# **php 之apache rewrite伪静态的简单配置和使用**

原创 2015年01月13日 14:46:09

* 标签：
* [apache](http://so.csdn.net/so/search/s.do?q=apache&t=blog" \t "http://blog.csdn.net/u013537717/article/details/_blank)
* 4343

      1、为什么要使用伪静态：

真正的静态页面空间储存量大，进行删除或者更新这些html文件时可造成大量文件碎片，破坏磁盘坏道，

动态页面虽然可以实时更新，但是有时会导致死循环，对搜索引擎不友好，而伪静态却不会出现这种情况。

伪静态可以更好的缓解服务器的压力，增强搜索引擎对页面的收录;

      2、伪静态可以做什么：

****rewrite 主要的功能就是实现url跳转和隐藏真实地址，可以实现虚拟的静态，虚拟的目录，域名跳转，隐藏后台真实地址等，基于perl语言的正则表达式规范。****

一个普通的动态链接可以通过rewrite 重写为.html  的链接

/article.php?id=23

可以重写为：/article\_23.html

也可重写为目录的形式：/article\_23/

      3、apache开启rewrite重写：

打开apache的配置文件（httpd.conf），搜索****# LoadModule rewrite\_module modules/mod\_rewrite.so****将前面的“#”去掉，这样就可以直接在httpd.conf中写rewrite规则了！

如果想在.htaccess文件中重写规则，可以 在httpd.conf文件中搜索AllowOverride None 修改为AllowOverride All  这样就可以在网站根目录建立.htaccess文件写rewrite规则了！

      4、mod\_rewrite 规则的使用：

****RewriteEngine on   //启动rewrite引擎****

****RewriteCond %{HTTP\_HOST} !^www.test.com  [NC]       //****如果访问网址不是www.test.com   后面的[NC]  N重新从第一条规则开始运行重写过程 C 与下一条相关联

****RewriteRule   ^/(.\*) http://www.**test**.com/ [L]         //****跳转到这个地址

****RewriteEngine on****

****RewriteRule ^/article\_([0-9]\*).html$ /article.php?id=$1      //****将伪静态 地址/article\_XX.html 转换为文件加参数的格式  
****RewriteRule ^/article\_([0-9]\*)/$ /article.php?id=$1 [R]       //****将伪静态 地址/article\_XX/ 转换为文件加参数的格式

      5、mod\_rewrite 规则修正符  修正符 即 Rewrite 语句后面 “[]” 里面的内容

1) R 强制外部重定向

2) F 禁用URL,返回403HTTP状态码。

3) G 强制URL为GONE，返回410HTTP状态码。

4) P 强制使用代理转发。

5) L 表明当前规则是最后一条规则，停止分析以后规则的重写。

6) N 重新从第一条规则开始运行重写过程。

7) C 与下一条规则关联

如果规则匹配则正常处理，以下修正符无效

8) T=MIME-type(force MIME type) 强制MIME类型

9) NS  只用于不是内部子请求

10) NC 不区分大小写

11) QSA 追加请求字符串

12) NE 不在输出转义特殊字符   \%3d$1  等价于 =$1

求不拍砖~~~~~！

<http://blog.csdn.net/u013537717/article/details/42676595>

# **开启apache中的URL重写组件并配置.htaccess实现伪静态**

原创 2015年04月09日 15:33:16

* 标签：
* [伪静态](http://so.csdn.net/so/search/s.do?q=%E4%BC%AA%E9%9D%99%E6%80%81&t=blog" \t "http://blog.csdn.net/hpy1165331898/article/details/_blank) /
* [URL重写](http://so.csdn.net/so/search/s.do?q=URL%E9%87%8D%E5%86%99&t=blog" \t "http://blog.csdn.net/hpy1165331898/article/details/_blank) /
* [htaccess](http://so.csdn.net/so/search/s.do?q=htaccess&t=blog" \t "http://blog.csdn.net/hpy1165331898/article/details/_blank) /
* [虚拟主机](http://so.csdn.net/so/search/s.do?q=%E8%99%9A%E6%8B%9F%E4%B8%BB%E6%9C%BA&t=blog" \t "http://blog.csdn.net/hpy1165331898/article/details/_blank) /
* [Rewrite规则](http://so.csdn.net/so/search/s.do?q=Rewrite%E8%A7%84%E5%88%99&t=blog" \t "http://blog.csdn.net/hpy1165331898/article/details/_blank)
* 3463

