

mod_rewrite Cookbook

Rich Bowen
Asbury College
Apache Software Foundation
rbowen@apache.org





Agenda

- Common tasks with mod_rewrite
- A few advanced rewrite rules
- Some things you didn't know mod_rewrite could do



Quick intro

- Last time I did this, I misjudged my audience, so ...



Regex

- How many of you are brand-new to mod_rewrite?



Oh

- That many, hmm? Well, let's start with a few basics



RewriteRule

- Apply a regex to a request
- Modify a request in some way
- Alter aspects of the request in addition to the URI



RewriteRule Syntax

RewriteRule PATTERN TARGET

- Which means ...
 - If the request looks like THIS, send it HERE instead
- 



RewriteRule Syntax

RewriteRule PATTERN TARGET



- The pattern is a regular expression
- A substring match
- Describes what the request might look like



RewriteRule Syntax

RewriteRule PATTERN TARGET

- 
- The target is a URI, or perhaps a URL, or maybe a file path, depending on context



RewriteRule Syntax

RewriteRule PATTERN TARGET [X]

- The rule may be modified by one or more flags



When not to

- The most important thing is to know when not to use `mod_rewrite`
- Be aware of the tools to do things that `mod_write` is commonly misused for



Redirect

Redirect /one <http://example.com/two>



RedirectMatch

RedirectMatch [mM]onkey \
<http://example.com/ape>



Alias

Alias /images /var/upload/img



AliasMatch

- Like Alias, but with regular expressions



A word about SEO

- “Pretty” URLs don’t guarantee search engine ranking
- Most “SEO” is misinformation
- Content causes people to link to you, which, in turn, drives search engine rankings
- Don’t believe people who claim that they can guarantee the top spot on Google. They’re lying.



Mapping path to QS

- Mapping <http://example.com/one/two> to <http://example.com/index.php?x=one&y=two>



Step one

- First, the easy bit - map the path information to arguments:

```
RewriteEngine On  
RewriteRule ^/(.*)/(.*) \  
    /index.php?one=$1&two=$2 [PT]
```




Details

- Starts with slash:

RewriteEngine On
RewriteRule [^]/(.*)/(.*) \
/index.php?one=\$1&two=\$2 [PT]



Details

- Followed by some stuff

RewriteEngine On
RewriteRule ^/(.*)/(.*) \
/index.php?one=\$1&two=\$2 [PT]



Details

- Then another slash

RewriteEngine On
RewriteRule ^/(.*)/(.*) \
/index.php?one=\$1&two=\$2 [PT]



Details



- And some more stuff

RewriteEngine On
RewriteRule ^/(.*)/(.*) \
/index.php?one=\$1&two=\$2 [PT]



Details



- The first bit becomes \$1

RewriteEngine On
RewriteRule ^/(.*)/(.*) \
/index.php?one=\$1&two=\$2 [PT]



Details



- The second bit becomes \$2

RewriteEngine On

RewriteRule ^/(.*)/(.*) \

/index.php?one=\$1&two=\$2 [PT]



Details

- And [PT] ensures that php gets a whack at it

```
RewriteEngine On  
RewriteRule ^/(.*)/(.*) \  
    /index.php?one=$1&two=$2 [PT]
```



[PT]

- Passthrough
- Sends the URI back to the URL-mapping engine
- Ensures that things like Aliases, Redirects are honored
- Ensures that handlers (like PHP) fire



Unfortunately ...

- The rule is not precise enough
- It's rather prone to matching the wrong things



Better ...

- `.*` is too greedy. Use something more specific
- `[^/]` matches all “not slash” characters

RewriteEngine On

RewriteRule `^/([^\/*]+)/([^\/*]*) \`
`/index.php?one=$1&two=$2 [PT]`



Better ...

- Thus, for `/one/two/three`, `$1` is `'one'` and `$2` is `'two'`
- `'three'` would be silently discarded

RewriteEngine On

RewriteRule `^/([^\/]*)/([^\/]*) \`
`/index.php?one=$1&two=$2 [PT]`



Legitimate files

- Requests for real files would run afoul of this
- /images/toad.gif would be mapped to /index.php?one=images&two=toad.gif
- That's not what we want
- What is to be done? Alas and alack!



