**URL重写简介**

本篇文档是[mod\_rewrite参考文档](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)的补充，阐述了使用[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)之前必须了解的基本概念。其它文档则作了更加深入的探讨。不过这篇文档对于初学者是一篇很好的入门读物。

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_intro.html#page-header)

**简介**

Apache的[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)是提供了强大URL操作的杀手级模块，可以实现几乎所有你梦想的URL操作，其代价是你必须接受其复杂性，因为[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)对于初学者的主要障碍就是不容易理解和运用，即使是Apache专家有时也会发掘出[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)的新用途。换句话说：你或者是打退堂鼓永不再用，或者是喜欢它并一生受用。目前存在这样一种倾向：许多初学者只是把URL重写规则当着是会变戏法的魔咒，而并未在使用中真正理解这些规则的含义。

本篇文档试图给出充分的背景知识，以便于初学者随后的理解，而不是盲目的复制和粘贴。

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_intro.html#page-header)

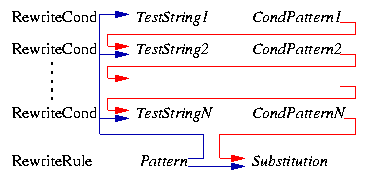
**正则表达式**

[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)使用的是[Perl兼容的正则表达式](http://pcre.org/)语法。本文不打算详细讲解正则表达式语法，你可以到[PCRE man page](http://pcre.org/pcre.txt), [Perl regular expression man page](http://www.perldoc.com/perl5.8.0/pod/perlre.html), [Mastering Regular Expressions, by Jeffrey Friedl](http://www.oreilly.com/catalog/regex2/index.html)获得这些内容。

[RewriteRule](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html#rewriterule)指令的说明部分有一个简单的正则表达式语法简介，可以去参考一下。  
另外需要说明的是可以在表达式的最前面加上一个感叹号('!')表示**不匹配**，不过这种用法并不符合正则表达式语法。

**正则表达式的反向引用能力**

这是很重要的一点：一旦在*Pattern*或者*CondPattern*中使用了圆括号，就会建立内部的反向引用，可以使用$N和%N来调用(见下述)，并且在*Substitution*和*TestString*中都有效。图-2说明了反向引用被转换和展开的位置。

  
*图-2:* The back-reference flow through a rule.

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_intro.html#page-header)

**RewriteRule basics**

(此部分文档尚未完成) Basic anatomy of a RewriteRule, with exhaustively annotated simple examples.

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_intro.html#page-header)

**Rewrite Flags**

(此部分文档尚未完成) Discussion of the flags to RewriteRule, and when and why one might use them.

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_intro.html#page-header)

**Rewrite conditions**

(此部分文档尚未完成) Discussion of RewriteCond, looping, and other related concepts.

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_intro.html#page-header)

**Rewrite maps**

(此部分文档尚未完成) Discussion of RewriteMap, including simple, but heavily annotated, examples.

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_intro.html#page-header)

**.htaccess files**

(此部分文档尚未完成) Discussion of the differences between rewrite rules in httpd.conf and in .htaccess files.

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_intro.html#page-header)

**环境变量**

此模块会跟踪两个额外的(非标准)CGI/SSI环境变量：SCRIPT\_URL和SCRIPT\_URI。他们包含了当前资源的*逻辑*网络视图，而标准CGI/SSI变量SCRIPT\_NAME和SCRIPT\_FILENAME包含的是*物理*系统视图。

注意：这些变量保持的是*其最初被请求时的*URI/URL，即在任何重写操作*之前*的URI/URL。其重要性在于他们是重写操作重写URL到物理路径名的原始依据。

**示例**

SCRIPT\_NAME=/sw/lib/w3s/tree/global/u/rse/.www/index.html

SCRIPT\_FILENAME=/u/rse/.www/index.html

SCRIPT\_URL=/u/rse/

SCRIPT\_URI=http://en1.engelschall.com/u/rse/

**URL重写技术细节**

本文探讨一些[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)和URL匹配的技术细节。

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_tech.html#page-header)

**内部处理**

此模块的内部处理极为复杂，但是为了使一般用户避免犯低级错误，也让管理员能充分利用其功能，在此仍然做一下说明。

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_tech.html#page-header)

**API阶段**

首先，你必须了解Apache是分若干阶段来处理HTTP请求的。Apache API对每个阶段都提供了一个hook程序。[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)使用两个hook程序：其一，从URL到文件名的转换hook(用在读取HTTP请求之后、授权开始之前)； 其二，修正hook(用在授权阶段和读取目录级配置(.htaccess)之后、内容处理器激活之前)。

所以，Apache收到一个请求并且确定了响应主机(或虚拟主机)之后，重写引擎即开始处理服务器级配置中的所有[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)指令(此时处于从URL到文件名转换的阶段)，此阶段完成后，最终的数据目录便确定了。接下来进入修正程序段并触发目录级配置中的[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)指令。这两个阶段并不是泾渭分明的，但都实施了把URL重写成新的URL或者文件名。虽然API最初不是为此目的而设计的，但是现在它已经成为了API的一种用途。记住以下两点，会有助于更好地理解：

1. 虽然[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)可以将URL重写为新的URL或文件名，甚至将文件名重写为新的文件名，但是之前的API只提供从URL到文件名的hook。在Apache 2.0中，增加了两个丢失的hook以使得处理过程更加清晰。不过这样做并没有给用户带来麻烦，用户只需记住这样一个事实：借助从URL到文件名的hook比最初API设计的目标功能更强大。
2. 令人难以置信的是，[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)还提供了目录级的URL操作(.htaccess文件)，而这些文件必须在将URL转换成文件名之后才会被处理(这是必须的，因为.htaccess存在于文件系统中)。换句话说，根据API阶段，这时再处理任何URL操作已经太晚了。为了解决这个"鸡和蛋"的问题，[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)使用了一个小技巧：在进行一个目录级的URL/文件名操作时，先把文件名重写回相应的URL(通常这个操作是不可行的，但是参考下面的RewriteBase指令就能明白它是怎么实现的了)，然后，对这个新的URL建立一个新的内部的子请求，再重新开始API阶段的执行。

另外，[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)尽力使这些复杂的操作对用户透明。但仍须记住：服务器级的URL操作速度快而且效率高，而目录级的操作由于这个"鸡和蛋"的问题速度较慢而且效率也低。但从另一个侧面看，这却是[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)得以为一般用户提供(局部限制的)URL操作的唯一方法。

牢记这两点！

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_tech.html#page-header)

**规则集的处理**

当[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)在这两个API阶段中开始执行时，它会读取配置结构中配置好的 (或者是在服务启动时建立的服务器级的，或者是在遍历目录采集到的目录级的)规则集，然后，启动URL重写引擎来处理(带有一个或多个条件的)规则集。无论是服务器级的还是目录级的规则集，都是由同一个URL重写引擎处理，只是最终结果处理不同而已。