****伪静态****的产生主要是为了增强搜索引擎的友好度，方便用户记忆。一般通过正则匹配来重写动态网站的URL使其静态化。那么在Windows平台下如何设置Apache支持URL重写功能，以及如何配置.htaccess文件来实现伪静态呢？

****一、让Apache支持URL重写****

       在httpd.conf中加载****mod\_rewrite.so****组件开启Apache中的URL重写功能。

****二、配置.htaccess文件，定义重写规则****

1、 在配置重写规则之前，需要在httpd.conf中设置AllowOverride All（至少设置为AllowOverride FileInfo）让Apache能够正常的读取.htaccess 文件的内容。

<Directory />

Options FollowSymLinks

AllowOverride FileInfo

</Directory>

* 1
* 2
* 3
* 4

2、在.htaccess中定义重写规则。

RewriteEngine onRewriteCond %{REQUEST\_FILENAME} !-d RewriteCond %{REQUEST\_FILENAME} !-f RewriteRule ^([A-Za-z\_]+)$ index.php?m=$1

* 1
* 2
* 3
* 4

****RewriteEngine on打开rewrite功能****

****RewriteCond指令格式****

* 语法: ****RewriteCond TestString CondPattern [flags]****
* 生效域: server config, virtual host, directory, .htaccess
* ****RewriteCond****指令定义一条规则条件。在一条RewriteRule指令前面可能会有一条或多条RewriteCond指令，只有当自身的模板(pattern)匹配成功且这些条件也满足时规则才被应用于当前URL处理。
* ****TestString****是一个纯文本的字符串，除了包含普通的字符外，还可以包括下列的可扩展结构：
  + $N：RewriteRule后向引用，其中(0 <= N <= 9) 。$N引用紧跟在RewriteCond后面的RewriteRule中模板中的括号中的模板在当前URL中匹配的数据。
  + %N：RewriteCond后向引用，其中(0 <= N <= 9) 。%N引用最后一个RewriteCond的模板中的括号中的模板在当前URL中匹配的数据。
  + ${mapname:key|default}：RewriteMap扩展。

****CondPattern****是条件pattern，即一个应用于当前实例TestString的正则表达式，即TestString将会被计算然后与CondPattern匹配。作为一个标准的扩展正则式，CondPattern有以下补充：

* + 可以在模板串前增加一个!前缀，以用表示不匹配模板。但并不是所有的test都可以加！前缀。
  + CondPattern中可以使用以下特殊变量：
    - ****>CondPattern(大于)**** 将condPattern当作一个普通字符串，将它和TestString进行比较，当TestString 的字符大于CondPattern为真。
    - ****=CondPattern(等于)**** 将condPattern当作一个普通字符串，将它和TestString进行比较，当TestString 与CondPattern完全相同时为真.如果CondPattern只是”“(两个引号紧挨在一起) 此时需TestString 为空字符串方为真。
    - ****-d(是否为目录)**** 将testString当作一个目录名，检查它是否存在以及是否是一个目录。
    - ****-f (是否是regular file)**** 将testString当作一个文件名，检查它是否存在以及是否是一个regular文件。
    - ****-s (是否为长度不为0的regular文件)**** 将testString当作一个文件名，检查它是否存在以及是否是一个长度大于0的regular文件。
    - ****-l (是否为symbolic link)**** 将testString当作一个文件名，检查它是否存在以及是否是一个 symbolic link。
    - ****-F(通过subrequest来检查某文件是否可访问)**** 检查TestString是否是一个合法的文件，而且通过服务器范围内的当前设置的访问控制进行访问。这个检查是通过一个内部subrequest完成的, 因此需要小心使用这个功能以降低服务器的性能。
    - ****-U (通过subrequest来检查某个URL是否存在)**** 检查TestString是否是一个合法的URL，而且通过服务器范围内的当前设置的访问控制进行访问。这个检查是通过一个内部subrequest完成的, 因此需要小心使用这个功能以降低服务器的性能。

