colour'



Anchors



- `^` means "starts with". `^a` matches `'alpha'` and `'arnold'`
- `$` means "ends with". `a$` matches `'alpha'` and `'stella'`
- `^^$` matches an empty string
- `^^` matches every string (every string has a start)



() – Grouping

- () allows you to group several characters into one thingy, and apply other modifiers to it.
- “(ab)+” matches ababababababab



() – Backreferences

- () allows you to capture a match so that you can use it later (called a backreference)
- It might be called \$1 or %1 depending on the context
- The second match is called \$2 (or %2) and so on



[]

- [] defines a “character class”
- [abc] matches a or b or c
- “c[uoa]t” matches cut, cot, or cat
- It also matches cote
- It does not match coat



NOT

- In mod_rewrite regular expressions, ! negates any match
- In a character class, ^ negates the character class
- [^ab] matches any character except for a or b.

So, what does this have to do with Apache?

- `mod_rewrite` lets you match URLs (or other things) and transform the target of the URL based on that match.

```
RewriteEngine On  
RewriteRule (.*)\.cfm$ $1.php [PT]
```


RewriteEngine

- “RewriteEngine On” enables the mod_rewrite rewriting engine
- No rewrite rules will be performed unless this is enabled in the active scope





RewriteLog

```
RewriteLog /www/logs/rewrite_log  
RewriteLogLevel 9
```

You should turn on the RewriteLog before you do any troubleshooting.

RewriteRule

RewriteRule pattern target [flags]

- The pattern part is the regular expression that you want to look for in the URL.
- If they try to go HERE send them HERE instead.
- The behavior can be further modified by the use of one or more flags



Example 1

- SEO - "Search Engine Optimization"
- Frequently based on misconceptions about how search engines work
- Typical strategy is to make "clean URLs" -
Avoid `?argument=value&xyz=123`



URL beautification

- A URL looks like:

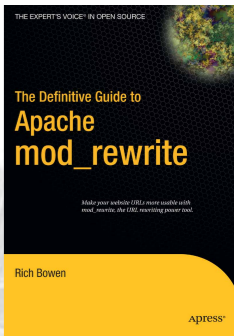
```
http://example.com/cgi-bin/book.cgi?author=bowen&topic=apache
```

We would prefer that it looked like

```
http://example.com/book/bowen/apache
```

It's easier to type, and easier to remember

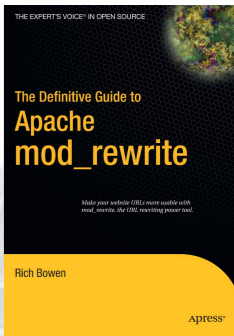
Example 1, cont'd



```
RewriteRule ^/book/(.*)/(.*) \  
/cgi-bin/book.cgi?topic=$1&author=$2 [PT]
```

- User does not notice that the transformation has been made
- Used \$1 and \$2 to capture what was requested
- Slight oversimplification. Should probably use $([^\wedge/]+)$ instead.

Example 1, cont'd

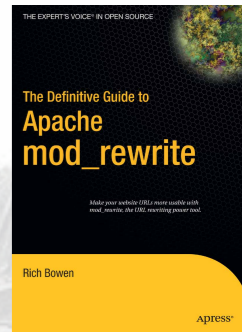


```
RewriteRule ^/book/([^/]+)/([^/]+) \
/cgi-bin/book.cgi?topic=$1&author=$2 [PT]
```

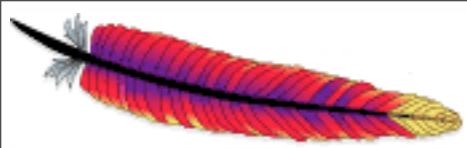
- Should probably use `([^/]+)` instead of `(.*)`
- `(.*)` is frequently used when something else would be faster, or at least more correct.



