链滴

Nginx 配置 location 以及 return、rewrite 和 try_files 指令

作者：zwxbest

原文链接：https://ld246.com/article/1543136511525

来源网站：链滴

```
<h2 id="正则表达式">正则表达式</h2>
<p>Nginx 使用 <a href="https://ld246.com/forward?goto=http%3A%2F%2Fperldoc.perl.org
2Fperlre.html" target="_blank" rel="nofollow ugc">perl</a> 语法的正则表达式。</p>
<p>正则表达式的用法可以参考 <a href="https://ld246.com/forward?goto=https%3A%2F%2Fb
og.csdn.net%2Fkikajack%2Farticle%2Fdetails%2F79035061" target="_blank" rel="nofollow ug
">这里</a>。</p>
<h2 id="Nginx-内置的全局变量">Nginx 内置的全局变量</h2>
<p><a href="https://ld246.com/forward?goto=https%3A%2F%2Fmoonbingbing.gitbooks.io
2Fopenresty-best-practices%2Fopenresty%2Finline_var.html" target="_blank" rel="nofollow
gc">使用 Nginx 内置绑定变量 · OpenResty最佳实践</a></p>
<h2 id="location">location</h2>
<p>在 Nginx 的配置文件中，通过 location 匹配用户请求中的 URI。格式如下：</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">location 前缀字符串  URL {
</span></span><span class="highlight-line"><span class="highlight-cl">    [ 配置 ]
</span></span><span class="highlight-line"><span class="highlight-cl">}
</span></span></code></pre>
<h2 id="前缀字符串及优先级">前缀字符串及优先级</h2>
<p>其中，前缀字符串部分支持 5 种：</p>
<ul>
<li><code>=</code>：精确匹配，优先级最高。如果找到了这个精确匹配，则停止查找。</li>
<li><code>^~</code>：URI 以某个常规字符串开头，不是正则匹配</li>
<li><code>~</code>：区分大小写的正则匹配</li>
<li><code>~*</code>：不区分大小写的正则匹配</li>
<li><code>/</code>：通用匹配, 优先级最低。任何请求都会匹配到这个规则</li>
</ul>
<p>优先级为：  <code>=</code> &gt; 完整路径 &gt; <code>^~</code> &gt; <code>~</c
de>、<code>~*</code> &gt; 部分起始路径 &gt; <code>/</code></p>
<h2 id="示例">示例</h2>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl"># 精确匹配 / ，域名后面不能带任何字符串。匹配到后，停止继续匹配
</span></span><span class="highlight-line"><span class="highlight-cl">location  = / {
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">}
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl"># 匹配到所有请求
</span></span><span class="highlight-line"><span class="highlight-cl">location  / {
</span></span><span class="highlight-line"><span class="highlight-cl">    if (-f $request_fi
ename) {
</span></span><span class="highlight-line"><span class="highlight-cl">        expires max;
</span></span><span class="highlight-line"><span class="highlight-cl">        break;
</span></span><span class="highlight-line"><span class="highlight-cl">    }
</span></span><span class="highlight-line"><span class="highlight-cl">    if (!-e $request_f
lename) {
</span></span><span class="highlight-line"><span class="highlight-cl">        rewrite ^/(.*)
 /index.php/$1 break;
</span></span><span class="highlight-line"><span class="highlight-cl">    }
</span></span><span class="highlight-line"><span class="highlight-cl">    index index.php

</span></span><span class="highlight-line"><span class="highlight-cl">    autoindex off;
</span></span><span class="highlight-line"><span class="highlight-cl">}
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl"># 匹配任何以 /doc
ments/ 开头的 URI。优先级低于正则表达式，匹配到后还会继续往下匹配，当后面没有正则匹配或
```