规则集中规则的顺序是很重要的，因为重写引擎是按一种特殊的顺序处理的：逐个遍历每个规则([RewriteRule](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html#rewriterule)指令)，如果出现一个匹配条件的规则，则可能回头遍历已有的规则条件(RewriteCond指令)。由于历史的原因，条件规则是前置的，所以控制流程略显冗长，细节见图-1。

**初级URL重写指南**

本文是[mod\_rewrite参考文档](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)的补充材料。阐述在实际应用中如何解决网管所面临的基于URL的典型问题，并详细描述了如何配置URL重写规则集以解决这些问题。

注意：根据你的服务器配置，有可能必须对这里的例子作些小修改，比如，在额外启用[mod\_alias](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_alias.html)和[mod\_userdir](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_userdir.html)的情况下要增加[PT]标志，或者为了适应目录级(.htaccess)的配置而将针对服务器级的规则集进行重写。对一个特定的规则集应该先透彻理解然后再考虑应用，这样才能避免出现问题。

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide.html#page-header)

**规范化URL**

**描述：**

在有些web服务器上，一个资源会拥有多个URL。在实际应用和发布中应该使用的是规范的URL，其他的则是简写或者只在内部使用。无论用户在请求中使用什么形式的URL，最终看见的都应该是规范的URL。

**解决方案：**

对所有不规范的URL执行一个外部HTTP重定向，以改变它在浏览器地址栏中的显示及其后继请求。下例中的规则集用规范的/u/user替换/~user，并修正了/u/user所遗漏的后缀斜杠。

RewriteRule ^/**~**([^/]+)/?(.\*) /**u**/$1/$2 [**R**]

RewriteRule ^/u/(**[^/]+**)$ /$1/$2**/** [**R**]

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide.html#page-header)

**规范化主机名**

**描述：**

这个规则的目的是强制使用特定的主机名以代替其他名字。比如，你想强制使用**www.example.com**代替**example.com**，就可以在以下方案的基础上进行修改：

**解决方案：**

对运行在非80端口的站点

RewriteCond %{HTTP\_HOST} !^fully\.qualified\.domain\.name [NC]

RewriteCond %{HTTP\_HOST} !^$

RewriteCond %{SERVER\_PORT} !^80$

RewriteRule ^/(.\*) http://fully.qualified.domain.name:%{SERVER\_PORT}/$1 [L,R]

对运行在80端口的站点

RewriteCond %{HTTP\_HOST} !^fully\.qualified\.domain\.name [NC]

RewriteCond %{HTTP\_HOST} !^$

RewriteRule ^/(.\*) http://fully.qualified.domain.name/$1 [L,R]

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide.html#page-header)

**移动过的DocumentRoot**

**描述：**

通常，web服务器的[DocumentRoot](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/core.html#documentroot)直接对应于URL"/"，但是它常常不是处于最高的一级。比如，你希望访问者在进入网站时首先进入/about/目录。可以使用下面给出的规则集。

**解决方案：**

只需将"/"重定向到"/about/"即可：

RewriteEngine on

RewriteRule **^/$** /about/ [**R**]

也可以使用[RedirectMatch](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_alias.html#redirectmatch)指令解决问题：

RedirectMatch ^/$ http://example.com/e/www/

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide.html#page-header)

**结尾斜杠问题**

**描述：**

每个网管对引用目录的结尾斜杠问题都有一本苦经，如果遗漏了，服务器会产生一个错误，因为如果请求是"/~quux/foo"而不是"/~quux/foo/"，服务器就会去找一个叫foo的文件，而它是一个目录，所以就报错了。通常，可以使用这个[FAQ entry](http://httpd.apache.org/docs/1.3/misc/FAQ-E.html#set-servername)里面提到的方法解决问题。但是有时候需要使用重写规则来解决问题，比如，在应用了许多复杂的重写规则之后。

**解决方案：**

解决这个微妙问题的方案是让服务器自动添加后缀斜杠。为了达到目的，必须使用一个外部重定向，以使浏览器能够正确地处理后继的请求(比如对图片的请求)。如果仅仅执行一个内部重写，可能仅仅对目录页面有效，而对含有相对URL的图片的页面无效，因为浏览器有请求内嵌目标的可能。比如，如果不用外部重定向，对/~quux/foo/index.html页面中的image.gif的请求将变成对/~quux/image.gif的请求！所以，应该这样写：

RewriteEngine on

RewriteBase /~quux/

RewriteRule ^foo**$** foo**/** [**R**]

又懒又疯狂的做法是把这些写入其宿主目录中的顶级.htaccess中：

RewriteEngine on

RewriteBase /~quux/

RewriteCond %{REQUEST\_FILENAME} **-d**

RewriteRule ^(.+**[^/]**)$ $1**/** [R]

但是这样一来会增加处理上的开销。

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide.html#page-header)

**将用户主目录移动到不同web服务器**

**描述：**

通常，许多网管在建立一个新的web服务器时，都会有这样的要求：重定向一个web服务器上的所有用户主目录到另一个web服务器。

**解决方案：**

很简单，在老的web服务器上重定向所有的"/~user/anypath"到"http://newserver/~user/anypath"：

RewriteEngine on

RewriteRule ^/~(.+) http://**newserver**/~$1 [R,L]

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide.html#page-header)

**在多个目录中搜索页面**

**描述：**

有时会有必要使web服务器在多个目录中搜索页面，对此，MultiViews或者其他技术无能为力。

**解决方案：**

编制一个明确的规则集以搜索目录中的文件：

RewriteEngine on

# 首先尝试在 **dir1** 中寻找，找到即停

RewriteCond /your/docroot/**dir1**/%{REQUEST\_FILENAME} -f

RewriteRule ^(.+) /your/docroot/**dir1**/$1 [L]

# 然后尝试在 **dir2** 中寻找，找到即停

RewriteCond /your/docroot/**dir2**/%{REQUEST\_FILENAME} -f

RewriteRule ^(.+) /your/docroot/**dir2**/$1 [L]

# 再找不到就继续寻找其他的 Alias 或 ScriptAlias 目录

RewriteRule ^(.+) - [PT]

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide.html#page-header)

**按照URL的片段设置环境变量**

**描述：**

希望保持请求之间的状态信息，又不希望使用CGI来包装所有页面，只是通过分离URL中的有用信息来做到。

**解决方案：**

可以用一个规则集来分离出状态信息，并设置环境变量以备此后用于XSSI或CGI。这样，一个"/foo/S=java/bar/"的URL会被解析为"/foo/bar/"，而环境变量STATUS则被设置为"java"。

RewriteEngine on

RewriteRule ^(.\*)/**S=([^/]+)**/(.\*) $1/$3 [E=**STATUS:$2**]

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide.html#page-header)

**虚拟用户主机**

**描述：**

如果需要为用户username支持一个www.**username**.host.domain.com的主页，但不在此机器上建虚拟主机，而是仅用在此机器上增加一个DNS A记录的方法实现。

