

## 第 7 章

# Nginx 的 Rewrite 规则与实例

## 7.1 什么是 Nginx 的 Rewrite 规则

Rewrite 主要的功能就是实现 URL 的重写, Nginx 的 Rewrite 规则采用 PCRE (Perl Compatible Regular Expressions) Perl 兼容正则表达式的语法进行规则匹配, 如果您需要 Nginx 的 Rewrite 功能, 在编译 Nginx 之前, 须要编译安装 PCRE 库。

正则表达式 (英文: Regular Expression) 在计算机科学中, 是指一个用来描述或匹配一系列符合某个句法规则的字符串的单个字符串。最初的正则表达式出现于理论计算机科学的自动控制理论和形式化语言理论中。在这些领域中有对计算 (自动控制) 的模型和对形式化语言描述与分类的研究。20 世纪 40 年代, Warren McCulloch 与 Walter Pitts 将神经系统中的神经元描述成小而简单的自动控制元。20 世纪 50 年代, 数学家斯蒂芬·科尔·克莱尼利用称之为正则集合的数学符号来描述此模型。肯·汤普逊将此符号系统引入编辑器 QED, 然后是 Unix 上的编辑器 ed, 并最终引入 grep。自此, 正则表达式被广泛使用于各种 Unix 或类似 Unix 的工具, 例如 Perl。

通过 Rewrite 规则, 可以实现规范的 URL、根据变量来做 URL 转向及选择配置。例如, 一些使用 MVC 框架的程序只有一个入口, 可以通过 Rewrite 来实现。一些动态 URL 地址须要伪装成静态 HTML, 便于搜索引擎抓取, 也需要 Rewrite 来处理。一些由于目录结构、域名变化的旧 URL, 须要跳转到新的 URL 上, 也可以通过 Rewrite 来处理。

## 7.2 Nginx Rewrite 规则相关指令

Nginx Rewrite 规则相关指令有 if、rewrite、set、return、break 等，其中 rewrite 是最关键的指令。一个简单的 Nginx Rewrite 规则语法如下：

```
rewrite ^/b/(.*)\.html /play.php?video=$1 break;
```

如果加上 if 语句，示例如下：

```
if (!-f $request_filename)
{
    rewrite ^/img/(.*)$ /site/$host/images/$1 last;
}
```

### 7.2.1 break 指令

语法：break

默认值：none

使用环境：server, location, if

该指令的作用是完成当前的规则集，不再处理 rewrite 指令。

示例如下：

```
if ($slow) {
    limit_rate 10k;
    break;
}
```

### 7.2.2 if 指令

语法：if (condition) { ... }

默认值：none

使用环境：server, location

该指令用于检查一个条件是否符合，如果条件符合，则执行大括号内的语句。if 指令不支持嵌套，不支持多个条件&&和||处理。

以下信息可以被指定为条件：

(1) 变量名，错误的值包括：空字符串""，或者任何以 0 开始的字符串；

- (2) 变量比较可以使用 “=”（表示等于）和 “!=”（表示不等于）运算符；
- (3) 正则表达式模式匹配可以使用 “~\*” 和 “~” 符号；
- (4) “~” 符号表示区分大小写字母的匹配；
- (5) “~\*” 符号表示不区分大小写字母的匹配（例如 firefox 与 FireFox 是匹配的）；
- (6) “!~” 和 “!~\*” 符号的作用刚好和 “~”、“~\*” 相反，表示不匹配；
- (7) “-f” 和 “!-f” 用来判断文件是否存在；
- (8) “-d” 和 “!-d” 用来判断目录是否存在；
- (9) “-e” 和 “!-e” 用来判断文件或目录是否存在；
- (10) “-x” 和 “!-x” 用来判断文件是否可执行。

部分正则表达式可以在圆括号 “()” 内，其值可以通过后面的变量 \$1 至 \$9 访问，示例如代码 7-1 所示：

代码 7-1

```
if ($http_user_agent ~ MSIE) {
    rewrite ^(.*)$ /msie/$1 break;
}

if ($http_cookie ~* "id=([^;] +)(?:;|$)" ) {
    set $id $1;
}

if ($request_method = POST ) {
    return 405;
}

if (!-f $request_filename) {
    break;
    proxy_pass http://127.0.0.1;
}

if ($slow) {
    limit_rate 10k;
}

if ($invalid_referer) {
    return 403;
}

if ($args ^~ post=140){
    rewrite ^ http://example.com/permanent;
}
```

