.htaccess中的apache rewrite规则写法详解

.htaccess中的apache rewrite写法：

```
RewriteEngine On
RewriteCond %{HTTP_HOST} ^(www\.)?xx
RewriteCond %{REQUEST_URI} !^/blog/
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ /blog/$1
# 没有输入文件名的默认到到首页
RewriteCond %{HTTP_HOST} ^(www\.)?xx
RewriteRule ^(/)?$ blog/index.php [L
```

下面我开始解说一下上面的意思：

【RewriteEngine On】表示重写引擎开，关闭off，作用就是方便的开启或关闭以下的语句，这样就不需要一条一条的注释语句了。

【RewriteCond %{HTTP_HOST} ^(www\.)?xxx\.com$】
这是重写条件，前面%{HTTP_HOST}表示当前访问的网址，只是指前缀部分，格式是www.xxx.com不包括“http://”和“/”，^表示字符串开始，$表示字符串结尾，\.表示转义的.，如果不转义也行，推荐转义，防止有些服务器不支持，?表示前面括号www\.出现0次或1次，这句规则的意思就是如果访问的网址是xxx.com或者 www.xxx.com就执行以下的语句，不符合就跳过。

【RewriteCond %{REQUEST_URI} !^/blog/】
也是重写条件，%{REQUEST_URI}表示访问的相对地址，就是相对根目录的地址，就是域名/后面的成分，格式上包括最前面的“/”，!表示非，这句语句表示访问的地址不以/blog/开头，只是开头^，没

有结尾$

【RewriteCond %{REQUEST_FILENAME} !-f】
【RewriteCond %{REQUEST_FILENAME} !-d】
这两句语句的意思是请求的文件或路径是不存在的，如果文件或路径存在将返回已经存在的文件或路径

【RewriteRule ^(.*)$ /blog/$1】重写规则，最重要的部分，意思是当上面的RewriteCond条件都满足的时候，将会执行此重写规则，^(.*)$是一个正则表达的 匹配，匹配的是当前请求的URL，^(.*)$意思是匹配当前URL任意字符，.表示任意单个字符，*表示匹配0次或N次（N>0），后面 /blog/$1是重写成分，意思是将前面匹配的字符重写成/blog/$1，这个$1表示反向匹配，引用的是前面第一个圆括号的成分，即^(.*)$中 的.* ，其实这儿将会出现一个问题，后面讨论。

【RewriteCond %{HTTP_HOST} ^(www\.)?xxx\.com$】
【RewriteRule ^(/)?$ blog/index.php [L]】
这两句的意思是指请求的host地址是www.xxx.com是，如果地址的结尾只有0个或者1个“/”时，将会重写到子目录下的主页，我猜想这主要因为重写后的地址是不能自动寻找主页的，需要自己指定。

现在说说出现的问题，RewriteRule ^(.*)$ /blog/$1前部分^(.*)$将会匹配当前请求的url，例如：请求网址是http://www.xxx.com/a.html，到底是匹配整个 http://www.xxx.com/a.html，还是只匹配/a.html即反斜杠后面的成分，还是只匹配

a.html。

答案是：根据RewriteBase规则规定，如果rewritebase 为/，将会匹配a.html，不带前面的反斜杠，所以上条语句应该写成RewriteRule ^(.*)$ blog/$1（不带/），不过实际应用上带上前面的反斜杠，也可以用，可能带不带都行。现在问题出来了，如果不设置rewritebase 为/ ，将会匹配整个网址http://www.xxx.com/a.html，显然这是错误的，所以应该添加这条：

RewiteBase /

还有一个问题是，不能保证每个人输入的网址都是小写的，如果输入大写的呢，linux系统是区分大小写的，所以应该在RewriteCond后添加[NC]忽略大小写的。

至此，完整的语句应该是：

```
####start####
RewriteEngine On
RewiteBase /
RewriteCond %{HTTP_HOST} ^(www\.)?xx
RewriteCond %{REQUEST_URI} !^/blog/
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ blog/$1
# 没有输入文件名的默认到到首页
RewriteCond %{HTTP_HOST} ^(www\.)?xx
RewriteRule ^(/)?$ blog/index.php [L
####end####
```

如果后面还继续有语句的，就不应该加上最后的[L]，因为这是表示最后一条语句的意思

防盗链的语句，同样需要添加RewiteBase /，如下：

```
RewriteEngine on
RewiteBase /
RewriteCond %{HTTP_REFERER} !^$ [NC]
```

```
4 RewriteCond %{HTTP_REFERER} !xxx.inf
5 RewriteRule \.(jpg|gif|png|bmp|swf|j
```