Ignore files, directories

- If it's **not a file**
- And **not a directory**

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^/([^/]+)/([^/]) \
    /index.php?one=$1&two=$2 [PT]
```




Existing Aliases, etc

- RewriteRule runs before things like Aliases and Redirects
- May need to explicitly exempt them

RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-f

RewriteCond %{REQUEST_FILENAME} !-d

RewriteCond %{REQUEST_URI} \

!^(icons|errors|styles|js)

RewriteRule ^/([^/]+)/([^/]+) \

/index.php?one=\$1&two=\$2 [PT]



Existing Aliases, etc

- RewriteRule runs before things like Aliases and Redirects
- May need to **explicitly exempt them**

RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-f

RewriteCond %{REQUEST_FILENAME} !-d

RewriteCond %{REQUEST_URI} \

!^(icons|errors|styles|js)

RewriteRule ^/([^/]*)/([^/]*) \

/index.php?one=\$1&two=\$2 [PT]



.htaccess

- Remember that in .htaccess files or <Directory> scope, you'll need to remove additional path information

RewriteEngine On

RewriteCond /var/www/%{REQUEST_FILENAME} !-f

RewriteCond /var/www/%{REQUEST_FILENAME} !-d

RewriteCond %{REQUEST_URI} \

!(icons|errors|styles|js)

RewriteRule ^([^\/*]+)/([^\/*]) \

index.php?one=\$1&two=\$2 [PT]



QSA

- If you need to preserve existing query string arguments, use QSA:

RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-f

RewriteCond %{REQUEST_FILENAME} !-d

RewriteCond %{REQUEST_URI} \
 !^/(icons|errors|styles|js)

RewriteRule ^/([^/]+)/([^/]+) \
 /index.php?one=\$1&two=\$2 [PT,QSA]



QSA

- Query String Append (qsappend)
- Existing query string is preserved
- New stuff is tacked on to the end



Virtual Hosts

- Dynamic name-based vhosts



Consider mod_vhost_alias

- Always try to avoid mod_rewrite if possible
- If there's another way, it's pretty much guaranteed to be more efficient.



But ...

- mod_vhost_alias has some unfortunate shortcomings
- (That's the polite way to say it)



Vhosts with mod_rewrite

- First, you'll need the hostname
- RewriteRule doesn't have access to the hostname
- You'll need to use RewriteCond for this



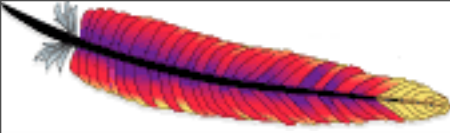
Hostname

- Snag the **first part of the hostname**
- Copy the **entire request** into a file path

RewriteEngine On

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

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



[NC]

- NC matches in a case-insensitive manner

RewriteEngine On
RewriteCond %{HTTP_HOST} (.*)\.example\.com [NC]
RewriteRule (.*) /home/%1/www\$1



Details

- This will need to go in a wildcard virtual host
- ServerAlias *.example.com
- Must have that in a wildcard DNS record, too
- The request uri starts with / so \$1 does too

RewriteEngine On

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

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



Aliases

- RewriteRules run prior to Aliases
- Exclude that from the rewrite

Alias **/icons/** /var/www/icons/

RewriteEngine On

RewriteCond %{REQUEST_URI} !**/icons/**

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

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



Using [S] as a “goto”

- RewriteCond applies ONLY to the RewriteRule immediately following it
- What if you want a RewriteCond to apply to multiple rules?
- Use the [S] flag to create a logical block



RewriteCond

- What you want:

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
# Do BOTH of the following
RewriteRule ^/icons/(.*) /var/www/icons/$1
RewriteRule (.*?) /home/bob/www$1
```

- Unfortunately, that's not what that does ...



[S]

- Instead, reverse the RewriteCond and use [S]

RewriteEngine On

RewriteCond %{REQUEST_FILENAME} -f

Skip the next two rules ...

RewriteRule ^ - [S=2]

RewriteRule ^/icons/(.*) /var/www/icons/\$1 [L]

RewriteRule (.*?) /home/bob/www\$1 [L]



[S]

RewriteEngine On

RewriteCond %{REQUEST_FILENAME} -f

Skip the next two rules ...