****[flags]****是第三个参数，多个标志之间用逗号分隔。

* + ****nocase|NC (不区分大小写)**** 在扩展后的TestString和CondPattern中，比较时不区分文本的大小写。注意，这个标志对文件系统和subrequest检查没有影响。

****ornext|OR(建立与下一个条件的或的关系)**** 默认的情况下，二个条件之间是AND的关系，用这个标志将关系改为OR。例如：

RewriteCond %{REMOTE\_HOST} ^host1.\* [OR] RewriteCond %{REMOTE\_HOST} ^host2.\* [OR] RewriteCond %{REMOTE\_HOST} ^host3.\*RewriteRule …

* + - 1
    - 2
    - 3
    - 4

如果没有[OR]标志，需要写三个条件/规则。

****RewriteRule 指令****

* ****语法: RewriteRule Pattern Substitution [flags]****
* Pattern是一个作用于当前URL的兼容perl的正则表达式. 这里的“当前”是指该规则生效时的URL的值。
* Substitution是，当原始URL与Pattern相匹配时，用以替代(或替换)的字符串此外，Substitution还可以追加特殊标记[flags] 作为RewriteRule指令的第三个参数。

Flags是一个包含以逗号分隔的下列标记

* + ****forbidden|F (强制URL为被禁止的 forbidden)**** 强制当前URL为被禁止的，即，立即反馈一个HTTP响应代码403(被禁止的)。使用这个标记，可以链接若干RewriteConds以有条件地阻塞某些URL。
  + ****gone|G(强制URL为已废弃的 gone)**** 强制当前URL为已废弃的，即，立即反馈一个HTTP响应代码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.
  + ****proxy|P (强制为代理 proxy)**** 此标记使替换成分被内部地强制为代理请求，并立即(重写规则处理立即中断)把处理移交给代理模块。你必须确保此替换串是一个有效的(比如常见的以 [http://hostname](http://hostname/" \t "http://blog.csdn.net/hpy1165331898/article/details/_blank)开头的)能够为Apache代理模块所处理的URI。使用这个标记，可以把某些远程成分映射到本地服务器名称空间，从而增强了ProxyPass指令的功能。(注意: 要使用这个功能，代理模块必须编译在Apache服务器中。如果你不能确定，可以检查“httpd -l”的输出中是否有mod\_proxy.c。如果有，则mod\_rewrite可以使用这个功能；如果没有，则必须启用mod\_proxy并重新编译“httpd”程序。)

****last|L (最后一个规则 last)**** 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。这个标记可以阻止当前已被重写的URL为其后继的规则所重写。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 (重新执行 next round)**** 重新执行重写操作(从第一个规则重新开始). 这时再次进行处理的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，即, 立即回到循环的头部。但是要小心，不要制造死循环！

* + ****chain|C (与下一个规则相链接 chained)**** 此标记使当前规则与下一个(其本身又可以与其后继规则相链接的， 并可以如此反复的)规则相链接。 它产生这样一个效果: 如果一个规则被匹配，通常会继续处理其后继规则， 即，这个标记不起作用；如果规则不能被匹配，则其后继的链接的规则会被忽略。比如，在执行一个外部重定向时， 对一个目录级规则集，你可能需要删除“.www” (此处不应该出现“.www”的)。
  + ****type|T=MIME-type(强制MIME类型type)**** 强制目标文件的MIME类型为MIME-type 比如，它可以用于模拟mod\_alias中的ScriptAlias指令，以内部地强制被映射目录中的所有文件的MIME类型为“application/x-httpd-cgi”。
  + ****nosubreq|NS (仅用于不对内部子请求进行处理 no internal sub-request)**** 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试图搜索可能的目录默认文件(index.xxx)时， Apache会内部地产生子请求。对子请求，它不一定有用的，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。根据你的需要遵循以下原则: 如果你使用了有CGI脚本的URL前缀，以强制它们由CGI脚本处理，而对子请求处理的出错率(或者开销)很高，在这种情况下，可以使用这个标记。
  + ****nocase|NC (忽略大小写 no case)**** 它使Pattern忽略大小写，即, 在Pattern与当前URL匹配时，’A-Z’ 和’a-z’没有区别。
  + ****qsappend|QSA (追加请求串 query string append)**** 此标记强制重写引擎在已有的替换串中追加一个请求串，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。
  + ****noescape|NE (在输出中不对URI作转义 no URI escaping)**** 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(如’%’, ‘$’, ‘;’等)会被转义为等值的十六进制编码。 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，如：   
    RewriteRule /foo/(.\*) /bar?arg=P1\=$1 [R,NE] 可以使’/foo/zed’转向到一个安全的请求’/bar?arg=P1=zed’