示例代码 7-1 中，内置的变量 `$invalid_referer` 值由 `valid_referers` 指令提供。



### 7.2.3 return 指令

语法: `return code`

默认值: `none`

使用环境: `server, location, if`

该指令用于结束规则的执行并返回状态码给客户端。状态码可以使用这些值: 204, 400, 402~406, 408, 410, 411, 413, 416 及 500~504。此外，非标准状态码 444 将以不发送任何 Header 头的方式结束连接。

示例，如果访问的 URL 以 “.Sh” 和 “.Bash” 结尾，则返回状态码 403:

```
location ~ /\. (sh|bash)?$
{
    return 403;
}
```

下面，我们来详细介绍 `return` 指令支持的状态码。

#### 204 No Content

服务器成功处理了请求，但无须返回任何实体内容，并且希望返回更新了的元信息。响应可能通过实体头部的形式，返回新的或更新后的元信息。如果存在这些头部信息，则应当与所请求的变量相呼应。

如果客户端是浏览器，那么用户浏览器应保留发送了该请求的页面，而不产生任何文档视图上的变化，即使按照规范新的或更新后的元信息，也应当被应用到用户浏览器活动视图中的文档。

#### 400 Bad Request

由于包含语法错误，当前请求无法被服务器理解。除非进行修改，否则客户端不应该重复提交这个请求。

#### 402 Payment Required

该状态码是为了将来可能的需求而预留的。

#### 403 Forbidden

服务器已经理解请求，但是拒绝执行它。与 401 响应不同的是，身份验证并不能提供任何帮助，而且这个请求也不应该被重复提交。如果这不是一个 `HEAD` 请求，而且服务器希望能够讲清楚为何请求不能被执行，就应该在实体内描述拒绝的原因。当然服务器也可以返回一个 404 响

应，假如它不希望让客户端获得任何信息。

#### 404 Not Found

请求失败，请求所希望得到的资源未在服务器上发现。没有信息能够告诉用户这个状况到底是暂时的还是永久的。假如服务器知道情况，应当使用 410 状态码来告知旧资源因为某些内部的配置机制问题，已经永久地不可用，而且没有任何可以跳转的地址。404 这个状态码被广泛应用于当服务器不想揭示为何请求被拒绝，或者没有其他适合的响应可用的情况下。

#### 405 Method Not Allowed

请求行中指定的请求方法不能被用于请求相应的资源。该响应必须返回一个 **Allow** 头信息，用以表示出当前资源能够接受的请求方法的列表。

鉴于 **PUT**，**DELETE** 方法会对服务器上的资源进行写操作，因而绝大部分的网页服务器都不支持或在默认配置下不支持上述请求方法，对于此类请求均会返回 405 错误。

#### 406 Not Acceptable

请求的资源的内容特性无法满足请求头中的条件，因而无法生成响应实体。

除非这是一个 **HEAD** 请求，否则该响应就应当返回一个包含可以让用户或浏览器从中选择最合适的实体特性及地址列表的实体。实体的格式由 **Content-Type** 头中定义的媒体类型决定。浏览器可以根据格式及自身能力自行作出最佳选择。但是，规范中并没有定义任何作出此类自动选择的标准。

#### 408 Request Timeout

请求超时。客户端没有在服务器预备等待的时间内完成一个请求的发送。客户端可以随时再次提交这一请求而无须进行任何更改。

#### 410 Gone

被请求的资源在服务器上已经不再可用，而且没有任何已知的转发地址。这样的状况应当被认为是永久性的。如果可能，拥有链接编辑功能的客户端应当在获得用户许可后删除所有指向这个地址的引用。如果服务器不知道或无法确定这个状况是否是永久的，就应该使用 404 状态码。除非额外说明，否则这个响应是可缓存的。

410 响应的目的主要是帮助网站管理员维护网站，通知用户该资源已经不再可用，并且服务器拥有者希望所有指向这个资源的远端连接也被删除。这类事件在限时、增值服务中很普遍。同样，410 响应也被用于通知客户端在当前服务器站点上，原本属于某个个人的资源已经不再可用。当然，是否要把所有永久不可用的资源标记为 '410 Gone'，以及是否要保持此标记多长时间，完全取决于服务器拥有者。

#### 411 Length Required

服务器拒绝在没有定义 `Content-Length` 头的情况下接受请求。在添加了表明请求消息体长度的有效 `Content-Length` 头之后，客户端可以再次提交该请求。

#### 413 Request Entity Too Large

服务器拒绝处理当前请求，因为该请求提交的实体数据大小超过了服务器愿意或能够处理的范围。此种情况下，服务器可以关闭连接以免客户端继续发送此请求。

如果这个状况是临时的，服务器应当返回一个 `Retry-After` 的响应头，以告知客户端可以在多少时间以后重新尝试。

#### 416 Requested Range Not Satisfiable

如果请求中包含了 `Range` 请求头，并且 `Range` 中指定的任何数据范围都与当前资源的可用范围不重合，同时请求中又没有定义 `If-Range` 请求头，那么服务器就应当返回 416 状态码。