RewriteRule ^ - [S=2]

RewriteRule ^/icons/(.*) /var/www/icons/\$1 [L]

RewriteRule (.*?) /home/bob/www\$1 [L]

- [L] (last) says “do it now”. If the first rule runs, the second one won’t.



URL Handler

- aka "rewrite everything"
- All non-file requests go to handler.php



Two usual approaches:

- Rewrite the request as a query string
- Just rewrite, and let the handler figure it out
- I like the second approach a lot more, but it's less common.



Rewrite as query string

- The original request is passed to the handler as a query string:

RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-d

RewriteCond %{REQUEST_FILENAME} !-f

RewriteRule (.*) /index.php?q=\$1 [PT,L,QSA]



Details

RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-d

RewriteCond %{REQUEST_FILENAME} !-f

RewriteRule (.*) /index.php?q=\$1 [PT,L,QSA]

- Don't rewrite requests that already map to valid files
- This should handle images, css, js, existing html files, etc



Details

- Will NOT protect Aliases – you'll need to explicitly exclude those with RewriteCond

RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-d

RewriteCond %{REQUEST_FILENAME} !-f

RewriteCond %{REQUEST_URI} \
 !^(icons|cgi-bin)

RewriteRule (.*) /index.php?q=\$1 [PT,L,QSA]



Or ...

- Rewrite to the handler, let it figure it out

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_URI} \
    !^/(icons/cgi-bin)
RewriteRule (.*) /index.php [PT,L,QSA]
```




Handler



- For example:

```
<?php
$uri = $_SERVER['REQUEST_URI'];
$parts = explode('/', $uri);
// ... etc
?>
```



Basic RewriteMap

- 1-1 mapping using RewriteMap
- http2dbm and dbm rewrite maps



RewriteMap

- Creates a rewrite map or function
- Simplifies complicated RewriteRule directives
- Can call an external source for the rewrite logic



Internal RewriteMap

- RewriteMap int
- toupper
- tolower
- escape
- unescape

```
# Lower-case all requests  
RewriteMap lc int:tolower  
RewriteRule (.*) ${lc:$1}
```




mod_speling

```
# Lower-case all requests  
RewriteMap lc int:tolower  
RewriteRule (.*) ${lc:$1}
```

- If you're trying to make URLs case insensitive, mod_speling might be what you're looking for
- CheckSpelling On



RewriteMap txt

- Large number of static 1-1 mappings

dogs.txt

```
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
```




RewriteMap txt:

- Map <http://example.com/dog/poodle> to the correct URL

```
RewriteMap dogs txt:/etc/dogs.txt  
RewriteRule ^/dog/(.*) ${dogs:$1}
```



Default value

- What if it doesn't match anything?

```
RewriteMap dogs txt:/etc/dogs.txt  
RewriteRule ^/dog/(.*) ${dogs:$1|/index.php}
```




Updates

- mod_rewrite will reload the file if the mdate is updated
- Otherwise, contents of file are cached in memory



Caveat

- File is unindexed – lookups are slow



Faster

- Convert the text file to a dbm
- Indexed – faster lookups



httx+2dbm

- New in 2.x
- Converts to a dbm
- The example Perl code in the 1.3 docs doesn't actually work



txt2dbm

```
txt2dbm -i dogs.txt -o dogs.map
```



dbm

- Change previous config for dbm

```
RewriteMap dogs dbm:dogs.map  
RewriteRule ^/dog/(.*) ${dogs:$1}/index.php}
```




RewriteMap prg

- Simple Perl-based RewriteMap



Caveats

- One copy running – all child processes wait for it
- Should use a RewriteLock to avoid contention
- Use only as a last resort



The basics



```
RewriteEngine On  
RewriteMap name prg:/var/www/bin/map.pl  
RewriteRule (.*) ${name:$1}
```



Map script

- Request comes in on **STDIN**
- Do something useful with it
- Response output on **STDOUT**