如果后面还继续有语句的，就不应该加上最后的[L]，/error/daolian.gif为别人盗链时显示的图片。

下面附上简单的语法规则和flags：

【RewriteCond语法：】

RewriteCond TestString CondPattern [flags]

rewritecond的其他用法：

‘-d’ (目录)

将TestString视为一个路径名并测试它是否为一个存在的目录。

‘-f’ (常规文件)

将TestString视为一个路径名并测试它是否为一个存在的常规文件。

‘-s’ (非空的常规文件)

将TestString视为一个路径名并测试它是否为一个存在的、尺寸大于0的常规文件。

‘-l’ (符号连接)

将TestString视为一个路径名并测试它是否为一个存在的符号连接。

‘-x’ (可执行)

将TestString视为一个路径名并测试它是否为一个存在的、具有可执行权限的文件。该权限由操作系统检测。

‘-F’ (对子请求存在的文件)

检查TestString是否为一个有效的文件，而且可以在服务器当前的访问控制配置下被访问。它使用一个内部子请求来做检查，由于会降低服务器的性能，所以请谨慎使用!

‘-U’ (对子请求存在的URL)

检查TestString是否为一个有效的URL，而且可以在服务器当前的访问控制配置下被访问。它使用一个

内部子请求来做检查，由于会降低服务器的性能，所以请谨慎使用!

【RewriteRule语法：】
RewriteRule Pattern Substitution [flags]

【flags】:

‘chain|C’ (链接下一规则)
此标记使当前规则与下一个规则相链接。它产生这样的效果：如果一个规则被匹配，则继续处理其后继规则，也就是这个标记不起作用；如果该规则不被匹 配，则其后继规则将被跳过。比如，在一个目录级规则中执行一个外部重定向时，你可能需要删除”.www”(此处不应该出现”.www”)。

‘cookie|CO=NAME:VAL:domain[:lifetime[:path]]’ (设置cookie)
在客户端设置一个cookie。cookie的名称是NAME，值是VAL。domain是该cookie的域，比如’.apache.org’，可选的lifetime是cookie的有效期(分钟)，可选的path是cookie的路径。

‘env|E=VAR:VAL’ (设置环境变量)
此标记将环境变量VAR的值为VAL，VAL可以包含可扩展的正则表达式反向引用($N和%N)。此标记可以多次使用以设置多个变量。这些变量可 以在其后许多情况下被间接引用，通常是在XSSI(<!–#echo var=”VAR”–>)或CGI($ENV{‘VAR’})中，也可以在后继的RewriteCond指令的CondPattern参数中通过% {ENV:VAR}引用。使用它可以记住从URL中剥离的信息。

‘forbidden|F’（强制禁止URL)
强制禁止当前URL，也就是立即反馈一个HTTP响应码403(被禁止的)。使用这个标记，可以链接若干个RewriteConds来有条件地阻塞某些URL。

‘gone|G’（强制废弃URL)
强制当前URL为已废弃，也就是立即反馈一个HTTP响应码410(已废弃的)。使用这个标记，可以标明页面已经被废弃而不存在了。

‘handler|H=Content-handler’（强制指定内容处理器)
强自制定目标文件的内容处理器为Content-handler。例如，用来模拟mod_alias模块的ScriptAlias指令，以强制映射文件夹内的所有文件都由”cgi-script”处理器处理。

‘last|L’（结尾规则)
立即停止重写操作，并不再应用其他重写规则。它对应于Perl中的last命令或C语言中的break命令。这个标记用于阻止当前已被重写的URL被后继规则再次重写。例如，使用它可以重写根路径的URL(‘/’)为实际存在的URL(比如：’/e/www/’)。

‘next|N’（从头再来)
重新执行重写操作(从第一个规则重新开始)。此时再次进行处理的URL已经不是原始的URL了，而是经最后一个重写规则处理过的URL。它对应于 Perl中的next命令或C语言中的continue命令。此标记可以重新开始重写操作(立即回到循环的开头)。但是要

小心，不要制造死循环！

‘nocase|NC’（忽略大小写）
它使Pattern忽略大小写，也就是在Pattern与当前URL匹配时，‘ A-Z’ 和’ a-z’ 没有区别。

‘noescape|NE’（在输出中不对URI进行转义）
此标记阻止mod_rewrite对重写结果应用常规的URI转义规则。 一般情况下，特殊字符(‘%’ , ‘$’ , ‘;’ 等)会被转义为等值的十六进制编码(‘%25’, ‘%24’, ‘%3B’ 等)。此标记可以阻止这样的转义，以允许百分号等符号出现在输出中，比如：
RewriteRule /foo/(.*) /bar?arg=P1\%3d$1 [R,NE]
可以使’ /foo/zed转向到一个安全的请求’ /bar?arg=P1=zed’ 。