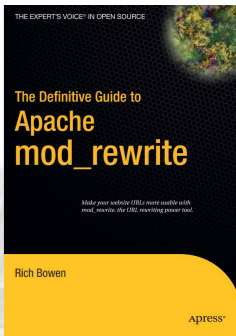
Flags



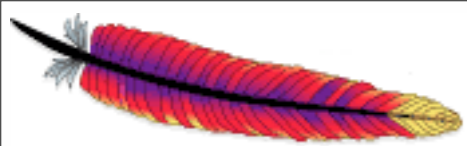
- Flags can modify the behavior of a RewriteRule
- I used a flag in the example, and didn't tell you what it meant



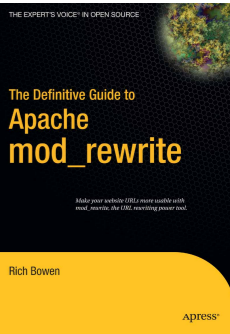
Default



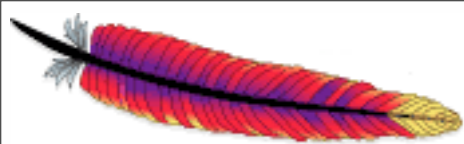
- Default is to treat the rewrite target as a file path
- If the target starts in `http://` or `https://` then it is treated as a URL, and a [R] is assumed (Redirect)
- In a `.htaccess` file, or in `<Directory>` scope, the file path is assumed to be relative to that scope



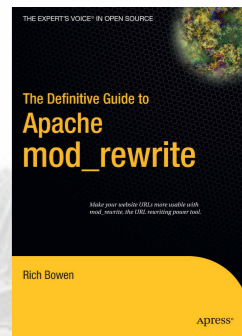
RewriteRule flags



- [Flag] appears at end of RewriteRule
- More than one flag separated by commas - eg [R,L,NE] (no spaces)
- There's *lots* of flags



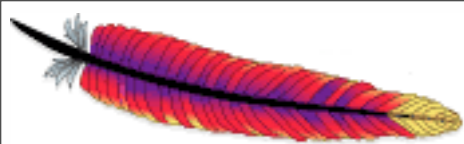
Cookie



- [CO=NAME:Value:Domain[:lifetime[:path]]
- Long form [cookie=...]
- Sets a cookie

```
RewriteRule ^/index.html - [CO=frontdoor:yes:.example.com]
```

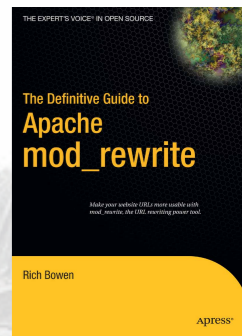
In this case, the default values for path ("/") and lifetime ("session") are assumed.



Env

- [E=var:val]
- Long form [env=...]
- Sets environment variable
- Note that most of the time, SetEnvIf works just fine

```
RewriteRule \.(gif|jpg|png)$ - [env=image:1]  
CustomLog /var/log/access_log \  
    combined env=!image
```





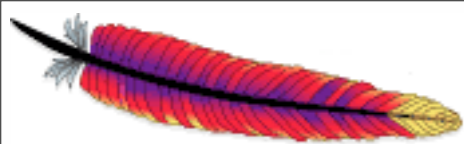
Forbidden

- [F] or [Forbidden] forces a 403 Forbidden response
- Consider mod_security instead for pattern-based URL blocking

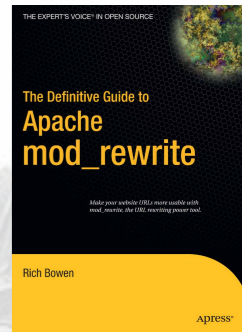
```
RewriteEngine On  
RewriteRule (cmd|root)\.exe - [F]
```

You could use this in conjunction with [E] to avoid logging that stuff

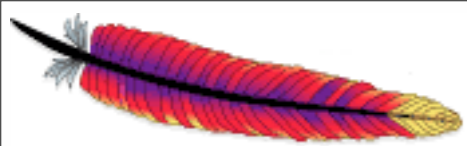
```
RewriteRule (cmd|root)\.exe - [F,E=dontlog:1]  
CustomLog /var/log/apache/access_log combined \  
env=!dontlog
```



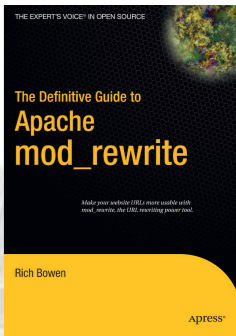
Handler



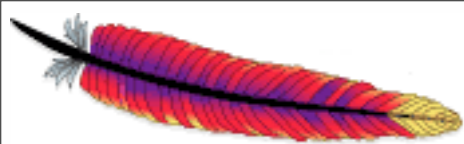
- [H=application/x-httpd-php]
- Forces the use of a particular handler to handle the resulting URL
- Can often be replaced with using [PT] but is quite a bit faster
- Available in Apache 2.2



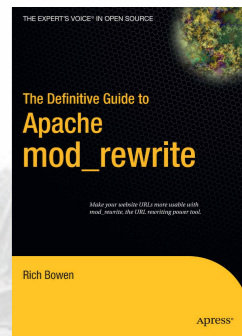
Last



- [L] indicates that you've reached the end of the current ruleset
- Any rules following this will be considered as a completely new ruleset
- It's a good idea to use it, even when it would otherwise be default behavior. It helps make rulesets more readable.