```
#!/usr/bin/perl
while ( $req = <STDIN> ) {
    $resp = something_useful( $req );
    print $resp;
}
```




Standard example

- Convert “-” to “_” everywhere in a URI
- Necessary because RewriteRule doesn’t have a “global replace” flag
- Note that there’s a better way to do this. (I’ll show you in a moment.)



dash2score

- Run it on any URI which contains a dash:

```
RewriteMap d2s prg:/www/bin/dash2score.pl  
RewriteRule (.*-.* ) ${d2s:$1} [R]
```




dash2score.pl



- The script:

```
#!/usr/bin/perl
$| = 1;
while ( $req = <STDIN> ) {
    $req =~ s/-/_/g;
    print $req;
}
```



dash2score.pl



- Turn off buffering

```
#!/usr/bin/perl
$| = 1;
while ( $req = <STDIN> ) {
    $req =~ s/-/_/g;
    print $req;
}
```




dash2score.pl

- Loop for the lifetime of the server

```
#!/usr/bin/perl  
$| = 1;  
while ( $req = <STDIN> ) {  
    $req =~ s/-/_/g;  
    print $req;  
}
```



dash2score.pl

- Replace “-” with “_” globally

```
#!/usr/bin/perl  
$| = 1;  
while ( $req = <STDIN> ) {  
    $req =~ s/-/_/g;  
    print $req;  
}
```




dash2score.pl



- Print the resulting string

```
#!/usr/bin/perl
$| = 1;
while ( $req = <STDIN> ) {
    $req =~ s/-/_/g;
    print $req;
}
```



Better way

- Now I will show you a more excellent way
- The [N] flag actually does have the occasional use

```
# Replace "-" with "_" and start over  
RewriteRule (.*)-(.*?) $1_$2 [N]
```




So ...

- It's a simple example
- But not particularly practical
- Useful examples are too complicated to fit on the screen
- See also the dbd: method ...



dbd:

- New in 2.3
- Store rewrite maps in a sql database

```
DBDriver mysql
```

```
DBDParams \
```

```
    host=localhost,user=bob,pass=larry,dbname=rewrite
```

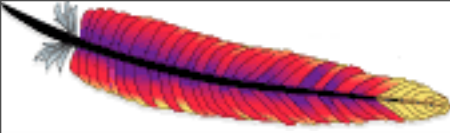
```
RewriteMap mymap \
```

```
    "dbd: select dest from rewrite where uri = %s"
```




Look elsewhere

- If the file isn't here, look there
- Smarter than just a 404 error page
- Useful when you've got files stored in a couple possible locations



-f

- Check for the existence of a file
- If it's not there, look somewhere else

```
<Directory /var/www/one>  
RewriteEngine On  
RewriteCond /var/www/one/%{REQUEST_FILENAME} !-f  
RewriteRule (.*) /var/www/two/$1  
</Directory>
```




<Directory>

- In a <Directory> everything is relative to the local path

```
<Directory /var/www/one>
```

```
RewriteEngine On
```

```
RewriteCond /var/www/one/{REQUEST_FILENAME} !-f
```

```
RewriteRule (.*?) /var/www/two/$1
```

```
</Directory>
```




Or, more than one place

```
<Directory /var/www/one>  
RewriteEngine On  
RewriteCond /var/www/one/%{REQUEST_FILENAME} -f  
RewriteRule (.*?) /var/www/one/$1 [L]  
RewriteCond /var/www/two/%{REQUEST_FILENAME} -f  
RewriteRule (.*?) /var/www/two/$1 [L]  
RewriteCond /var/www/three/%{REQUEST_FILENAME} -f  
RewriteRule (.*?) /var/www/three/$1 [L]  
</Directory>
```




ErrorDocument

- Note that you could accomplish the same thing with an ErrorDocument handler script

ErrorDocument 404 /handler/404.cgi



Image theft

- “Image theft” refers to folks including their images in their web pages
- Uses your bandwidth, your copyright
- Want to deny requests that don’t originate from your own pages
- Can check the referer



Referer

- Ensure that the referer comes from here
- If the referer **isn't from here** ...

RewriteEngine On

RewriteCond **%{HTTP_REFERER} !myhost.com**

RewriteRule \.(gif|jpg|png)\$ - [NC,F]



Image request

- And if the request was for an image ...

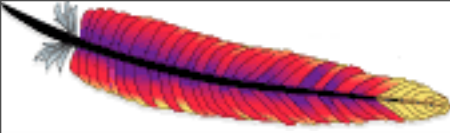
```
RewriteEngine On  
RewriteCond %{HTTP_REFERER} !myhost.com  
RewriteRule \.(gif|jpg|png)$ - [NC,F]
```




Forbidden

- Don't rewrite it, just fail the request