**解决方案：**

仅能对包含"Host: "头的HTTP/1.1请求实现。可以使用以下规则集内部地将http://www.username.host.com/anypath重写为/home/username/anypath

RewriteEngine on

RewriteCond %{**HTTP\_HOST**} ^www\.**[^.]+**\.host\.com$

RewriteRule ^(.+) %{HTTP\_HOST}$1 [C]

RewriteRule ^www\.**([^.]+)**\.host\.com(.\*) /home/**$1**$2

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide.html#page-header)

**为外来访问者重定向用户主目录**

**描述：**

对不是来自本地域ourdomain.com的外来访问者的请求，重定向其用户主目录URL到另一个web服务器www.somewhere.com，有时这种做法也会用在虚拟主机的配置段中。

**解决方案：**

只须一个重写条件：

RewriteEngine on

RewriteCond %{REMOTE\_HOST} **!^.+\.ourdomain\.com$**

RewriteRule ^(/~.+) http://www.somewhere.com/$1 [R,L]

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide.html#page-header)

**重定向锚**

**描述：**

默认情况下，重定向到一个HTML锚是不可行的，因为'#'会被转义为'%23'。This, in turn, breaks the redirection.

**解决方案：**

在RewriteRule指令中使用[NE]标志(不转义)。

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide.html#page-header)

**依赖于时间的重写**

**描述：**

在页面内容需要按时间的不同而变化的场合，比如重定向特定页面等，许多网管仍然采用CGI脚本的方法，如何用[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)来实现呢？

**解决方案：**

有许多名为TIME\_xxx的变量可以用在重写条件中，联合使用词典模式的"<STRING", "=STRING", ">STRING"比较，就可以实现依赖于时间的重写：

RewriteEngine on

RewriteCond %{TIME\_HOUR}%{TIME\_MIN} >0700

RewriteCond %{TIME\_HOUR}%{TIME\_MIN} <1900

RewriteRule ^foo\.html$ foo.day.html

RewriteRule ^foo\.html$ foo.night.html

此例使foo.html在07:00-19:00时间内指向foo.day.html，而在其余时间指向foo.night.html，对主页是一个不错的功能...

**对YYYY转变为XXXX的向前兼容**

**描述：**

在转变了大批document.YYYY文件为document.XXXX后(比如.html→.phtml)，如何保持URL的向前兼容(仍然虚拟地存在)？

**解决方案：**

只须按基准文件名重写，并测试带有新的扩展名的文件是否存在，如果存在则用新的，否则仍然用原来的。

# 将document.html重写为document.phtml的向后兼容的规则集

# 当且仅当document.phtml存在且document.html不存在的时候

RewriteEngine on

RewriteBase /~quux/

# 剪切并记住basename

RewriteRule ^(.\*)\.html$ $1 [C,E=WasHTML:yes]

# 如果存在的话就重写为document.phtml

RewriteCond %{REQUEST\_FILENAME}.phtml -f

RewriteRule ^(.\*)$ $1.phtml [S=1]

# 否则返回先前的basename

RewriteCond %{ENV:WasHTML} ^yes$

RewriteRule ^(.\*)$ $1.html

**内容处理**

**从旧到新(内部)**

**描述：**

假定已经把文件foo.html改名为bar.html，需要对老的URL向后兼容，即让用户仍然可以使用老的URL，而感觉不到文件被改名了。

**解决方案：**

通过以下规则内部地将老的URL重写为新的：

RewriteEngine on

RewriteBase /~quux/

RewriteRule ^**foo**\.html$ **bar**.html

**从旧到新(外部)**

**描述：**

仍然假定已经把文件foo.html改名为bar.html，需要对老的URL向后兼容，但是要让用户得到文件被改名的暗示，即浏览器的地址栏中显示的是新的URL。

**解决方案：**

作一个HTTP的强制重定向以改变浏览器和用户界面上的显示：

RewriteEngine on

RewriteBase /~quux/

RewriteRule ^**foo**\.html$ **bar**.html [**R**]

**从静态到动态**

**描述：**

如何无缝转换静态页面foo.html为动态的foo.cgi，而不为浏览器/用户所察觉。

**解决方案：**

只须重写此URL为CGI-script，并强制作为CGI-script运行的MIME类型。比如对/~quux/foo.html的请求会执行/~quux/foo.cgi 。

RewriteEngine on

RewriteBase /~quux/

RewriteRule ^foo\.**html**$ foo.**cgi** [T=**application/x-httpd-cgi**]

**访问控制**

**阻止Robot**

**描述：**

如何阻止一个完全匿名的Robot取得特定网络区域的页面？/robots.txt文件可以包含若干"Robot排除协议"行，但不足以阻止此类Robot。

**解决方案：**

可以用一个规则集以拒绝对网络区域/~quux/foo/arc/(对一个很深的目录区域进行列表可能会使服务器产生很大的负载)的访问。还必须确保仅阻止特定的Robot，也就是说，仅仅阻止Robot访问主机是不够的(这样会同时阻止用户访问该主机)。为此，就需要对HTTP头的User-Agent信息作匹配。

RewriteCond %{HTTP\_USER\_AGENT} ^**NameOfBadRobot**.\*

RewriteCond %{REMOTE\_ADDR} ^**123\.45\.67\.[8-9]**$

RewriteRule ^**/~quux/foo/arc/**.+ - [**F**]

**阻止内嵌的图片**

**描述：**

假设http://www.quux-corp.de/~quux/有一些内嵌GIF图片的页面，这些图片很好，所以就有人盗链到他们自己的页面中了。由于这样徒然增加了我们服务器的流量，因此，我们不愿意这种事情发生。

**解决方案：**

虽然，我们不能100%地保护这些图片不被写入别人的页面，但至少可以对发出HTTP Referer头的浏览器加以限制。

RewriteCond %{HTTP\_REFERER} **!^$**

RewriteCond %{HTTP\_REFERER} !^http://www.quux-corp.de/~quux/.\*$ [NC]

RewriteRule **.\*\.gif$** - [F]

RewriteCond %{HTTP\_REFERER} !^$

RewriteCond %{HTTP\_REFERER} !.\*/foo-with-gif\.html$

RewriteRule **^inlined-in-foo\.gif$** - [F]

**拒绝代理**

**描述：**

如何拒绝某个主机或者来自特定主机的用户使用Apache代理？

**解决方案：**

首先，要确保在配置文件中[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)位于[mod\_proxy](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_proxy.html)之后！使它在[mod\_proxy](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_proxy.html)*之前*被调用。然后，使用如下方法拒绝某个主机：

RewriteCond %{REMOTE\_HOST} **^badhost\.mydomain\.com$**

RewriteRule !^http://[^/.]\.mydomain.com.\* - [F]

使用如下方法拒绝user@host-dependent用户：

RewriteCond %{REMOTE\_IDENT}@%{REMOTE\_HOST} **^badguy@badhost\.mydomain\.com$**

RewriteRule !^http://[^/.]\.mydomain.com.\* - [F]