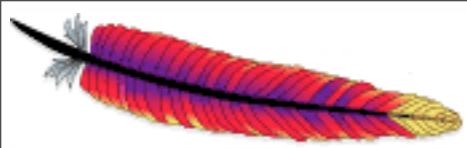


Proxy

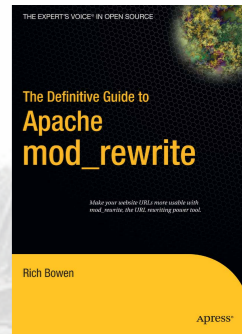


- [P] rules are served through a proxy subrequest
- mod_proxy must be installed for this flag to work

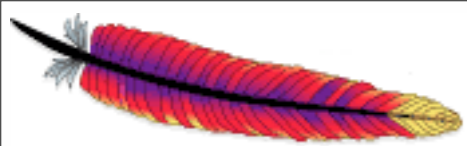
```
RewriteEngine On  
RewriteRule (.*)\.(jpg|gif|png) \  
http://images.example.com$1.$2 [P]
```

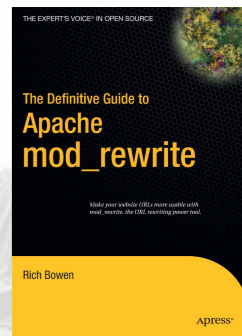
Passthrough



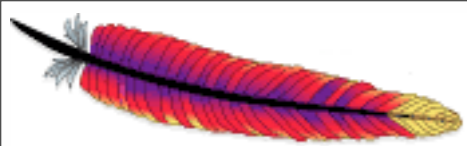
- [PT] or [passthrough]
- Hands it back to the URL mapping phase
- Treat this as though this was the original request



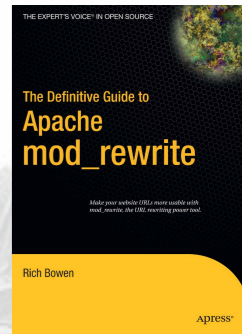
QSAappend



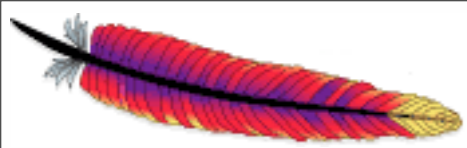
- [QSA] or [qsappend] appends to the query string, rather than replacing it.



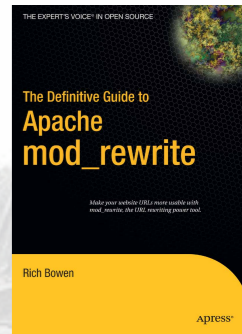
Redirect



- [R] or [redirect] forces a 302 Redirect
- Note that in this case, the user will see the new URL in their browser
- This is the default behavior when the target starts with http:// or https://



Redirect



- Can designate a different redirect status code with [R=305]
- Can even do [R=404] if you want



Skip

- [S=n] or [skip=n] skips the next n RewriteRules
- Can be used for negation of a block of rules
 - as a sort of inverse RewriteCond

```
# Don't run these rules if its an image  
RewriteRule ^/images - [S=4]
```



RewriteCond and [S]

- Problem: I want this RewriteCond to apply to all of the next 4 rules:

```
# Block One
```

```
RewriteCond %{HTTP_HOST} example.com
```

```
RewriteRule \.gif$ /something1.html
```

```
RewriteRule \.jpg$ /something2.html
```

```
RewriteRule \.ico$ /something3.html
```

```
RewriteRule \.png$ /something4.html
```

```
# Block Two
```

```
RewriteRule ^ - http://something.else.com/ [R]
```



RewriteCond and [S]