****passthrough|PT (移交给下一个处理器 pass through)**** 此标记强制重写引擎将内部结构request\_rec中的uri字段设置为 filename字段的值，它只是一个小修改，使之能对来自其他URI到文件名翻译器的 Alias，ScriptAlias, Redirect 等指令的输出进行后续处理。举一个能说明其含义的例子：如果要通过mod\_rewrite的重写引擎重写/abc为/def，然后通过mod\_alias使/def转变为/ghi，可以这样:

RewriteRule ^/abc(.\*) /def$1 [PT]Alias /def /ghi

* + - 1
    - 2

如果省略了PT标记，虽然mod\_rewrite运作正常，即作为一个使用API的URI到文件名翻译器，它可以重写uri=/abc/…为filename=/def/…，但是，后续的mod\_alias在试图作URI到文件名的翻译时，则会失效。

****skip|S=num (跳过后继的规则 skip)**** 此标记强制重写引擎跳过当前匹配规则后继的num个规则。 它可以实现一个伪if-then-else的构造: 最后一个规则是then从句，而被跳过的skip=N个规则是else从句. (它和’chain|C’标记是不同的!)

* + ****env|E=VAR:VAL (设置环境变量 environment variable)**** 此标记使环境变量VAR的值为VAL, VAL可以包含可扩展的反向引用的正则表达式$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，但通常是在XSSI (via ) or CGI (如 $ENV{’VAR’})中， 也可以在后继的RewriteCond指令的pattern中通过%{ENV:VAR}作引用。使用它可以从URL中剥离并记住一些信息。
  + ****cookie|CO=NAME:VAL:domain[:lifetime[:path]] (设置cookie)****它在客户端浏览器上设置一个cookie。cookie的名称是NAME，其值是VAL。domain字段是该cookie的域，比如’.apache.org’, 可选的lifetime是cookie生命期的分钟数，可选的path是cookie的路径。

       此外，rewrite规则中如果遇到中文，相当有可能会出现乱码问题，因为apache在rewrite时会做一次url解码，这时jk进行请求转发时，就不会再是编码后的字符串了。此种情况，可以在一开始就进行两次编码(encode)，或者在接收请求时先用ISO-8859-1取字节流，再使用UFT-8来new String。

****三、解决网站不在根目录Apache解析路径出错的问题****

若果网站程序不在根目录，那么伪静态的URL就容易出错，解决方法是配置****虚拟主机****。

1、在httpd.conf中加载httpd-vhosts.conf文件

#Virtual hostsInclude conf/extra/httpd-vhosts.conf

* 1
* 2

2、在httpd-vhosts.conf文件中配置虚拟主机

<VirtualHost \*:80>

ServerAdmin webmaster@dummy-host.example.com

DocumentRoot /www/docs/dummy-host.example.com #项目路径

ServerName dummy-host.example.com #ServerName

ErrorLog logs/dummy-host.example.com-error\_log

CustomLog logs/dummy-host.example.com-access\_log common

</VirtualHost>

* 1
* 2
* 3
* 4
* 5
* 6
* 7

3、在****hosts****中加载serverName

       总之，要通过Apache服务器实现伪静态，首先要在httpd.conf中加载mod\_rewrite.so模块，然后配置RewriteRule。当然，在定义RewriteRule之前弄明白网站动态的URL有哪几类，要匹配的静态URL的类型，才能更好更快的写出适合自己网站的RewriteRule。

http://blog.csdn.net/hpy1165331898/article/details/44960867