**其它**

**外部重写引擎**

**描述：**

如何解决似乎无法用[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)解决的FOO/BAR/QUUX/之类的问题？

**解决方案：**

可以使用一个与[RewriteMap](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html#rewritemap)功能相同的外部[RewriteMap](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html#rewritemap)程序，一旦它在Apache启动时被执行，则从STDIN接收被请求的URL ，并将处理过(通常是重写过的)的URL(以相同顺序)在STDOUT输出。

RewriteEngine on

RewriteMap quux-map **prg:**/path/to/map.quux.pl

RewriteRule ^/~quux/(.\*)$ /~quux/**${quux-map:$1}**

#!/path/to/perl

# 禁止使用会导致Apache陷入死循环的I/O缓冲

$| = 1;

# 从stdin读取URL(每行一个)，并在stdout输出替换URL

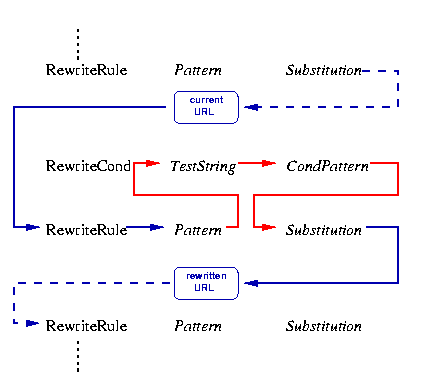
while (<>) {

s|^foo/|bar/|;

print $\_;

}

这是只是一个简单的示例，只是把所有的/~quux/foo/...重写为/~quux/bar/...而已。但事实上，可以把它修改成任何你想要的输出。但是要注意，虽然一般用户都可以**使用**，可是只有系统管理员才可以**定义**这样的映射。

  
*图-1:*重写规则集中的控制流

可见，URL首先与每个规则的*Pattern*匹配，如果匹配失败，[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)将立即终止此规则的处理，继而处理下一个规则。如果匹配成功，[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)将寻找相应的规则条件，如果一个条件都没有，则简单地用*Substitution*构造的新值来替换URL，然后继续处理其他规则；但是如果条件存在，则开始一个内部循环按其列出的顺序逐个处理。对规则条件的处理有所不同：URL并不与模式进行匹配，而是首先通过扩展变量、反向引用、查找映射表等步骤建立一个*TestString*字符串，然后用它来与*CondPattern*匹配。如果匹配失败，则整个条件集和对应的规则失败；如果匹配成功，则执行下一个规则直到所有条件执行完毕。如果所有条件得以匹配，则以*Substitution*替换URL，并且继续处理。

**高级URL重写指南**

本文是[mod\_rewrite参考文档](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)的补充材料。阐述在实际应用中如何解决网管所面临的基于URL的典型问题，并详细描述了如何配置URL重写规则集以解决这些问题。

注意：根据你的服务器配置，有可能必须对这里的例子作些小修改，比如，在额外启用[mod\_alias](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_alias.html)和[mod\_userdir](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_userdir.html)的情况下要增加[PT]标志，或者为了适应目录级(.htaccess)的配置而将针对服务器级的规则集进行重写。对一个特定的规则集应该先透彻理解然后再考虑应用，这样才能避免出现问题。

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide_advanced.html#page-header)

**集群网站的同类URL规划**

**描述：**

我们希望在一个Intranet集群网站中，对所有WWW服务器建立一致的URL规划，也就是说，所有的URL(针对每个服务器进行本地配置，因此是独立于各个服务器的)实际上都是*独立于*各个服务器的！我们需要的是一个具有独立于各个服务器的一致性的WWW域名空间，即URL不需要包含物理目标服务器，而由集群本身来自动定位物理目标主机。

**解决方案：**

首先，目标服务器的信息来自于(分布式的)包含有用户、组以及实体的外部映射表，其格式形如下：

user1 server\_of\_user1

user2 server\_of\_user2

: :

这些信息被存入map.xxx-to-host文件。其次，如果URL在一个服务器上无效，我们还需要引导所有的服务器将

/u/user/anypath

/g/group/anypath

/e/entity/anypath

重定向到

http://physical-host/u/user/anypath

http://physical-host/g/group/anypath

http://physical-host/e/entity/anypath

以下规则集依靠映射文件来完成这个操作(如果一个用户在映射表中没有对应的项，则使用server0做为默认服务器)：

RewriteEngine on

RewriteMap user-to-host txt:/path/to/map.user-to-host

RewriteMap group-to-host txt:/path/to/map.group-to-host

RewriteMap entity-to-host txt:/path/to/map.entity-to-host

RewriteRule ^/u/**([^/]+)**/?(.\*) http://**${user-to-host:$1|server0}**/u/$1/$2

RewriteRule ^/g/**([^/]+)**/?(.\*) http://**${group-to-host:$1|server0}**/g/$1/$2

RewriteRule ^/e/**([^/]+)**/?(.\*) http://**${entity-to-host:$1|server0}**/e/$1/$2

RewriteRule ^/([uge])/([^/]+)/?$ /$1/$2/.www/

RewriteRule ^/([uge])/([^/]+)/([^.]+.+) /$1/$2/.www/$3\

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide_advanced.html#page-header)

**结构化的用户主目录**

**描述：**

一些拥有几千个用户的网站通常都使用结构化的用户主目录规划，即每个用户主目录位于一个带有特定前缀(比如其用户名的第一个字符)的子目录下：/~foo/anypath代表/home/**f**/foo/.www/anypath，而/~bar/anypath代表/home/**b**/bar/.www/anypath

**解决方案：**

可以使用下列规则集来扩展'~'以达到上述目的。

RewriteEngine on

RewriteRule ^/~(**([a-z])**[a-z0-9]+)(.\*) /home/**$2**/$1/.www$3

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide_advanced.html#page-header)