- DOESN'T WORK

```
# Block One
```

```
RewriteCond %{HTTP_HOST} example.com
```

```
RewriteRule \.gif$ /something1.html
```

```
RewriteRule \.jpg$ /something2.html
```

```
RewriteRule \.css$ /something3.html
```

```
RewriteRule \.png$ /something4.html
```

```
# Block Two
```

```
RewriteRule ^ - http://something.else.com/ [R]
```



RewriteCond and [S]

- Solution: Use an [S] flag as a GoTo statement

```
# Block One
```

```
RewriteCond !%{HTTP_HOST} example.com
```

```
RewriteRule ^ - [S=4]
```

```
RewriteRule \.gif$ /something1.html
```

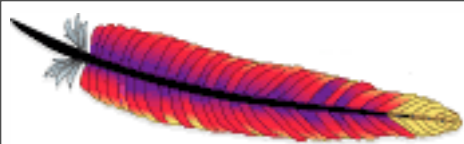
```
RewriteRule \.jpg$ /something2.html
```

```
RewriteRule \.ico$ /something3.html
```

```
RewriteRule \.png$ /something4.html
```

```
# Block Two
```

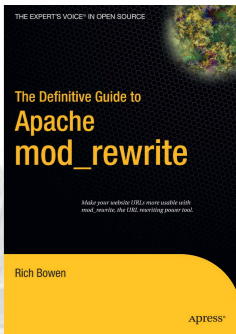
```
RewriteRule ^ - http://something.else.com/ [R]
```

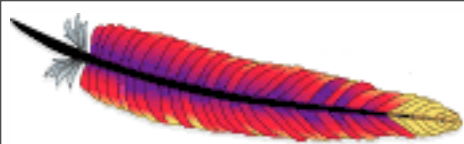



Type

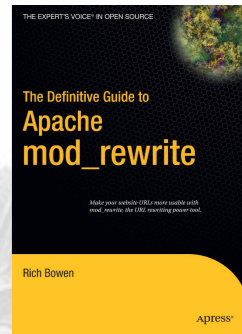
- [T=text/html]
- Forces the Mime type on the resulting URL
- Good to ensure that file-path redirects are handled correctly

```
RewriteRule ^(.+\.php)$ $1 [T=application/x-httpd-php-source]
```

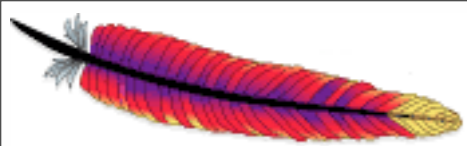




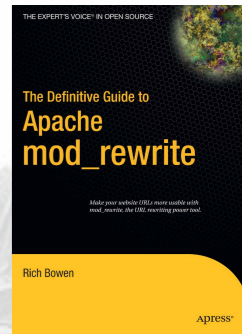
Others



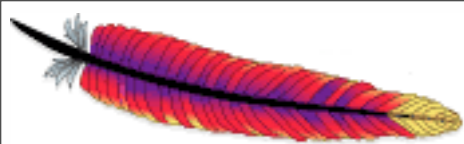
- [NC] or [nocase] makes the RewriteRule case insensitive
- [NE] or [NoEscape] - Don't URL-escape the results
- [NS] - Don't run on subrequests



Infrequently used



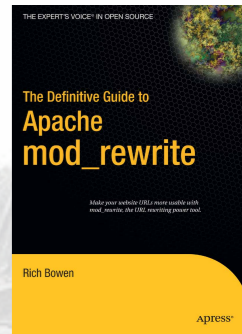
- [C] or [Chain] - Rules are considered as a whole. If one fails, the entire chain is abandoned
- [N] or [Next] - Start over at the top. Useful for rules that run in a while loop

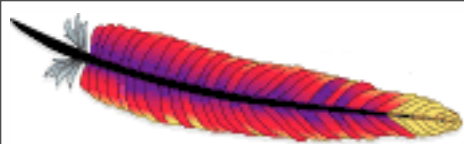


RewriteCond

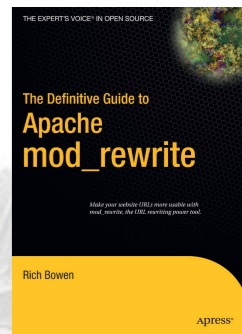
- Causes a rewrite to be conditional
- Can check the value of any variable and make the rewrite conditional on that.

```
RewriteCond TestString Pattern [Flags]
```