```
RewriteEngine On  
RewriteCond %{HTTP_REFERER} !myhost.com  
RewriteRule \.(gif|jpg|png)$ - [NC,F]
```



[NC]

- upper- or lower-case

```
RewriteEngine On  
RewriteCond %{HTTP_REFERER} !myhost.com  
RewriteRule \.(gif|jpg|png)$ - [NC,F]
```




no Referer?

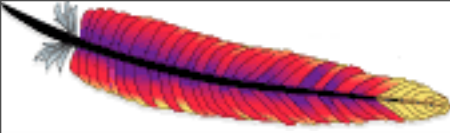
- Ensure that there is a non-null referer
- Some requests won't send one

RewriteEngine On

RewriteCond %{HTTP_REFERER} !myhost.com

RewriteCond %{HTTP_REFERER} .

RewriteRule \.(gif|jpg|png)\$ - [NC,F]



[F]

- This just forbids the request
- What if you want to ...

```
RewriteEngine On  
RewriteCond %{HTTP_REFERER} !myhost.com  
RewriteCond %{HTTP_REFERER} .  
RewriteRule \.(gif|jpg|png)$ - [NC,F]
```




Another image

- Display a “go away” image instead

RewriteEngine On

RewriteCond %{HTTP_REFERER} !myhost.com

RewriteCond %{HTTP_REFERER} .

RewriteRule \.(gif|jpg|png)\$ \

/images/goaway.gif [PT,L]



Another site

- Or perhaps another site entirely

RewriteEngine On

RewriteCond %{HTTP_REFERER} !myhost.com

RewriteCond %{HTTP_REFERER} .

RewriteRule \.(gif|jpg|png)\$ \

<http://other.site.com/images/x.jpg> [R,L]



Where they came from

- Or just back where they came from

RewriteEngine On

RewriteCond %{HTTP_REFERER} !myhost.com

RewriteCond %{HTTP_REFERER} .

RewriteRule \.(gif|jpg|png)\$ \

%{HTTP_REFERER} [R,L]



Canonical hostname

- Enforce a particular hostname
- Perhaps cookies require a particular hostname
- Or perhaps it's just preferred



Canonical hostname

- If it's **NOT** the preferred hostname

```
<VirtualHost *:80>
```

```
ServerName www.example.com
```

```
ServerAlias example.com
```

```
RewriteEngine On
```

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

```
RewriteRule (.*) http://www.example.com$1 [R,L]
```

```
</VirtualHost>
```




Canonical hostname

- Then **redirect** it there

```
<VirtualHost *:80>
```

```
ServerName www.example.com
```

```
ServerAlias example.com
```

```
RewriteEngine On
```

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

```
RewriteRule (.*) http://www.example.com$1 [R,L]
```

```
</VirtualHost>
```




.htaccess

- Or, if you have to put it in a .htaccess file:

```
RewriteEngine On  
RewriteCond %{HTTP_HOST} !=www.example.com  
RewriteRule (.*) http://www.example.com/$1 [R,L]
```



Canonical hostname

- May make sense to have two vhosts:

```
<VirtualHost *:80>  
  ServerName www.example.com  
  # ...  
</VirtualHost>
```

```
<VirtualHost *:80>  
  ServerName example.com  
  RedirectMatch /(.*) http://www.example.com/$1  
</VirtualHost>
```




http2https

- Redirect http requests to https
- Require https on certain resources



Like canonical hostname

- Special case of the previous rule
- Sort of



HTTPS

- If it's not already HTTPS

```
RewriteEngine On  
RewriteCond %{HTTPS} !=on  
RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R,L]
```



HTTPS



- Redirect it

```
RewriteEngine On  
RewriteCond %{HTTPS} !=on  
RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R,L]
```




Clever trick

- Usable in .htaccess or main config
- Slash is optional