**文件系统的重组**

**描述：**

这是一个不加雕琢的例子：一个大量使用目录级规则集以实现平滑的观感，并且从来不用调整数据结构的杀手级的应用。背景：从1992年开始，***net.sw***存放了我收集的免费Unix软件包。它是我的爱好也是我的工作，因为在学习计算机科学的同时，业余时间还做了多年的系统和网络管理员。每周我都需要整理软件，因而建立了一个层次很深的目录结构来存放各种软件包：

drwxrwxr-x 2 netsw users 512 Aug 3 18:39 Audio/

drwxrwxr-x 2 netsw users 512 Jul 9 14:37 Benchmark/

drwxrwxr-x 12 netsw users 512 Jul 9 00:34 Crypto/

drwxrwxr-x 5 netsw users 512 Jul 9 00:41 Database/

drwxrwxr-x 4 netsw users 512 Jul 30 19:25 Dicts/

drwxrwxr-x 10 netsw users 512 Jul 9 01:54 Graphic/

drwxrwxr-x 5 netsw users 512 Jul 9 01:58 Hackers/

drwxrwxr-x 8 netsw users 512 Jul 9 03:19 InfoSys/

drwxrwxr-x 3 netsw users 512 Jul 9 03:21 Math/

drwxrwxr-x 3 netsw users 512 Jul 9 03:24 Misc/

drwxrwxr-x 9 netsw users 512 Aug 1 16:33 Network/

drwxrwxr-x 2 netsw users 512 Jul 9 05:53 Office/

drwxrwxr-x 7 netsw users 512 Jul 9 09:24 SoftEng/

drwxrwxr-x 7 netsw users 512 Jul 9 12:17 System/

drwxrwxr-x 12 netsw users 512 Aug 3 20:15 Typesetting/

drwxrwxr-x 10 netsw users 512 Jul 9 14:08 X11/

1996年7月，我决定通过一个漂亮的Web接口公开我的收藏。"漂亮"是指提供一个直接浏览整个目录结构的接口，同时不对这个结构做任何改变，甚至也不在结构顶部放置CGI脚本。为什么呢？因为这个结构还要能够被FTP访问，而且我不希望其中有任何Web或者CGI成分。

**解决方案：**

这个方案分为两个部分：第一个部分是一组CGI脚本，用于实时建立所有目录层次的页面。我把它们放在/e/netsw/.www/中：

-rw-r--r-- 1 netsw users 1318 Aug 1 18:10 .wwwacl

drwxr-xr-x 18 netsw users 512 Aug 5 15:51 DATA/

-rw-rw-rw- 1 netsw users 372982 Aug 5 16:35 LOGFILE

-rw-r--r-- 1 netsw users 659 Aug 4 09:27 TODO

-rw-r--r-- 1 netsw users 5697 Aug 1 18:01 netsw-about.html

-rwxr-xr-x 1 netsw users 579 Aug 2 10:33 netsw-access.pl

-rwxr-xr-x 1 netsw users 1532 Aug 1 17:35 netsw-changes.cgi

-rwxr-xr-x 1 netsw users 2866 Aug 5 14:49 netsw-home.cgi

drwxr-xr-x 2 netsw users 512 Jul 8 23:47 netsw-img/

-rwxr-xr-x 1 netsw users 24050 Aug 5 15:49 netsw-lsdir.cgi

-rwxr-xr-x 1 netsw users 1589 Aug 3 18:43 netsw-search.cgi

-rwxr-xr-x 1 netsw users 1885 Aug 1 17:41 netsw-tree.cgi

-rw-r--r-- 1 netsw users 234 Jul 30 16:35 netsw-unlimit.lst

其中的DATA/子目录包含了上述目录结构，即实际的***net.sw***原始内容，由rdist在需要的时候自动更新。第二个部分的问题是：如何将这两个结构连接为一个观感平滑的URL树？我希望在为各种URL运行对应的CGI脚本的时候，用户感觉不到DATA/目录的存在。方案如下：首先，我把下列配置放在针对每个目录的配置文件里，将公布的URL"/net.sw/"重写为内部路径"/e/netsw"：

RewriteRule ^net.sw$ net.sw/ [R]

RewriteRule ^net.sw/(.\*)$ e/netsw/$1

第一条规则是针对遗漏后缀斜杠的请求的！第二条规则才是真正实现功能的。接着，就是放在目录级配置文件/e/netsw/.www/.wwwacl中的杀手级的配置了：

Options ExecCGI FollowSymLinks Includes MultiViews

RewriteEngine on

# 通过"/net.sw/"前缀到达

RewriteBase /net.sw/

# 首先将根目录重写到cgi处理脚本

RewriteRule ^$ netsw-home.cgi [L]

RewriteRule ^index\.html$ netsw-home.cgi [L]

# 当浏览器请求perdir页面时剥去子目录

RewriteRule ^.+/(netsw-[^/]+/.+)$ $1 [L]

# 现在打断对本地文件的重写

RewriteRule ^netsw-home\.cgi.\* - [L]

RewriteRule ^netsw-changes\.cgi.\* - [L]

RewriteRule ^netsw-search\.cgi.\* - [L]

RewriteRule ^netsw-tree\.cgi$ - [L]

RewriteRule ^netsw-about\.html$ - [L]

RewriteRule ^netsw-img/.\*$ - [L]

# 任何别的东西都是一个由另一个cgi脚本处理的子目录

RewriteRule !^netsw-lsdir\.cgi.\* - [C]

RewriteRule (.\*) netsw-lsdir.cgi/$1

阅读提示：

1. 注意前半部分中的L(最后)标志和非替换部分('-')
2. 注意后半部分中的!(非)符号和C(链)标志
3. 注意最后一条规则是全匹配模式

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide_advanced.html#page-header)

**将失败的URL重定向到其他web服务器**

**描述：**

一个常见的问题是如何将对web服务器A的失败请求重定向到服务器B。一般，可以使用借助[ErrorDocument](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/core.html#errordocument)的CGI脚本来解决，此外，还有使用[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)的方案。但是须注意，这种方法的执行效率不如使用[ErrorDocument](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/core.html#errordocument)的CGI脚本！

**解决方案：**

第一种方案，有最好的性能而灵活性欠佳，出错概率小所以安全：

RewriteEngine on

RewriteCond /your/docroot/%{REQUEST\_FILENAME} **!-f**

RewriteRule ^(.+) http://**webserverB**.dom/$1

但是问题在于，它只对位于[DocumentRoot](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/core.html#documentroot)中的页面有效。虽然可以增加更多的条件(比如同时还处理用户主目录，等等)，但是还有一个更好的方法：

RewriteEngine on

RewriteCond %{REQUEST\_URI} **!-U**

RewriteRule ^(.+) http://**webserverB**.dom/$1

这种方法使用了[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)提供的"前瞻"(look-ahead)的功能，是一种对所有URL类型都有效而且安全的方法，但是对web服务器的性能有不利影响。如果web服务器有一个强大的CPU，那就用这个方法。而在慢速机器上，可以用第一种方法，或者用性能更好的CGI脚本。

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide_advanced.html#page-header)

**文档访问的多路复用**

**描述：**

你知道<http://www.perl.com/CPAN>的CPAN(综合Perl存档网络)吗？它实现了一个重定向以提供全世界的CPAN镜像中离访问者最近的FTP站点，也可以称之为FTP访问多路复用服务。CPAN是通过CGI脚本实现的，那么用[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)如何实现呢？

**解决方案：**

首先，[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)从3.0.0版本开始可以重写"ftp:"类型。其次，可以用[RewriteMap](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html#rewritemap)取得对客户端顶级域名的最短路径。利用链式规则集，并用顶级域名作为查找多路复用映射表的键，可以这样做：

RewriteEngine on

RewriteMap multiplex txt:/path/to/map.cxan

RewriteRule ^/CxAN/(.\*) %{REMOTE\_HOST}::$1 [C]