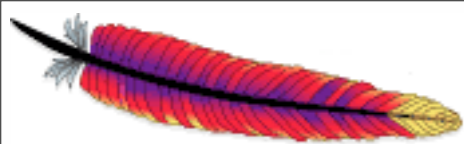




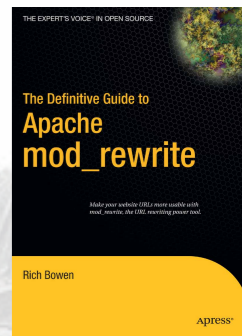
RewriteCond



- Test string can be Env vars, headers, or a literal expression
- Backreferences become %1, %2, etc

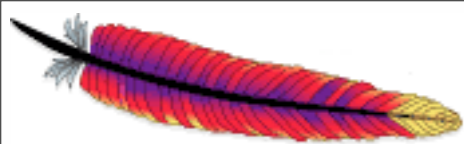


Example - Looping

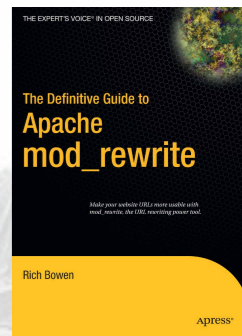


- Looping occurs when the target of a rewrite rule matches the pattern
- This results in an infinite loop of rewrites

```
RewriteRule ^/example /example.html [PT]
```

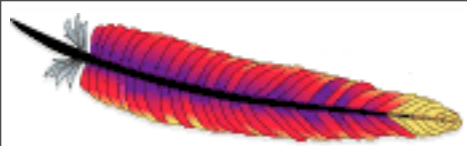


Example - Looping

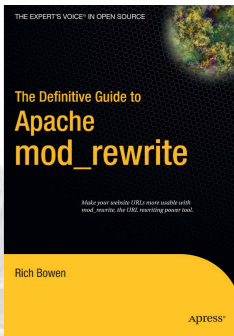


- Looping occurs when the target of a rewrite rule matches the pattern
- This results in an infinite loop of rewrites

```
RewriteRule ^/example /example.html [PT]
```



Example - Looping



- Solution: use RewriteCond to exclude that condition.

```
RewriteCond %{REQUEST_URI} \  
    !^/example.html  
RewriteRule ^/example /example.html [PT]
```


Conditional rewrites

- Rewrites conditional on some arbitrary thingy
- Only first Rule is dependent

```
RewriteEngine on
RewriteCond %{TIME_HOUR}%{TIME_MIN} >0700
RewriteCond %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule ^page\.html$ page.day.html
RewriteRule ^page\.html$ page.night.html
```

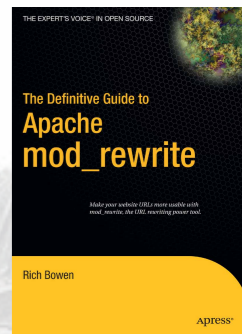
Conditional rewrites

- At 08:30 ...

```
RewriteEngine on
RewriteCond %{TIME_HOUR}%{TIME_MIN} >0700
RewriteCond %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule ^page\.html$ page.day.html
RewriteRule ^page\.html$ page.night.html
```



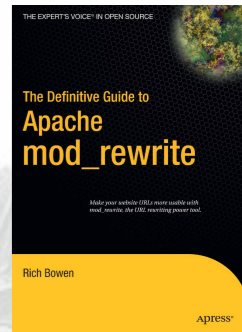
Conditional rewrites



- Could also use an [L] flag here

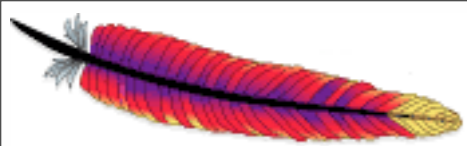
```
RewriteEngine on
RewriteCond %{TIME_HOUR}%{TIME_MIN} >0700
RewriteCond %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule ^page\.html$ page.day.html [L]
RewriteRule ^page\.html$ page.night.html
```

Conditional rewrites

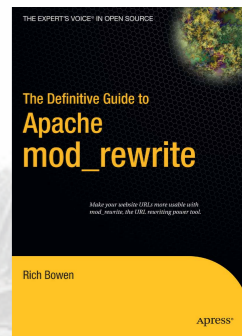


- At 21:49 ...

```
RewriteEngine on
RewriteCond %{TIME_HOUR}%{TIME_MIN} >0700
RewriteCond %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule ^page\.html$ page.day.html
RewriteRule ^page\.html$ page.night.html
```