假如 `Range` 使用的是字节范围，那么这种情况就是指请求指定的所有数据范围的首字节位置都超过了当前资源的长度。服务器也应当在返回 416 状态码的同时，包含一个 `Content-Range` 实体头，用以指明当前资源的长度。这个响应也被禁止使用 `multipart/byteranges` 作为其 `Content-Type`。

#### 500 Internal Server Error

服务器遇到了一个未曾预料的状态，导致了它无法完成对请求的处理。一般来说，这个问题都会在服务器的程序码出错时出现。

#### 501 Not Implemented

服务器不支持当前请求所需要的某个功能。当服务器无法识别请求的方法，并且无法支持其对任何资源的请求时。

#### 502 Bad Gateway

作为网关或代理工作的服务器尝试执行请求时，从上游服务器接收到无效的响应。

#### 503 Service Unavailable

由于临时的服务器维护或过载，服务器当前无法处理请求。这个状况是临时的，并且将在一段时间以后恢复。如果能够预计延迟时间，那么响应中可以包含一个 `Retry-After` 头用以标明这个延迟时间。如果没有给出这个 `Retry-After` 信息，那么客户端应当以处理 500 响应的方式处理它。

注意：503 状态码的存在并不意味着服务器在过载的时候必须使用它。某些服务器只不过是希望拒绝客户端的连接。

### 504 Gateway Timeout

作为网关或代理工作的服务器尝试执行请求时，未能及时从上游服务器（URI 标识出的服务器，例如 HTTP、FTP、LDAP）或辅助服务器（例如 DNS）收到响应。

注意：某些代理服务器在 DNS 查询超时时会返回 400 或 500 错误。

## 7.2.4 rewrite 指令

语法：rewrite regex replacement flag

默认值：none

使用环境：server, location, if

该指令根据表达式来重定向 URI，或者修改字符串。指令根据配置文件中的顺序来执行。

注意重写表达式只对相对路径有效。如果你想配对主机名，你应该使用 if 语句，代码如下：

```
if ($host ~* www\.(.*)) {  
    set $host_without_www $1;  
    rewrite ^(.*)$ http://$host_without_www$1 permanent; # $1 contains '/foo', not  
    'www.mydomain.com/foo'  
}
```

如果替换串以 `http://` 开头，将会采用 301 或 302 跳转进行 URL 重定向。

rewrite 指令的最后一项参数为 flag 标记，支持的 flag 标记有：

- last——相当于 Apache 里的 [L] 标记，表示完成 rewrite；
- break——本条规则匹配完成后，终止匹配，不再匹配后面的规则；
- redirect——返回 302 临时重定向，浏览器地址栏会显示跳转后的 URL 地址；
- permanent——返回 301 永久重定向，浏览器地址栏会显示跳转后的 URL 地址。

在以上的标记中，last 和 break 用来实现 URI 重写，浏览器地址栏的 URL 地址不变，但在服务器端访问的路径发生了变化。redirect 和 permanent 用来实现 URL 跳转，浏览器地址栏会显示跳转后的 URL 地址。

last 和 break 标记的实现功能类似，但二者之间有细微的差别，使用 alias 指令时必须用 last 标记，使用 proxy\_pass 指令时要使用 break 标记。last 标记在本条 rewrite 规则执行完毕后，会对其所在的 server{.....} 标签重新发起请求，而 break 标记则在本条规则匹配完成后，终止匹配，不再匹配后面的规则。例如以下这段规则，就必须使用 break 标记，使用 last 标记会导致死循环：



```
location /cms/ {
    proxy_pass http://test.yourdomain.com;
    rewrite "^/cms/(.*)\.html$" /cms/index.html break;
}
```

因此，一般在根 location 中（即 location /{.....}）或直接在 server 标签中编写 rewrite 规则，推荐使用 last 标记，在非根 location 中（例如 location /cms/{.....}），则使用 break 标记。例如：

```
rewrite ^(/download/(.*)/media/(.*)\..*$ $1/mp3/$2.mp3 last;
rewrite ^(/download/(.*)/audio/(.*)\..*$ $1/mp3/$2.ra last;
return 403;
```

```
location /download/ {
    rewrite ^(/download/(.*)/media/(.*)\..*$ $1/mp3/$2.mp3 break;
    rewrite ^(/download/(.*)/audio/(.*)\..*$ $1/mp3/$2.ra break;
    return 403;
}
```