RewriteRule ^.+\.**([a-zA-Z]+)**::(.\*)$ ${multiplex:**$1**|ftp.default.dom}$2 [R,L]

##

## map.cxan -- CxAN多路映射表

##

de ftp://ftp.cxan.de/CxAN/

uk ftp://ftp.cxan.uk/CxAN/

com ftp://ftp.cxan.com/CxAN/

:

##EOF##

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide_advanced.html#page-header)

**内容处理**

**依赖于浏览器的内容**

**描述：**

有时候有必须提供依赖于浏览器的最佳内容(至少对重要的顶级页面)，即对最新的Netscape提供最大化的版本，对Lynx提供最小化的版本，而对其他的浏览器则提供一个一般的版本。

**解决方案：**

对此，内容协商无能为力，因为浏览器不提供那种形式的类型，所以只能在"User-Agent"头上想办法。以下规则集可以完成这个操作：如果"User-Agent"以"Mozilla/3"开头，则将foo.html重写为foo.NS.html，并终止重写操作；如果是"Lynx"或者版本号为1和2的"Mozilla"，则重写为foo.20.html；而对其他所有浏览器则是foo.32.html。

RewriteCond %{HTTP\_USER\_AGENT} ^**Mozilla/3**.\*

RewriteRule ^foo\.html$ foo.**NS**.html [**L**]

RewriteCond %{HTTP\_USER\_AGENT} ^**Lynx/**.\* [OR]

RewriteCond %{HTTP\_USER\_AGENT} ^**Mozilla/[12]**.\*

RewriteRule ^foo\.html$ foo.**20**.html [**L**]

RewriteRule ^foo\.html$ foo.**32**.html [**L**]

**动态镜像**

**描述：**

假定，需要在我们的域名空间里加入其他远程主机的页面。对FTP服务器，可以用mirror程序在本地机器上维持一个对远程数据的最新的拷贝；对HTTP服务器，可以使用webcopy程序。但这两种技术都有一个主要的缺点：本地拷贝必须通过这个程序来更新。所以，比较好的方法是不采用静态镜像，而采用动态镜像，即在有数据请求时自动更新(远程主机上更新的数据)。

**解决方案：**

为达到此目的，可以使用*代理吞吐*(Proxy Throughput)功能([P]标志)，将远程页面甚至整个远程网络区域映射到我们的域名空间：

RewriteEngine on

RewriteBase /~quux/

RewriteRule ^**hotsheet/**(.\*)$ **http://www.tstimpreso.com/hotsheet/**$1 [**P**]

RewriteEngine on

RewriteBase /~quux/

RewriteRule ^**usa-news\.html**$ **http://www.quux-corp.com/news/index.html** [**P**]

**反向动态镜像**

**描述：**

...

**解决方案：**

RewriteEngine on

RewriteCond /mirror/of/remotesite/$1 -U

RewriteRule ^http://www\.remotesite\.com/(.\*)$ /mirror/of/remotesite/$1

**通过Intranet取得丢失的数据**

**描述：**

这在受防火墙保护的内部网(www2.quux-corp.dom)上保存和维护实际数据，而在Internet上虚拟地运行web服务器(www.quux-corp.dom)。方法是外部服务器在空闲时间从内部服务器取得被请求的数据。

**解决方案：**

首先，必须确保防火墙对内部服务器的保护，并只允许此外部服务器获取数据。对包过滤(packet-filtering)防火墙，可以制定如下防火墙规则：

**ALLOW** Host www.quux-corp.dom Port >1024 --> Host www2.quux-corp.dom Port **80**

**DENY** Host \* Port \* --> Host www2.quux-corp.dom Port **80**

请按你的实际情况，对上例稍作调整。接着，建立通过代理后台获取丢失数据的重写规则：

RewriteRule ^/~([^/]+)/?(.\*) /home/$1/.www/$2

RewriteCond %{REQUEST\_FILENAME} **!-f**

RewriteCond %{REQUEST\_FILENAME} **!-d**

RewriteRule ^/home/([^/]+)/.www/?(.\*) http://**www2**.quux-corp.dom/~$1/pub/$2 [**P**]

**负载均衡**

**描述：**

如何将www.foo.com的负载均衡到www[0-5].foo.com(一共是6个服务器)?

**解决方案：**

这个问题有许多可能的解决方案，在此，我们讨论通称为"基于DNS"的方案，和特殊的使用[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)的方案

1. **DNS循环**

最简单的方法是用BIND的DNS循环特性，只要按惯例设置www[0-9].foo.com的DNS的A(地址)记录，如：

www0 IN A 1.2.3.1

www1 IN A 1.2.3.2

www2 IN A 1.2.3.3

www3 IN A 1.2.3.4

www4 IN A 1.2.3.5

www5 IN A 1.2.3.6

然后，增加以下各项：

www IN CNAME www0.foo.com.

IN CNAME www1.foo.com.

IN CNAME www2.foo.com.

IN CNAME www3.foo.com.

IN CNAME www4.foo.com.

IN CNAME www5.foo.com.

IN CNAME www6.foo.com.

注意，上述看起来似乎是错误的，但事实上，它的确是BIND中一个特意的特性，而且也可以这样用。在解析www.foo.com时，BIND将给出www0-www6的结果(虽然每次在次序上会有轻微的置换/循环)，客户端的请求可以被分散到各个服务器。但这并不是一个优秀的负载均衡方案，因为DNS解析信息可以被网络中其他域名服务器缓冲，一旦www.foo.com被解析为wwwN.foo.com，该请求的所有后继请求都将被送往wwwN.foo.com。但是最终结果是正确的，因为请求的总量的确被分散到各个服务器了。

1. **DNS负载均衡**

一种成熟的基于DNS的负载均衡方法是使用[lbnamed](http://www.stanford.edu/~schemers/docs/lbnamed/lbnamed.html)程序，它是一个Perl程序，并带有若干辅助工具，实现了真正的基于DNS的负载均衡。

1. **代理吞吐循环(Proxy Throughput Round-Robin)**

这是一个使用[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)以及代理吞吐特性的方法。首先，在DNS记录中将www0.foo.com固定为www.foo.com，如下：

www IN CNAME www0.foo.com.