```
RewriteEngine On  
RewriteCond %{HTTPS} !=on  
RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R,L]
```



Just one directory

- Some folks want one particular directory SSL
- And everything else NOT SSL
- I think this is silly, but the customer is always right



One directory

```
RewriteEngine On  
RewriteRule %{HTTPS} !=on  
RewriteRule ^/?secure(.*) \  
    https://%{SERVER_NAME}/secure$1 [R,L]
```

```
RewriteRule %{HTTPS} =on  
RewriteCond %{HTTP_REQUEST} !^/?secure  
RewriteRule ^/?(.*) \  
    http://%{SERVER_NAME}/$1 [R,L]
```




One directory

RewriteEngine On

RewriteRule **%{HTTPS} !=on**

RewriteRule **^/?secure(.*) **

https://%{SERVER_NAME}/secure\$1 [R,L]

RewriteRule **%{HTTPS} =on**

RewriteCond **%{HTTP_REQUEST} !^/?secure**

RewriteRule **^/?(.*) **

http://%{SERVER_NAME}/\$1 [R,L]



Caveat

- HTTPS pages containing non-HTTPS content (images, css, js) will generate browser errors.
- Ensure that these files are available via HTTP, HTTPS



Proxy

- Using rewriterules to force Proxy



Images elsewhere



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



Server Migration

- Proxy 404 requests through to old server

```
RewriteEngine On  
RewriteCond %{REQUEST_URI} !-U  
RewriteRule (.*) http://old.server$1 [P]
```




-U

- -U' (is existing URL, via subrequest)
- Checks whether or not TestString is a valid URL, accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to do the check, so use it with care - it can impact your server's performance!

```
RewriteEngine On  
RewriteCond %{REQUEST_URI} !-U  
RewriteRule (.*) http://old.server$1 [P]
```




lighttpd for static

- php stuff here, everything else on lighttpd

```
RewriteEngine On  
RewriteCond %{REQUEST_URI} !\.php$  
RewriteRule (.*) http://x.example.com$1 [P]  
ProxyPassReverse / http://x.example.com/
```




ProxyPassReverse

- Redirects usually contain the hostname
- Ensures that Redirects come from here instead of from there

```
RewriteEngine On  
RewriteCond %{REQUEST_URI} !\.php$  
RewriteRule (.*) http://x.example.com$1 [P]  
ProxyPassReverse / http://x.example.com/
```



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



Reading the RewriteLog

RewriteLog logs/rewrite.log
RewriteLogLevel 9

- Can't go in a .htaccess file
- Needs to be in global or vhost scope



Reading the RewriteLog

RewriteLog **logs/rewrite.log**
RewriteLogLevel 9

- Location of the log file
- Relative to ServerRoot



Reading the RewriteLog

RewriteLog logs/rewrite.log
RewriteLogLevel 9

- Number from 0 to 9
- 9 most verbose
- Below 3 not particularly useful



Log entries

- Let's look at a few log entries:

- 
- I recommend ignoring this bit:

203.167.144.193 - - [05/Nov/2007:22:29:53
--0500] [wooga.drbbacchus.com/sid#b93592c0]
[rid#b95c6c98/initial] (3) applying pattern '^/
books?/(.+) ' to uri '/favicon.ico'



In fact, before we go on

- I actually use a piped log handler to remove this superfluous stuff
- Like so ...

```
RewriteLog /usr/local/bin/rewrite_log_pipe  
RewriteLogLevel 9
```




```
RewriteLog |/usr/local/bin/rewrite_log_pipe  
RewriteLogLevel 9
```

- with ...