如果被替换的 URI 中含有参数（即类似/app/test.php?id=5 之类的 URI），默认情况下参数会被自动附加到替换串上，你可以通过在替换串的末尾加上?标记来解决这一问题。

```
rewrite ^/users/(.*)$ /show?user=$1? last;
```

下面我们来比较一下，不加?标记和加上?标记的 URL 跳转区别：

```
rewrite ^/test/(.*)$ http://www.yourdomain.com/home permanent;
```

访问 <http://www.yourdomain.com/test?id=5> 经过 301 跳转后的 URL 地址为 <http://www.yourdomain.com/home?id=5>

```
rewrite ^/test/(.*)$ http://www.yourdomain.com/home? permanent;
```

访问 <http://www.yourdomain.com/test?id=5> 经过 301 跳转后的 URL 地址为 <http://www.yourdomain.com/home>

注：对花括号（{ 和 }）来说，它们既能用在重定向的正则表达式里，也能用在配置文件里分割代码块，为了避免冲突，正则表达式里如果带花括号，应该用双引号（或者单引号）包围。比如，要将类似以下的 URL：

</photos/123456>

重定向到：

</path/to/photos/12/1234/123456.png>

可以用以下方法（注意双引号）：

```
rewrite "/photos/([0-9]{2})([0-9]{2})([0-9]{2})"
/path/to/photos/$1/$1$2/$1$2$3.png;
```



### 7.2.5 set 指令

语法: set variable value

默认值: none

使用环境: server, location, if

该指令用于定义一个变量，并给变量赋值。变量的值可以为文本、变量及文本变量的联合。

示例如下:

```
set $varname 'hello';
```

### 7.2.6 uninitialized\_variable\_warn 指令

语法: uninitialized\_variable\_warn on|off

默认值: uninitialized\_variable\_warn on

使用环境: http, server, location, if

该指令用于开启或关闭记录关于未初始化变量的警告信息，默认值为开启。

### 7.2.7 Nginx Rewrite 可以用到的全局变量

在 if、location、rewrite 指令中，可以使用以下全局变量:

- \$args
- \$content\_length
- \$content\_type
- \$document\_root
- \$document\_uri
- \$host
- \$http\_user\_agent
- \$http\_cookie
- \$limit\_rate
- \$request\_body\_file
- \$request\_method
- \$remote\_addr

- \$remote\_port
- \$remote\_user
- \$request\_filename
- \$request\_uri
- \$query\_string
- \$scheme
- \$server\_protocol
- \$server\_addr
- \$server\_name
- \$server\_port
- \$uri

## 7.3 PCRE 正则表达式语法

Perl 正则表达式源自于 Henry Spencer 写的 regex，它已经演化成了 PCRE（Perl 兼容正则表达式，Perl Compatible Regular Expressions），一个由 Philip Hazel 开发的，为很多现代工具所使用的库。在 Nginx 的 rewrite 指令的语法中，“rewrite regex replacement flag”之中的 regex 使用的是 PCRE 正则表达式。表 7-1 是在 PCRE 中元字符及其在正则表达式上下文中行为的一个完整列表。

表 7-1 PCRE 正则表达式语法一览表

字符	描述
\	将下一个字符标记为一个特殊字符，或一个原义字符，或一个向后引用，或一个八进制转义符。例如，“\n”匹配一个换行符。序列“\\”匹配“\”而“\"则匹配“（”
^	匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^也匹配“\n”或“\r”之后的位置
\$	匹配输入字符串的结束位置。如果设置了 RegExp 对象的 Multiline 属性，\$也匹配“\n”或“\r”之前的位置
*	匹配前面的子表达式零次或多次。例如，zo*能匹配“z”及“zoo”。*等价于{0,}
+	匹配前面的子表达式一次或多次。例如，“zo+”能匹配“zo”及“zoo”，但不能匹配“z”。+等价于{1,}
?	匹配前面的子表达式零次或一次。例如，“do(es)?”可以匹配“do”或“does”中的“do”。?等价于{0,1}
?	当该字符紧跟在任何一个其他限制符(*,+,?,{n},{n,},{n,m})后面时，匹配模式是非贪婪的。非贪婪模式尽可能少地匹配所搜索的字符串，而默认的贪婪模式则尽可能多地匹配所搜索的字符串。例如，对于字符串“oooo”，“o+?”将匹配单个“o”，而“o+”将匹配所有“o”