其次，将www0.foo.com转换为一个专职代理服务器，即由这个机器把所有到来的URL通过内部代理分散到另外5个服务器(www1-www5)。为此，必须建立一个规则集，对所有URL调用一个负载均衡脚本lb.pl 。

RewriteEngine on

RewriteMap lb prg:/path/to/lb.pl

RewriteRule ^/(.+)$ ${lb:$1} [P,L]

下面是lb.pl的内容：

#!/path/to/perl

## lb.pl -- 负载平衡脚本

$| = 1;

$name = "www"; # the hostname base

$first = 1; # the first server (not 0 here, because 0 is myself)

$last = 5; # the last server in the round-robin

$domain = "foo.dom"; # the domainname

$cnt = 0;

while (<STDIN>) {

$cnt = (($cnt+1) % ($last+1-$first));

$server = sprintf("%s%d.%s", $name, $cnt+$first, $domain);

print "http://$server/$\_";

}

##EOF##

最后的说明：这样有用吗？似乎www0.foo.com也会超载呀？答案是：没错，它的确会超载，但是它超载的仅仅是简单的代理吞吐请求！所有诸如SSI、CGI、ePerl等的处理完全是由其他机器完成的，这个才是重点。

1. **硬件/TCP循环**

还有一个硬件解决方案。Cisco有一个叫LocalDirector的东西，实现了TCP/IP层的负载均衡，事实上，它是一个位于网站集群前端的电路级网关。如果你有足够资金而且的确需要高性能的解决方案，那么可以用这个。

**新的MIME类型，新的服务**

**描述：**

网上有许多很巧妙的CGI程序，但是用法晦涩，许多网管弃之不用。即使是Apache的MEME类型的动作处理器，也仅仅在CGI程序不需要在其输入中包含PATH\_INFO和QUERY\_STRINGS时才很好用。首先，配置一种新的后缀为.scgi的(安全CGI)文件类型，其处理器是很常见的cgiwrap程序。问题是：如果使用同类URL规划(见上述)，而用户宿主目录中的一个文件的URL是/u/user/foo/bar.scgi，可是cgiwrap要求的URL的格式是/~user/foo/bar.scgi/，以下重写规则解决了这个问题：

RewriteRule ^/[uge]/**([^/]+)**/\.www/(.+)\.scgi(.\*) ...

... /internal/cgi/user/cgiwrap/~**$1**/$2.scgi$3 [NS,**T=application/x-http-cgi**]

另外，假设还需要使用其他程序：wwwlog(显示access.log中的一个URL子树)和wwwidx(对一个URL子树运行Glimpse)，则必须为两个程序提供URL区域作为其操作对象。比如，对/u/user/foo/执行swwidx程序的超链是这样的：

/internal/cgi/user/swwidx?i=/u/user/foo/

其缺点是，必须**同时**硬编码超链中的区域**和**CGI的路径，如果重组了这个区域，就需要花费大量时间来修改各个超链。

**解决方案：**

用一个特殊的新的URL格式，自动拼装CGI参数：

RewriteRule ^/([uge])/([^/]+)(/?.\*)/\\* /internal/cgi/user/wwwidx?i=/$1/$2$3/

RewriteRule ^/([uge])/([^/]+)(/?.\*):log /internal/cgi/user/wwwlog?f=/$1/$2$3

现在，这个搜索/u/user/foo/的超链简化成了：

HREF="\*"

它会在内部被自动转换为

/internal/cgi/user/wwwidx?i=/u/user/foo/

这样就可以为使用":log"的超链拼装出调用CGI程序的参数。

**传输过程中的内容协商**

**描述：**

这是一个很深奥的功能：动态地生成但静态地发送(从文件系统中读出，然后直接发出去)，但是如果它丢失了，则由服务器动态生成。这样，可以静态地提供CGI生成的页面，除非有人(或者是一个计划任务)删除了这些静态页面，而且其内容可以得到更新。

**解决方案：**

以下规则集实现了这个功能：

RewriteCond %{REQUEST\_FILENAME} **!-s**

RewriteRule ^page\.**html**$ page.**cgi** [T=application/x-httpd-cgi,L]

这样，如果page.html不存在或者文件大小为null ，则对page.html的请求会导致page.cgi的运行。其中奥妙在于page.cgi是一个将输出写入到page.html的(同时也写入STDOUT)的CGI脚本。执行完毕，服务器则将page.html的内容发送出去。如果网管需要强制更新其内容，只须删除page.html即可(通常由一个计划任务完成)。

**自动更新的文档**

**描述：**

建立一个复杂的页面，能够在用编辑器写了一个更新的版本时自动在浏览器上得到刷新，这不是很好吗？这可能吗？

**解决方案：**

这是可行的！这需要综合利用MIME多成分、web服务器的NPH和[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)的URL操控特性。首先，建立一个新的URL特性：对在文件系统中需要刷新的所有URL加上":refresh" 。

RewriteRule ^(/[uge]/[^/]+/?.\*):refresh /internal/cgi/apache/nph-refresh?f=$1

现在当我们引用如下URL

/u/foo/bar/page.html:refresh

时，将导致在内部调用

/internal/cgi/apache/nph-refresh?f=/u/foo/bar/page.html

接着就是NPH-CGI脚本部分了。虽然，人们常说"将此作为一个练习留给读者"，但我还是给出答案了。

#!/sw/bin/perl

##

## nph-refresh -- 用于自动刷新页面的 NPH/CGI 脚本

## Copyright (c) 1997 Ralf S. Engelschall, All Rights Reserved.

##

$| = 1;

# 分解 QUERY\_STRING 变量

@pairs = split(/&/, $ENV{'QUERY\_STRING'});

foreach $pair (@pairs) {

($name, $value) = split(/=/, $pair);

$name =~ tr/A-Z/a-z/;

$name = 'QS\_' . $name;

$value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;

eval "\$$name = \"$value\"";

}

$QS\_s = 1 if ($QS\_s eq ");

$QS\_n = 3600 if ($QS\_n eq ");

if ($QS\_f eq ") {

print "HTTP/1.0 200 OK\n";

print "Content-type: text/html\n\n";

print "&lt;b&gt;ERROR&lt;/b&gt;: No file given\n";

exit(0);

}

if (! -f $QS\_f) {

print "HTTP/1.0 200 OK\n";

print "Content-type: text/html\n\n";

print "&lt;b&gt;ERROR&lt;/b&gt;: File $QS\_f not found\n";

exit(0);

}

sub print\_http\_headers\_multipart\_begin {

print "HTTP/1.0 200 OK\n";

$bound = "ThisRandomString12345";

print "Content-type: multipart/x-mixed-replace;boundary=$bound\n";

&print\_http\_headers\_multipart\_next;

}

sub print\_http\_headers\_multipart\_next {

print "\n--$bound\n";

}

sub print\_http\_headers\_multipart\_end {

print "\n--$bound--\n";

}

sub displayhtml {

local($buffer) = @\_;

$len = length($buffer);

print "Content-type: text/html\n";

print "Content-length: $len\n\n";

print $buffer;

}

sub readfile {

local($file) = @\_;

local(\*FP, $size, $buffer, $bytes);

($x, $x, $x, $x, $x, $x, $x, $size) = stat($file);

$size = sprintf("%d", $size);

open(FP, "&lt;$file");

$bytes = sysread(FP, $buffer, $size);

close(FP);

return $buffer;

}

$buffer = &readfile($QS\_f);

&print\_http\_headers\_multipart\_begin;

&displayhtml($buffer);

sub mystat {

local($file) = $\_[0];

local($time);

($x, $x, $x, $x, $x, $x, $x, $x, $x, $mtime) = stat($file);

return $mtime;