‘nosubreq|NS’（不对内部子请求进行处理）
在当前请求是一个内部子请求时，此标记强制重写引擎跳过该重写规则。比如，在mod_include试图搜索目录默认文件(index.xxx) 时，Apache会在内部产生子请求。对于子请求，重写规则不一定有用，而且如果整个规则集都起作用，它甚至可能会引发错误。所以，可以用这个标记来排除 某些规则。
使用原则：如果你为URL添加了CGI脚本前缀，以强制它们由CGI脚本处理，但对子请求处理的出错率(或者资源开销)很高，在这种情况下，可以使用这个标记。

‘proxy|P’（强制为代理）
此标记使替换成分被内部地强制作为代理请求发

送，并立即中断重写处理，然后把处理移交给mod_proxy模块。你必须确保此替换串是一个能够被 mod_proxy处理的有效URI(比如以http://hostname开头)，否则将得到一个代理模块返回的错误。使用这个标记，可以把某些远程成 分映射到本地服务器域名空间，从而增强了ProxyPass指令的功能。
注意：要使用这个功能，必须已经启用了mod_proxy模块。

‘passthrough|PT’ (移交给下一个处理器)
此标记强制重写引擎将内部request_rec结构中的uri字段设置为filename字段的值，这个小小的修改使得RewriteRule 指令的输出能够被(从URI转换到文件名的)Alias, ScriptAlias, Redirect等指令进行后续处理[原文：This flag is just a hack to enable post-processing of the output of RewriteRule directives, using Alias, ScriptAlias, Redirect, and other directives from various URI-to-filename translators.]。举一个能说明其含义的例子： 如果要将/abc重写为/def， 然后再使用mod_alias将/def转换为/ghi，可以这样：
RewriteRule ^/abc(.*) /def$1 [PT]
Alias /def /ghi
如果省略了PT标记，虽然将uri=/abc/…重写为filename=/def/…的部分运作正常，但是后续的mod_alias在试图将URI转换到文件名时会遭遇失效。
注意：如果需要混合使用多个将URI转换到文件名的模块时，就必须使用这个标记。。此处混合使用mod_alias和mod_rewrite就是个典型的例子。

‘qsappend|QSA’(追加查询字符串)
此标记强制重写引擎在已有的替换字符串中追加一个查询字符串，而不是简单的替换。如果需要通过重写规则在请求串中增加信息，就可以使用这个标记。

‘redirect|R [=code] ‘(强制重定向)
若Substitution以http://thishost[:thisport]/(使新的URL成为一个URI)开头，可以强制性执行一个 外部重定向。如果没有指定code，则产生一个HTTP响应码302(临时性移动)。如果需要使用在300-400范围内的其他响应代码，只需在此指定即 可(或使用下列符号名称之一：temp(默认), permanent, seeother)。使用它可以把规范化的URL反馈给客户端，如将”/~”重写为”/u/”，或始终对/u/user加上斜杠，等等。
注意：在使用这个标记时，必须确保该替换字段是一个有效的URL。否则，它会指向一个无效的位置！并且要记住，此标记本身只是对URL加上http://thishost[:thisport]/前缀，重写操作仍然会继续进行。通常，你还会希望停止重写操作而立即重定向，那么就还需要使 用’ L'标记。

’ skip|S=num’(跳过后继规则)
此标记强制重写引擎跳过当前匹配规则之后的num个规则。它可以模拟if-then-else结构：最后一个规则是then从句，而被跳过的skip=N个规则是else从句。注意：它和’ chain|C’ 标记是不同的！

‘type|T=MIME-type’(强制MIME类型)
强制目标文件的MIME类型为MIME-type，可以用来基于某些特定条件强制设置内容类型。比如，下

面的指令可以让.php文件在 以.phps扩展名调用的情况下由mod_php按照PHP源代码的MIME类型(application/x-httpd-php-source)显 示:
RewriteRule ^(.+\.php)s$ $1 [T=application/x-httpd-php-source]

如果熟练掌握rewrite规则的编写，能够加强对网站URL的控制，对用户体验、SEO都十分有利。

一、防盗链功能

```
1 RewriteEngine On
2 RewriteCond %{HTTP_REFERER} !^http:/
3 RewriteCond %{HTTP_REFERER} !^$
4 RewriteRule .*.(jpe?g|gif|bmp|png)$
```