续表

字符	描述
{n}	n 是一个非负整数。匹配确定的 n 次。例如，“o{2}”不能匹配“Bob”中的“o”，但是能匹配“food”中的两个 o
{n,}	n 是一个非负整数。至少匹配 n 次。例如，“o{2,}”不能匹配“Bob”中的“o”，但能匹配“foooooo”中的所有 o。“o{1,}”等价于“o+”。“o{0,}”则等价于“o*”
{n,m}	m 和 n 均为非负整数，其中 n<=m。最少匹配 n 次且最多匹配 m 次。例如，“o{1,3}”将匹配“foooooo”中的前三个 o。“o{0,1}”等价于“o?”。请注意在逗号和两个数之间不能有空格
.	匹配除“\n”之外的任何单个字符。要匹配包括“\n”在内的任何字符，请使用像“[\n]”的模式
(pattern)	匹配 pattern 并获取这一匹配。所获取的匹配可以从产生的 Matches 集合得到，在 VBScript 中使用 SubMatches 集合，在 JScript 中则使用 \$0...\$9 属性。要匹配圆括号字符，请使用 “\(” 或 “\)”
(?:pattern)	匹配 pattern 但不获取匹配结果，也就是说这是一个非获取匹配，不进行存储供以后使用。这在使用“或”字符( )来组合一个模式的各个部分是很有用的。例如:'industr(?:lies)'就是一个比'industryindustries'更简略的表达式
(?=pattern)	正向预查，在任何匹配 pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配无须获取供以后使用。例如，“Windows(?:95 98 NT 2000)”能匹配“Windows2000”中的“Windows”，但不能匹配“Windows3.1”中的“Windows”。预查不消耗字符，也就是说，在一个匹配发生后，在本次匹配搜索的最后一次匹配，而不是从包含预查的字符之后开始
(?!pattern)	负向预查，在任何不匹配 pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配无须获取供以后使用。例如“Windows(?:!95 !98 !NT !2000)”能匹配“Windows3.1”中的“Windows”，但不能匹配“Windows2000”中的“Windows”。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始
x y	匹配 x 或 y。例如，“z food”能匹配“z”或“food”。“(z f)ood”则匹配“zood”或“food”
[xyz]	字符集合。匹配所包含的任意一个字符。例如，“[abc]”可以匹配“plain”中的“a”
[^xyz]	负值字符集合。匹配未包含的任意字符。例如，“[^abc]”可以匹配“plain”中的“p”
[a-z]	字符范围。匹配指定范围内的任意字符。例如，“[a-z]”可以匹配“a”到“z”范围内的任意小写字母字符
[^a-z]	负值字符范围。匹配任何不在指定范围内的任意字符。例如，“[^a-z]”可以匹配任何不在“a”到“z”范围内的任意字符
\b	匹配一个单词边界，也就是指单词和空格间的位置。例如，“er\b”可以匹配“never”中的“er”，但不能匹配“verb”中的“er”

续表

字符	描述
\B	匹配非单词边界。“er\b”能匹配“verb”中的“er”，但不能匹配“never”中的“er”
\cx	匹配由 x 指明的控制字符。例如，\cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的“c”字符
\d	匹配一个数字字符。等价于[0-9]
\D	匹配一个非数字字符。等价于[^\d]
\f	匹配一个换页符。等价于\x0c 和 \cL
\n	匹配一个换行符。等价于\x0a 和 \cJ
\r	匹配一个回车符。等价于\x0d 和 \cM
\s	匹配任何空白字符，包括空格、制表符、换页符等。等价于[\f\n\r\t\v]
\S	匹配任何非空白字符。等价于[^\f\n\r\t\v]
\t	匹配一个制表符。等价于\x09 和 \cI
\v	匹配一个垂直制表符。等价于\x0b 和 \cK
\w	匹配包括下划线的任何单词字符。等价于 “[A-Za-z0-9_]”
\W	匹配任何非单词字符。等价于 “[^A-Za-z0-9_]”
\xn	匹配 n，其中 n 为十六进制转义值。十六进制转义值必须为确定的两个数字长。例如，“\x41”匹配“A”。“\x041”则等价于“\x04”& “1”。正则表达式中可以使用 ASCII 编码
\num	匹配 num，其中 num 是一个正整数。对所获取的匹配的引用。例如，“(.)1”匹配两个连续的相同字符
\n	标识一个八进制转义值或一个向后引用。如果 \n 之前至少 n 个获取的子表达式，则 n 为向后引用。否则，如果 n 为八进制数字(0~7)，则 n 为一个八进制转义值
\nm	标识一个八进制转义值或一个向后引用。如果 \nm 之前至少有 nm 个获取的子表达式，则 nm 为向后引用。如果 \nm 之前至少有 n 个获取的子表达式，则 n 为一个后跟文字 m 的向后引用。如果前面的条件都不满足，若 n 和 m 均为八进制数字(0~7)，则 \nm 将匹配八进制转义值 nm
\nml	如果 n 为八进制数字(0~3)，且 m 和 l 均为八进制数字(0~7)，则匹配八进制转义值 nml
\un	匹配 n，其中 n 是一个用 4 个十六进制数字表示的 Unicode 字符。例如，\u00A9 匹配版权符号 (©)