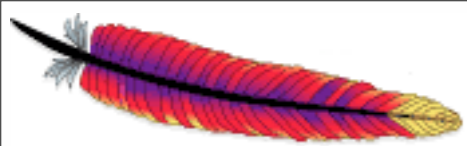


SSL Rewrites

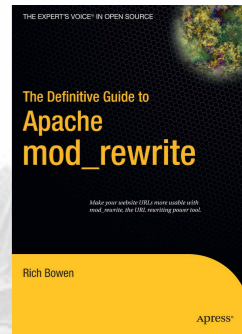


- Redirect requests to https:// if the request was for http

```
RewriteCond %{HTTPS} !on  
RewriteRule (.*) https://%{HTTP_HOST}/$1 [R]
```



RewriteMap



- Call an external program, or map file, to perform the rewrite
- Useful for very complex rewrites, or perhaps ones that rely on something outside of Apache



RewriteMap - file

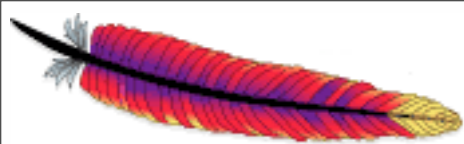
- File of one-to-one relationships

```
RewriteMap dogmap txt:/www/conf/dogmap.txt  
RewriteRule ^/dog/(.*) ${dogmap:$1} [NE,PT]
```

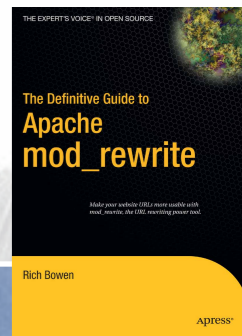
Where dogmap.txt contains:

```
doberman    /dogs.php?breed=278  
poodle      /dogs.php?breed=78  
collie      /dogs.php?breed=98  
terrier     /dogs.php?breed=148  
mutt        /dogs.php?breed=2  
alsatian    /dogs.php?breed=113
```

Requests for <http://example.com/dog/something> now get redirected to the page for that breed. [NE] ensures that the ? doesn't get escaped.



dbm



```
RewriteMap asbury \  
dbm:/usr/local/apache/conf/aliases.map
```

- Convert a one-to-one text mapping to a dbm file
- `httxt2dbm` utility (2.0)

RewriteMap - program

- Call an external program to do the rewrite
- Perl is a common choice here, due to its skill at handling text.

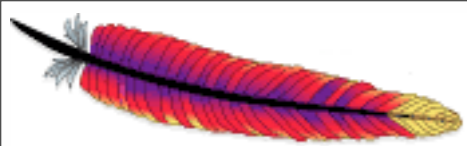
```
RewriteMap dash2score \  
    prg:/usr/local/apache/conf/dash2score.pl  
RewriteEngine On  
RewriteRule (*.*) ${dash2score:$1} [PT]
```



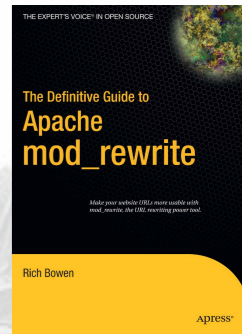
dash2score.pl

```
#!/usr/bin/perl
$| = 1; # Turn off buffering
while (<STDIN>) {
    s/-/_/g; # Replace - with _ globally
    print $_;
}
```

- * Turn off buffering
- * Script runs for lifetime of Apache process
- * Blocking - use RewriteLock

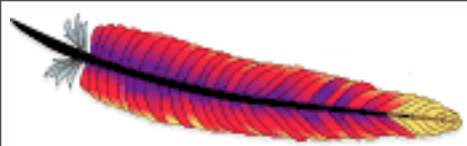


SQL (in 2.3-HEAD)

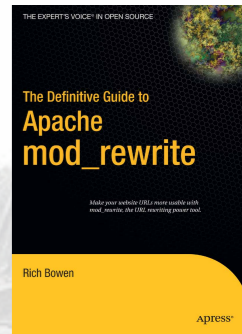


```
RewriteMap myquery "fastdbd:SELECT  
destination FROM rewrite WHERE source = %s"
```