逐行讲解一下:

1.打开Rewrite功能。有可能服务器设置里已经是全局下打开了，但是多写也没事。
2.RewriteCond指令，定义生效条件，用于寻找匹配条件的地址。后面内容用正则表达式匹配。代表含义是发送的请求不由mysite.com而来，那就是盗链啦。末尾的[NC]代表忽略大小写。
3.发送请求的主机前缀不为空。
4.RewriteRule指令，定义重写规则，把匹配的地址按此规则重写。本例中把这些后缀为这些图片格式的，都替换到某一个图片下。[L]表示这是最后一段规则。

只这四行就实现了防盗链是不是很神奇，编写起来是不是又觉得复杂。
这里总结了几个常用参数（不是全部）：

RewriteCond下：

[NC] 不分字母大小写

[OR] 用于连接下一条规则

RewriteRule下：

[R] 强制重定向，[R=code] code默认为302

[F] 禁用URL，返回HTTP 403 错误

[L] 这是最后一条规则，之后内容无用

还有一篇关于正则表达式的教程（很详细）：

http://www.unibetter.com/deerchao/zhengzhe-biaodashi-jiaocheng-se.htm

二、网址规范化

```
1 Options +FollowSymLinks
2 rewriteEngine on
3 rewriteCond %{http_host} ^yourdomain
4 rewriteRule ^(.*)$ http://www.yourdo
```

这个是把所有二级域名都重定向到www.yourdomain.com的例子，现在看来是不是很简单了?

需要注意的是，这里的Options +FollowSymLinks不是必须的，但在某些服务器如果不设置FollowSymLinks，可能引起500错误。

再来看一个好玩的重定向，把google蜘蛛指向某个网站

```
1 RewriteEngine On
2 RewriteBase /
3 RewriteCond %{HTTP_USER_AGENT} (Goog
4 RewriteRule ^ http://abc.com/ [R=301
```

1.打开Rewrite功能。

2.RewriteBase指令，设置目录级重写的基准URL。可以理解成把该目录（这个.htaccess所在目录）假定为基准的URL前缀。本例中这样的写法无用。

3.RewriteCond指令。匹配所有USER_AGENT为Googlebot的发送请求。

4.RewriteRule指令。本例中把这些请求都重定向到了abc.com。

在本例中，这个配置应该是黑客所为，把google蜘蛛指向某个网站，等于伪造PR。

三、临时错误页面

当你的网站在升级、修改的时候，你最好让访客转到指定的页面，而不是没做完的页面或者是错误页。
这时我们做一个302转跳就好。

```
1 RewriteEngine on
2 RewriteCond %{REQUEST_URI} !/mainten
3 RewriteCond %{REMOTE_ADDR} !^123.123
4 RewriteRule $ /error.html [R=302,L]
```

1.继续打开Rewrite功能。
2.REQUEST_URI，请求的URL值。这里指所有访问maintenance.html页面的请求。
3.REMOTE_ADDR，向服务器发送请求的IP地址。本例中此处应设为你自己的IP，这样就只有你能访问。
4.RewriteRule指令。本例中把这些请求都重定向到了error.html 。

在本例，我们总结几个常用的正则表达式和特殊符号。

(.*) 用于匹配某一区域内所有内容。如abc/def/ghi 可用 (.*)/(.*)/(.*) 匹配。
([a-zA-Z_]+) 匹配英文单词，允许用-和_连接。
([0-9]+) 匹配多位数字，通常用于匹配ID。
([0-9]) 只匹配一位的数字。

^ 表示正则的开始
$ 表示正则的结束

四、重定向RSS地址到Feedburner

除了可以更改模板里的RSS地址外，.htaccess也能实现RSS地址的更改，并更加方便。

```
1 RewriteEngine on
2 RewriteCond %{HTTP_USER_AGENT} !Feed
3 RewriteCond %{HTTP_USER_AGENT} !Feed
```

```
4 RewriteRule ^feed/?([_0-9a-z-]+)?/?$
```

有了上面的总结，本例其实就很简单了吧。
唯一要注意的是这样操作要确保填写正确的HTTP_USER_AGENT。其实你不常换模板的话。。可能还是直接改模板更省事。

在最后，推荐几个好东东：
在线.htaccess生成器：
htaccessEditor http://www.htaccesseditor.com/sc.shtml
mod_rewrite模块中文参考手册：
http://man.chinaunix.net/newsoft/Apache2.2_chinese_manual/mod/mod_rewrite.html