## 7.4 Nginx 的 Rewrite 规则编写实例

文件和目录不存在时，重定向到某个 PHP 文件上，适用于 WordPress 等 MVC 结构的开源博客系统：

```
if (!-e $request_filename) {
    rewrite ^/(.*)$ /index.php last;
}
```

多目录转成参数 abc.domian.com/sort/2 => abc.domian.com/index.php?act=sort&name= abc&id=2:

```
if ($host ~* (.*).\domain\.com) {
    set $sub_name $1;
    rewrite ^/sort/(\d+)/?$ /index.php?act=sort&cid=$sub_name&id=$1 last;
}
```

目录对换/123456/xxxx -> /xxxx?id=123456:

```
rewrite ^/(\d+)/(.+)/ /$2?id=$1 last;
```

如果客户端使用 IE 浏览器, 则重定向到/nginx-ie 目录下:

```
if ($http_user_agent ~ MSIE) {
    rewrite ^/(.*)$ /nginx-ie/$1 break;
}
```

禁止访问多个目录:

```
location ~ ^/(cron|templates)/ {
    deny all;
    break;
}
```

禁止访问以/data 开头的文件:

```
location ~ ^/data {
    deny all;
}
```

设置某些类型文件的浏览器缓存时间:

```
location ~ .*\. (gif|jpg|jpeg|png|bmp|swf)$
{
    expires      30d;
}
location ~ .*\. (js|css)?$
{
    expires      1h;
}
```

将多级目录下的文件转换成一个文件/job-123-456-789.html 指向/job/123/456/789.html:

```
rewrite ^/job-([0-9]+)-([0-9]+)-([0-9]+)\.html$ /job/$1/$2/jobshow_$3.html last;
```

禁止访问以.sh、.flv、.mp4 为文件名后缀的 URL 地址:

```
location ~ .*\. (sh|flv|mp4)?$ { return 403; }
```

#适用于 Zend Framework 的重写规则:

```
if ($request_uri ~* "^/pay(.*)" )
{
```

```

    set $var_pay_public '1';
}
if ($request_uri ~* "\.(js|ico|gif|jpg|png|css)$")
{
    set $var_pay_public '0';
}
if ($var_pay_public ~ '1')
{
    rewrite ^(.*)$ /pay/index.php last;
}

```

Bo-blog 开源 PHP 博客系统伪静态重写规则，如代码 7-2 所示：

#### 代码 7-2

```

if (!-x $request_filename)
{
    rewrite ^/post/([0-9]+)/?([0-9]+)/?([0-9]+)/?$
        /read.php?entryid=$1&page=$2&part=$3 last;
    rewrite ^/page/([0-9]+)/([0-9]+)/?$ /index.php?mode=$1&page=$2 last;
    rewrite ^/starred/([0-9]+)/?([0-9]+)/?/$ /star.php?mode=$1&page=$2 last;
    rewrite ^/category/([0-9]+)/?([0-9]+)/?([0-9]+)/?$
        /index.php?go=category_$1&mode=$2&page=$3 last;
    rewrite ^/archiver/([0-9]+)/([0-9]+)/?([0-9]+)/?([0-9]+)/?$
        /index.php?go=archive&cm=$1&cy=$2&mode=$3&page=$4 last;
    rewrite ^/date/([0-9]+)/([0-9]+)/([0-9]+)/?([0-9]+)/?([0-9]+)/?$
        /index.php?go=showday_$1-$2-$3&mode=$4&page=$5 last;
    rewrite ^/user/([0-9]+)/?$ /view.php?go=user_$1 last;
    rewrite ^/tags/([0-9]+)/?([0-9]+)/?([0-9]+)/?/$
        /tag.php?tag=$1&mode=$2&page=$3 last;
    rewrite ^/component/id/([0-9]+)/?$ /page.php?pageid=$1 last;
    rewrite ^/component/([0-9]+)/?$ /page.php?pagealias=$1 last;
    #Force redirection for old rules
    rewrite ^/read\.php/([0-9+)\.htm$ http://$host/post/$1/ permanent;
    rewrite ^/post/([0-9+)\.htm$ http://$host/post/$1/ permanent;
    rewrite ^/post/([0-9+)\_([0-9+)\.htm$ http://$host/post/$1/$2/ permanent;
    rewrite ^/post/([0-9+)\_([0-9+)\_([0-9+)\.htm$
        http://$host/post/$1/$2/$3/ permanent;
    rewrite ^/index\_([0-9+)\_([0-9+)\.htm$ http://$host/page/$1/$2/ permanent;
    rewrite ^/star\_([0-9+)\_([0-9+)\.htm$
        http://$host/starred/$1/$2/ permanent;
    rewrite ^/category\_([0-9+)\.htm$ http://$host/category/$1/ permanent;
    rewrite ^/category\_([0-9+)\_([0-9+)\_([0-9+)\.htm$
        http://$host/category/$1/$2/$3/ permanent;
    rewrite ^/archive\_([0-9+)\_([0-9+)\.htm$
        http://$host/archiver/$1/$2/ permanent;
    rewrite ^/archive\_([0-9+)\_([0-9+)\_([0-9+)\_([0-9+)\.htm$
        http://$host/archiver/$1/$2/$3/$4/ permanent;
    rewrite ^/showday\_([0-9+)\_([0-9+)\_([0-9+)\.htm$
        http://$host/date/$1/$2/$3/ permanent;
    rewrite ^/showday\_([0-9+)\_([0-9+)\_([0-9+)\_([0-9+)\_([0-9+)\.htm$
        http://$host/date/$1/$2/$3/$4/$5/ permanent;
}