}

$mtimeL = &mystat($QS\_f);

$mtime = $mtime;

for ($n = 0; $n &lt; $QS\_n; $n++) {

while (1) {

$mtime = &mystat($QS\_f);

if ($mtime ne $mtimeL) {

$mtimeL = $mtime;

sleep(2);

$buffer = &readfile($QS\_f);

&print\_http\_headers\_multipart\_next;

&displayhtml($buffer);

sleep(5);

$mtimeL = &mystat($QS\_f);

last;

}

sleep($QS\_s);

}

}

&print\_http\_headers\_multipart\_end;

exit(0);

##EOF##

**海量虚拟主机**

**描述：**

Apache的[<VirtualHost>](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/core.html#virtualhost)功能很强，在有几十个虚拟主机的情况下运行得仍然很好，但是如果你是ISP，需要提供成百上千个虚拟主机，那么这就不是最佳选择了。

**解决方案：**

为此，需要用*代理吞吐*(Proxy Throughput)功能([P]标志)映射远程页面甚至整个远程网络区域到自己的域名空间：

##

## vhost.map

##

www.vhost1.dom:80 /path/to/docroot/vhost1

www.vhost2.dom:80 /path/to/docroot/vhost2

:

www.vhostN.dom:80 /path/to/docroot/vhostN

##

## httpd.conf

##

:

# 在重定向时使用规范化的主机名等等

UseCanonicalName on

:

# 在CLF-format之前添加虚拟主机

CustomLog /path/to/access\_log "%{VHOST}e %h %l %u %t \"%r\" %>s %b"

:

# 为主服务器启用重写引擎

RewriteEngine on

# define two maps: one for fixing the URL and one which defines

# the available virtual hosts with their corresponding

# DocumentRoot.

RewriteMap lowercase int:tolower

RewriteMap vhost txt:/path/to/vhost.map

# Now do the actual virtual host mapping

# via a huge and complicated single rule:

#

# 1. make sure we don't map for common locations

RewriteCond %{REQUEST\_URI} !^/commonurl1/.\*

RewriteCond %{REQUEST\_URI} !^/commonurl2/.\*

:

RewriteCond %{REQUEST\_URI} !^/commonurlN/.\*

#

# 2. make sure we have a Host header, because

# currently our approach only supports

# virtual hosting through this header

RewriteCond %{HTTP\_HOST} !^$

#

# 3. lowercase the hostname

RewriteCond ${lowercase:%{HTTP\_HOST}|NONE} ^(.+)$

#

# 4. lookup this hostname in vhost.map and

# remember it only when it is a path

# (and not "NONE" from above)

RewriteCond ${vhost:%1} ^(/.\*)$

#

# 5. finally we can map the URL to its docroot location

# and remember the virtual host for logging puposes

RewriteRule ^/(.\*)$ %1/$1 [E=VHOST:${lowercase:%{HTTP\_HOST}}]

:

[top](http://lamp.linux.gov.cn/Apache/ApacheMenu/rewrite/rewrite_guide_advanced.html#page-header)

**访问控制**

**对主机的拒绝**

**描述：**

如何禁止一批外部列表中的主机对我们服务器的访问？

**解决方案：**

RewriteEngine on

RewriteMap hosts-deny txt:/path/to/hosts.deny

RewriteCond ${hosts-deny:%{REMOTE\_HOST}|NOT-FOUND} !=NOT-FOUND [OR]

RewriteCond ${hosts-deny:%{REMOTE\_ADDR}|NOT-FOUND} !=NOT-FOUND

RewriteRule ^/.\* - [F]

##

## hosts.deny

##

## 注意! 这是一个映射而不是列表(即使我们这样看待它)。

## mod\_rewrite会将它作为 键/值 对进行解析。

## 所以每一项至少要存在一个伪值："-"

##

193.102.180.41 -

bsdti1.sdm.de -

192.76.162.40 -

**对代理的拒绝**

**描述：**

如何拒绝某个主机或者来自特定主机的用户使用Apache代理？

**解决方案：**

首先，要确保Apache配置文件中[mod\_rewrite](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_rewrite.html)在[mod\_proxy](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_proxy.html)的下面！使它在[mod\_proxy](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_proxy.html)*之前*被调用。然后使用如下方法拒绝某个主机：

RewriteCond %{REMOTE\_HOST} **^badhost\.mydomain\.com$**

RewriteRule !^http://[^/.]\.mydomain.com.\* - [F]

使用如下方法拒绝user@host-dependent用户：

RewriteCond %{REMOTE\_IDENT}@%{REMOTE\_HOST} **^badguy@badhost\.mydomain\.com$**

RewriteRule !^http://[^/.]\.mydomain.com.\* - [F]

**特殊的认证**

**描述：**

有时候，会需要一种非常特殊的认证，即对一组明确指定的用户，允许其访问，且没有(在使用[mod\_auth\_basic](http://lamp.linux.gov.cn/Apache/ApacheMenu/mod/mod_auth_basic.html)的基本认证方法时可能会出现的)任何提示。

**解决方案：**

使用一个重写条件列表来拒绝朋友以外的所有人：

RewriteCond %{REMOTE\_IDENT}@%{REMOTE\_HOST} **!^friend1@client1.quux-corp\.com$**

RewriteCond %{REMOTE\_IDENT}@%{REMOTE\_HOST} **!^friend2**@client2.quux-corp\.com$

RewriteCond %{REMOTE\_IDENT}@%{REMOTE\_HOST} **!^friend3**@client3.quux-corp\.com$

RewriteRule ^/~quux/only-for-friends/ - [F]

**基于Referer的转向器**

**描述：**

如何配置一个基于HTTP头"Referer"的转向器以转向到任意数量的引用页(referring page)？

**解决方案：**

使用下面这个很巧妙的规则集：

RewriteMap deflector txt:/path/to/deflector.map

RewriteCond %{HTTP\_REFERER} !=""

RewriteCond ${deflector:%{HTTP\_REFERER}} ^-$

RewriteRule ^.\* %{HTTP\_REFERER} [R,L]

RewriteCond %{HTTP\_REFERER} !=""

RewriteCond ${deflector:%{HTTP\_REFERER}|NOT-FOUND} !=NOT-FOUND

RewriteRule ^.\* ${deflector:%{HTTP\_REFERER}} [R,L]

并结合相应的重写映射表：

##

## deflector.map

##

http://www.badguys.com/bad/index.html -

http://www.badguys.com/bad/index2.html -

http://www.badguys.com/bad/index3.html http://somewhere.com/

它可以自动将请求(在映射表中指定了"-"值的时候)重定向回其引用页(referring page)，或者(在映射表中URL有第二个参数时)重定向到一个特定的URL。

本篇译者：[金步国](http://lamp.linux.gov.cn/jinbuguo_florilegium.html)([其他作品](http://lamp.linux.gov.cn/jinbuguo_florilegium.html))