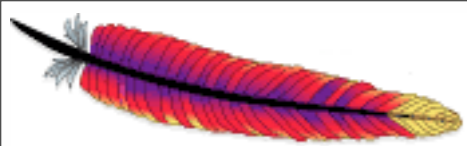
- Have a SQL statement in the RewriteMap directive which returns the mapping
- 'fastdbd' caches, 'dbd' doesn't



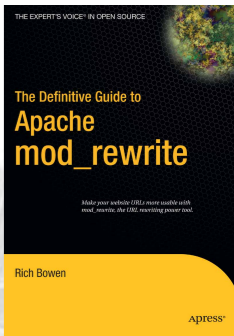
.htaccess files



- .htaccess files introduce many additional complexities
- However, a lot of people have no choice
- So ...



.htaccess files



- In .htaccess files, or `<Directory>` scope, everything is assumed to be relative to that current scope
- So, that scope is removed from the RewriteRule
- `^/index.html` in `httpd.conf` becomes `^index.html` in a .htaccess file or `<Directory>` scope

.htaccess files

```
# In httpd.conf
```

```
RewriteRule ^/images/(.+)\.jpg /images/$1.png
```

```
# In .htaccess in root dir
```

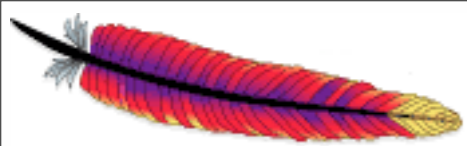
```
RewriteBase /
```

```
RewriteRule ^images/(.+)\.jpg images/$1.png
```

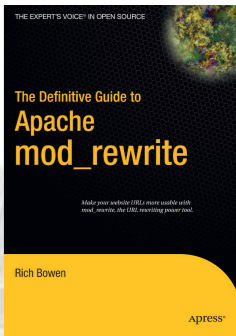
```
# In .htaccess in images/
```

```
RewriteBase /images/
```

```
RewriteRule ^(.+)\.jpg $1.png
```



.htaccess files



- RewriteLog is particularly useful when trying to get .htaccess file RewriteRules working.
- However, you can't turn on RewriteLog in a .htaccess file, and presumably you're using .htaccess files because you don't have access to the main server config.
- It's a good idea to set up a test server and test there with RewriteLog enabled

Redirect to one thing

```
RewriteEngine On  
RewriteCond %{REQUEST_FILENAME} !-d  
RewriteCond %{REQUEST_FILENAME} !-f  
RewriteCond %{REQUEST_URI} !handler.php  
RewriteRule (.*) /handler.php?$1 [PT,L,NE,QSA]
```

All requests are sent to handler.php
The request is passed as a QUERY_STRING
argument to handler.php so that it knows what
was requested.



Query String

- RewriteRule doesn't have access to the Query String

```
# Rewrite based on query string
RewriteCond %{QUERY_STRING} \
    \bfoo=(.*?)\b
RewriteRule /something /somewhere/%1 [QSA]
```

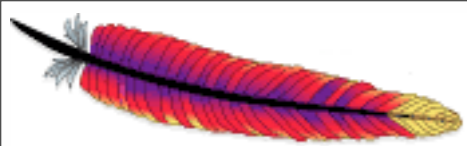


Query String

- "word" boundaries

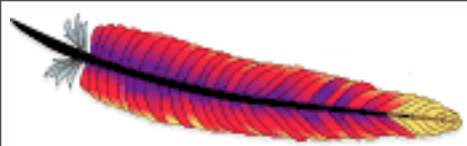
```
# Rewrite based on query string  
RewriteCond %{QUERY_STRING} \  
  \bfoo=(.*?)\b
```

```
RewriteRule /something /somewhere/%1 [QSA]
```



Virtual Hosts

- Rewrite a request to a directory based on the requested hostname.



RewriteEngine On

RewriteCond %{HTTP_HOST} (.*)\.example\.com [NC]

RewriteRule (.*) /home/%1/www\$1

- The hostname ends up in %1
- The requested path is in \$1 – includes leading slash
- Will probably have to do special things for handlers (like .php files)



Dav upload php

- Upload php, then execute it. Bad.