```

```
#Filename alias
rewrite ^/([a-zA-Z0-9_-]+)/?([0-9]+)?/?([0-9]+)?/?$
/read.php?blogalias=$1&page=$2&part=$3 last;
}
```

根据 Referer 信息防盗链，代码如下：

```
location ~* \.(gif|jpg|png|swf|flv)$ {
    valid_referers none blocked www.yourdomain.com *.yourdomain.com;
    if ($invalid_referer) {
        rewrite ^/(.*) http://www.yourdomain.com/blocked.html;
    }
}
```

## 7.5 Nginx 与 Apache 的 Rewrite 规则实例对比

### 7.5.1 简单的 Nginx 与 Apache Rewrite 规则

一般情况下，简单的 Nginx 和 Apache Rewrite 规则区别不大，基本上能够完全兼容。例如：

Apache Rewrite 规则，代码如下：

```
RewriteRule ^/(mianshi|xianjing)/$ /z1/index.php?name=$1 [L]
RewriteRule ^/ceshi/$ /z1/ceshi.php [L]
RewriteRule ^/(mianshi)_([a-zA-Z]+)/$ /z1/index.php?name=$1_$2 [L]
RewriteRule ^/pingce([0-9]*)/$ /z1/pingce.php?id=$1 [L]
```

Nginx Rewrite 规则，代码如下：

```
rewrite ^/(mianshi|xianjing)/$ /z1/index.php?name=$1 last;
rewrite ^/ceshi/$ /z1/ceshi.php last;
rewrite ^/(mianshi)_([a-zA-Z]+)/$ /z1/index.php?name=$1_$2 last;
rewrite ^/pingce([0-9]*)/$ /z1/pingce.php?id=$1 last;
```

由以上示例可以看出，Apache 的 Rewrite 规则改为 Nginx 的 Rewrite 规则，其实很简单：Apache 的 RewriteRule 指令换成 Nginx 的 rewrite 指令，Apache 的 [L] 标记换成 Nginx 的 last 标记，中间的内容不变。

如果 Apache 的 Rewrite 规则改为 Nginx 的 Rewrite 规则后，使用 `nginx -t` 命令检查发现 `nginx.conf` 配置文件有语法错误（主要是大括号引起的），那么可以尝试给条件加上引号。例如以下的 Nginx Rewrite 规则会报语法错误：

```
rewrite ^/([0-9]{5}).html$ /x.jsp?id=$1 last;
```

加上引号就正确了：

```
rewrite "^/([0-9]{5}).html$" /x.jsp?id=$1 last;
```





Apache 与 Nginx 的 Rewrite 规则在 URL 跳转时有细微的区别:

Apache Rewrite 规则, 如下:

```
RewriteRule ^/html/tagindex/([a-zA-Z]+)/.*$ /$1/ [R=301,L]
```

Nginx Rewrite 规则, 如下:

```
rewrite ^/html/tagindex/([a-zA-Z]+)/.*$ http://$host/$1/ permanent;
```

以上示例中, 我们注意到, Nginx Rewrite 规则的置换串中增加了“http://\$host”, 这是在 Nginx 中要求的。

另外, Apache 与 Nginx 的 Rewrite 规则在变量名称方面也有区别, 例如:

Apache Rewrite 规则, 如下:

```
RewriteRule ^/user/login/$ /user/login.php?login=1&forward=http://%{HTTP_HOST} [L]
```

Nginx Rewrite 规则, 如下:

```
rewrite ^/user/login/$ /user/login.php?login=1&forward=http://$host last;
```