```
#!/usr/bin/perl  
$|++;  
  
open (F, ">>/tmp/rewrite");  
select F;  
while (<>) {  
    s/^.*(\(\d\).*)/$1/;  
    print;  
}
```

```
#!/usr/bin/perl  
$|++;
```

```
open (F, ">>/tmp/rewrite");  
select F;  
while (<>) {  
    s/^.*(\d).*/$1/;  
    print;  
}
```

- Look for the (1) or (2) bit
- drop everything before that



Results in:

(4) RewriteCond: input='wooga.dr bacchus.com' pattern='!^wooga\.dr bacchus
\.com' [NC] => not-matched

(3) applying pattern 'wp-rss2.php' to uri '/index.php'

(3) applying pattern '(journal/)?index.rdf' to uri '/index.php'

(3) applying pattern '^/wordpress/wp-comments' to uri '/index.php'

(3) applying pattern '^/perm/(.*)' to uri '/index.php'

(3) applying pattern '^/articles?/(.*)' to uri '/index.php'

(3) applying pattern '^/blog/(.*)' to uri '/index.php'

(3) applying pattern '^/book/(mod)?_?rewrite' to uri '/index.php'

(3) applying pattern '^/book/cookbook' to uri '/index.php'

(3) applying pattern '^/book/2.2' to uri '/index.php'

(3) applying pattern '^/booklink/(.*)' to uri '/index.php'

(3) applying pattern '^/books?/(.+)' to uri '/index.php'

(1) pass through /index.php



Requested URI

(4) RewriteCond: input='wooga.drbacchus.com' pattern='!^wooga\.drbacchus
\.com' [NC] => not-matched

(3) applying pattern 'wp-rss2.php' to uri '/index.php'

(3) applying pattern '(journal/)?index.rdf' to uri '/index.php'

(3) applying pattern '^/wordpress/wp-comments' to uri '/index.php'

(3) applying pattern '^/perm/(.*)' to uri '/index.php'

(3) applying pattern '^/articles?/(.*)' to uri '/index.php'

(3) applying pattern '^/blog/(.*)' to uri '/index.php'

(3) applying pattern '^/book/(mod)?_?rewrite' to uri '/index.php'

(3) applying pattern '^/book/cookbook' to uri '/index.php'

(3) applying pattern '^/book/2.2' to uri '/index.php'

(3) applying pattern '^/booklink/(.*)' to uri '/index.php'

(3) applying pattern '^/books?/(.+)' to uri '/index.php'

(1) pass through /index.php



Patterns applied

(4) RewriteCond: input='wooga.drbacchus.com' pattern='!^wooga\.drbacchus
\.com' [NC] => not-matched

(3) applying pattern 'wp-rss2.php' to uri '/index.php'

(3) applying pattern '(journal/)?index.rdf' to uri '/index.php'

(3) applying pattern '^/wordpress/wp-comments' to uri '/index.php'

(3) applying pattern '^/perm/(.*)' to uri '/index.php'

(3) applying pattern '^/articles?/(.*)' to uri '/index.php'

(3) applying pattern '^/blog/(.*)' to uri '/index.php'

(3) applying pattern '^/book/(mod)?_?rewrite' to uri '/index.php'

(3) applying pattern '^/book/cookbook' to uri '/index.php'

(3) applying pattern '^/book/2.2' to uri '/index.php'

(3) applying pattern '^/booklink/(.*)' to uri '/index.php'

(3) applying pattern '^/books?/(.+)' to uri '/index.php'

(1) pass through /index.php



None of them matched

(4) RewriteCond: input='wooga.drbacchus.com' pattern='!^wooga\.drbacchus
\.com' [NC] => not-matched

(3) applying pattern 'wp-rss2.php' to uri '/index.php'

(3) applying pattern '(journal/)?index.rdf' to uri '/index.php'

(3) applying pattern '^/wordpress/wp-comments' to uri '/index.php'

(3) applying pattern '^/perm/(.*)' to uri '/index.php'

(3) applying pattern '^/articles?/(.*)' to uri '/index.php'

(3) applying pattern '^/blog/(.*)' to uri '/index.php'

(3) applying pattern '^/book/(mod)?_?rewrite' to uri '/index.php'

(3) applying pattern '^/book/cookbook' to uri '/index.php'

(3) applying pattern '^/book/2.2' to uri '/index.php'

(3) applying pattern '^/booklink/(.*)' to uri '/index.php'

(3) applying pattern '^/books?/(.+)' to uri '/index.php'

(1) **pass through /index.php**



And now

- We can actually make some sense of what's happening
- Less inscrutable noise
- Yes, it means something, but not to normal people



Examples

(4) RewriteCond: input='wooga.drbacchus.com' pattern='!^wooga\.drbacchus\.com' [NC] => not-matched

- This was the result of

RewriteCond %{HTTP_HOST} \
!^wooga\.drbacchus\.com [NC]



Examples

(4) RewriteCond: input='wooga.drbacchus.com' pattern='!^wooga\.drbacchus
\.com' [NC] => not-matched

- It shows what the input variable looked like

RewriteCond **%{HTTP_HOST}** \
!^wooga\.drbacchus\.com [NC]



Examples

(4) RewriteCond: input='wooga.drbacchus.com' pattern='![^]wooga\.drbacchus
\.com' [NC] => not-matched

- And what pattern was applied

RewriteCond %{HTTP_HOST} \
![^]wooga\.drbacchus\.com [NC]



Examples

(4) RewriteCond: input='wooga.drbacchus.com' pattern='!^wooga\.drbacchus
\.com' [NC] => **not-matched**

- As well as what happened

RewriteCond %{HTTP_HOST} \
!^wooga\.drbacchus\.com [NC]



Another example

(3) applying pattern '^/book/(mod)?_?rewrite'
to uri '/index.php'

- Was a result of