RewriteEngine On

RewriteCond %{REQUEST_METHOD} =PUT [OR]

RewriteCond %{REQUEST_METHOD} =MOVE


RewriteRule ^/dav/(.*)\.php /dav/\$1.nophp



PHP when no file ext

- Force files with no file extension to be handled by php



- 
- Allows you to have URLs without the annoying “.php” on the end.

```
RewriteEngine On  
RewriteRule !\. - [H=application/x-httpd-php]
```

- 
- Doesn't contain a dot

RewriteEngine On
RewriteRule !\. - [H=application/x-httpd-php]

- 
- Don't rewrite it

```
RewriteEngine On  
RewriteRule !\. - [H=application/x-httpd-php]
```

- 
- Force it to use the php handler

```
RewriteEngine On  
RewriteRule !\. - [H=application/x-httpd-php]
```



Use PATH_INFO

- Now you can have URLs like

```
http://example.com/handler/arg1/arg2
```

- Use `$_SERVER[PATH_INFO]` to grab the additional bits of the request



mod_negotiation

- Might be able to do the same thing with mod_negotiation
- Options +MultiViews



<If>

- Just added
- Makes much of mod_rewrite unnecessary.
- <http://httpd.apache.org/docs/trunk/mod/core.html#if>

```
<If "$req{Host} = 'myhost.com'">
```



<If>

- Variable can be in \$req, \$resp, or \$env
- Any Request, Response, or Environment variable

```
<If "$env{foo} = 'bar'">
```



Related modules

- mod_substitute
- mod_ext_filter
- mod_proxy_html
- mod_line_edit



How do I do that ...

- Questions like “How do I do XYZ with mod_rewrite” often have the same answer
- YOU DON'T
- These modules are sometimes the right answer



mod_substitute

- New in 2.2.8
- In-stream regex
- Replace a string, or a pattern, in the output
- Chain with other filters



mod_substitute

- One directive: Substitute

```
<Location />
```

```
AddOutputFilterByType SUBSTITUTE text/html  
Substitute s/ariel/verdana/ni
```

```
</Location>
```



mod_substitute

- n = treat as a fixed string
- Default - treat as regex

```
<Location />
```

```
AddOutputFilterByType SUBSTITUTE text/html  
Substitute s/ariel/verdana/ni
```

```
</Location>
```



mod_substitute

- i - Case insensitive match
- Default - Case sensitive

```
<Location />
```

```
AddOutputFilterByType SUBSTITUTE text/html  
Substitute s/ariel/verdana/ni
```

```
</Location>
```



mod_substitute

- Replace **ariel** with **verdana** everywhere
- Filter content as it passes through. Perhaps on a proxy server.

```
<Location />
```

```
AddOutputFilterByType SUBSTITUTE text/html  
Substitute s/ariel/verdana/ni
```

```
</Location>
```



More usefully ...

- Replace hard-coded hostnames in HTML proxied from a back-end
- `s/intranet.local/www.corpsite.com/i`



mod_ext_filter

- Calls an external command to filter the stream
- Hugely inefficient



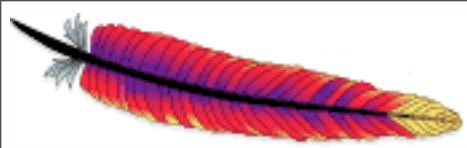
mod_proxy_html

- Rewrites HTML at the proxy
- Swap hostnames for absolute URLs
- Third-party module

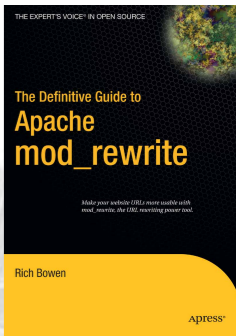


mod_line_edit

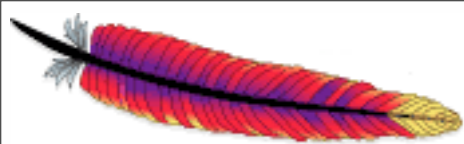
- Very similar to mod_substitute
- Third-party module



Further resources



- <http://rewrite.drbacchus.com/>
- <http://people.apache.org/~rbowen>
- “Definitive Guide to mod_rewrite” by Rich Bowen, from APress
- <http://httpd.apache.org/docs/2.2/rewrite/>



Questions?



THE EXPERT'S VOICE® IN OPEN SOURCE

The Definitive Guide to Apache mod_rewrite

*Make your website URLs more usable with
mod_rewrite, the URL rewriting power tool.*

Rich Bowen

Apress®

<http://people.apache.org/~rbowen/>

clearMYrecord.com