下面, 我们来介绍 Apache 与 Nginx Rewrite 规则的一些功能相同或类似的指令、标记对应关系:

- Apache 的 RewriteCond 指令对应 Nginx 的 if 指令;
- Apache 的 RewriteRule 指令对应 Nginx 的 rewrite 指令;
- Apache 的[R]标记对应 Nginx 的 redirect 标记;
- Apache 的[P]标记对应 Nginx 的 last 标记;
- Apache 的[R,L]标记对应 Nginx 的 redirect 标记;
- Apache 的[P,L]标记对应 Nginx 的 last 标记;
- Apache 的[PT,L]标记对应 Nginx 的 last 标记;

如果要编写复杂的 Rewrite 规则, 请阅读下一节的示例。

## 7.5.2 允许指定的域名访问本站, 其他域名一律跳转

Apache Rewrite 规则, 代码如下:

```
RewriteCond %{HTTP_HOST} !^(\.??)\.aaa\.com$ [NC]  
RewriteCond %{HTTP_HOST} !^192\.168\.1\.(\.??)$  
RewriteCond %{HTTP_HOST} !^localhost$  
RewriteRule ^/(.*)$ http://www.aaa.com [R,L]
```

Nginx Rewrite 规则如代码 7-3 所示:

### 代码 7-3

```
if ($host ~* ^(.*)\.aaa\.com$)
{
    set $var_tz '1';
}
if ($host ~* ^192\.168\.1\.(.*)$)
{
    set $var_tz '1';
}
if ($host ~* ^localhost)
{
    set $var_tz '1';
}
if ($var_tz != '1')
{
    rewrite ^/(.*)$ http://www.aaa.com/ redirect;
}
```

允许指定的域名访问本站，其他域名一律跳转到 <http://www.aaa.com>。

## 7.5.3 URL 重写与反向代理同时进行

Apache Rewrite 规则，代码如下：

```
ProxyRequests Off
RewriteRule ^/news/(.*)$ http://server.domain.com/$1 [P,L]
```

Nginx Rewrite 规则，代码如下：

```
location /news/
{
    proxy_pass http://server.domain.com/;
}
```

## 7.5.4 指定 URL 之外的 URL 进行 Rewrite 跳转

Apache Rewrite 规则，代码如下：

```
RewriteCond %{REQUEST_URI} !^/xiaoqu/admin/.*
RewriteCond %{REQUEST_URI} !^/xiaoqu/map/.*
RewriteCond %{REQUEST_URI} !^/xiaoqu/accounts/.*
RewriteCond %{REQUEST_URI} !^/xiaoqu/ajax/.*
RewriteRule ^/xiaoqu/(.*)/(.*)/ /xiaoqu/$2.php?name=$1 [L]
```

Nginx Rewrite 规则如代码 7-4 所示：

### 代码 7-4

```
if ($request_uri ~* "^/xiaoqu/admin/.*)"
```

```
{
    set $var_xiaoqu_admin '1';
}
if ($request_uri ~* "^/xiaoqu/map/.*)"
{
    set $var_xiaoqu_admin '1';
}
if ($request_uri ~* "^/xiaoqu/accounts/.*)"
{
    set $var_xiaoqu_admin '1';
}
if ($request_uri ~* "^/xiaoqu/ajax/.*)"
{
    set $var_xiaoqu_admin '1';
}
if ($var_xiaoqu_admin != '1')
{
    rewrite ^/xiaoqu/(.*/)(.*/)$ /xiaoqu/$2.php?name=$1 last;
}
```

---

### 7.5.5 域名前缀作为重写规则变量的示例

Apache Rewrite 规则，代码如下：

```
RewriteCond %{HTTP_HOST} ^(.*)\.domain\.com$
RewriteCond %{HTTP_HOST} !^qita\.domain\.com$
RewriteCond %{DOCUMENT_ROOT}/html/zhuanty/secondmarket/%1/index.htm -f
RewriteRule ^/wu/$ /html/zhuanty/secondmarket/%1/index.htm [L]
```

Nginx Rewrite 规则如代码 7-5 所示：

代码 7-5

```
if ($host ~* ^(.*)\.domain\.com$)
{
    set $var_wupin_city $1;
    set $var_wupin '1';
}
if ($host ~* ^qita\.domain\.com$)
{
    set $var_wupin '0';
}
if (!-f $document_root/html/zhuanty/secondmarket/$var_wupin_city/index.htm)
{
    set $var_wupin '0';
}
if ($var_wupin ~ '1')
{
    rewrite ^/wu/$ /html/zhuanty/secondmarket/$var_wupin_city/index.htm last;
}
```

---