```
RewriteRule ^/book/(mod)?_?rewrite \  
http://www.amazon.com/exec/obidos/asin/  
1590595610/dr bacchus/ [R,L]
```




Again ...

(3) applying pattern '^/book/(mod)?_?rewrite'
to uri '/index.php'

- What was requested

```
RewriteRule ^/book/(mod)?_?rewrite \  
http://www.amazon.com/exec/obidos/asin/  
1590595610/dr bacchus/ [R,L]
```




And ...

(3) applying pattern `^/book/(mod)?_?rewrite'`
to uri `/index.php'`

- What it was compared against

```
RewriteRule ^/book/(mod)?_?rewrite \  
http://www.amazon.com/exec/obidos/asin/  
1590595610/dr bacchus/ [R,L]
```




Matched?

(3) applying pattern '^/book/(mod)?_?rewrite'
to uri '/index.php'

- If it matched, the next line will be the action log

```
RewriteRule ^/book/(mod)?_?rewrite \  
http://www.amazon.com/exec/obidos/asin/  
1590595610/dr bacchus/ [R,L]
```




The whole thing

(3) applying pattern '^/books?/(mod)?_?rewrite' to uri '/books/rewrite'

(2) rewrite '/books/rewrite' -> 'http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/'

(2) explicitly forcing redirect with http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/

(1) escaping http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/ for redirect

(1) redirect to http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/ [REDIRECT/302]



The match:

(3) applying pattern '^/books?/(mod)?_?rewrite' to uri '/books/rewrite'

(2) rewrite '/books/rewrite' -> 'http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/'

(2) explicitly forcing redirect with http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/

(1) escaping http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/ for redirect

(1) redirect to http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/ [REDIRECT/302]



Followed by

(3) applying pattern '^/books?/(mod)?_?rewrite' to uri '/books/rewrite'

(2) rewrite '/books/rewrite' -> 'http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/'

(2) explicitly forcing redirect with http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/

(1) escaping http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/ for redirect

(1) redirect to http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/ [REDIRECT/302]



[R]

(3) applying pattern '^/books?/(mod)?_?rewrite' to uri '/books/rewrite'

(2) rewrite '/books/rewrite' -> 'http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/'

(2) explicitly forcing redirect with <http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/>

(1) escaping <http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/> for redirect

(1) redirect to <http://www.amazon.com/exec/obidos/asin/1590595610/dr bacchus/> [REDIRECT/302]



But it all runs together!

- Look for:

(2) **init rewrite engine** with requested uri /atom/1

- 'init rewrite engine' shows where a new request started being rewritten



.htaccess

- Can't use RewriteLog in a .htaccess file
- Test it on your dev server before moving to live server



TRACE



RewriteEngine on
RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
RewriteRule ^ - [F]



Security scanners

- Report this as a vulnerability
- It isn't
- But it doesn't hurt to shut them up



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>
```




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



fin

- rbowen@apache.org
- <http://drbacchus.com/>
- <http://drbacchus.com/books/>
- <http://people.apache.org/~rbowen>
- <http://httpd.apache.org/docs/2.2/rewrite/>