则匹配失败时，使用这里代码
location /documents/ {
}

# 最长字符匹配到 /images/abc，继续往下，会发现 ^~ 存在
location /images/abc {
}

# 匹配任何以 /images/ 开头的 URI。优先级高于正则表达式，匹配成功后，停止往下搜索正则。
location ^~ /images/ {
}

# 正则匹配，区分小写。匹配任何以 /documents/Abc 开头的地址，匹配符合以后，还要继续往下搜索
location ~ /documents/Abc {
}

# 正则匹配，忽略小写。匹配所有以 gif、jpg 或 jpeg 结尾的请求
location ~* \.(gif|jpg|jpeg)$ {
}
</code></pre>
<h2 id="location-匹配原则">location 匹配原则</h2>
可以 参考这篇译文。
每个请求的处理逻辑顺序如下：

用所有的前缀字符串测试 URI。
等号 <code>=</code> 定义了前缀字符串和 URI 的精确匹配关系。如果找到了这个精确匹配则停止查找。
如果 <code>^~</code> 修饰符预先匹配到最长的前缀字符串，则不检查正则表达式。
存储最长的匹配前缀字符串。
用正则表达式测试 URI。
匹配到第一个正则表达式后停止查找，使用对应的 location。
如果没有匹配到正则表达式，则使用之前存储的前缀字符串对应的 location。

<h2 id="if-和-break-指令">if 和 break 指令</h2>
可以参考 Nginx 块 - ngx_http_rewrite_module。
<h3 id="if">if</h3>
if 的可用上下文有：server、location。if 的条件可能是以下任何一种情况：
变量名；如果变量值是空字符串或“0”则为 FALSE。注意，在 1.0.1 版本之前，任何以“0”开的字符串都会被当做 FALSE。
使用“=”和“!=”的变量跟字符串的比较
使用“~”（区分大小写匹配）和“~*”（不区分大小写匹配）运算符将变量与正则表达式匹配。正

表达式可以包含捕获，之后可以通过 <code>$1</code> 到 <code>$9</code> 这几个变量名重使用。“!~”和“!~*”用作不匹配运算符。如果正则表达式包含“}”或“;”字符，则整个表达式应用单引号或双引号括起来。
用“-f”和“!-f”运算符检查文件是否存在
用“-d”和“!-d”运算符检查目录是否存在
用“-e”和“!-e”运算符检查文件、目录或符号链接的存在性
用“-x”和“!-x”运算符检查可执行文件
示例：
<pre><code class="highlight-chroma"># 如果用户代理 User-Agent 包含"MSIE"，rewrite 请求到 /msie/ 目录下。通过正则匹配的捕可以用 $1 $2 等使用
if ($http_user_agent ~ MSIE) {
    rewrite ^(.*)$ /msie/$1 break;
}

# 如果 cookie 匹正则，设置变量 $id 等于匹配到的正则部分
if ($http_cookie ~* "id=([^;]+)(?:;|$)") {
    set $id $1;
}

# 如果请求方法为OST，则返回状态 405（Method not allowed）
if ($request_method = POST) {
    return 405;
}

# 如果通过 set 指设置了 $slow，限速
if ($slow) {
    limit_rate 10k;
}

# 如果请求的文件在，则开启缓存，并通过 break 停止后面的检查
if (-f $request_filename) {
    expires max;
    break;
}

# 如果请求的文件目录或符号链接都不存在，则用 rewrite 在 URI 头部添加 /index.php
if (!-e $request_filename) {
    rewrite ^/(.*)$ /index.php/$1 break;
}
</code></pre>
<h3 id="break">break</h3>

break 的可用上下文有：server、location、if。用于停止处理当前的 ngx_http_rewrite_module 指令集合。
<pre><code class="highlight-chroma">if ($slow) {
    limit_rate 10k;
    break;
}
</code></pre>
<h2 id="return-rewrite-和-try-files-指令">return、rewrite 和 try_files 指令</h2>
NGINX rewrite 的两个通用指令是 <code>return</code> 和 <code>rewrite</code>，而 <code>try_files</code> 指令可以将请求定向到应用程序服务器。
<h2 id="return-指令">return 指令</h2>
return 指令手册
在重定向满足两个条件时适用：

重写的 URL 适用于每个匹配的 <code>server</code> 或 <code>location</code> 的请求
可以使用标准的 NGINX 变量构建重写的 URL

<code>return</code> 指令简单高效，建议尽量使用 <code>return</code>，而不是 <code>rewrite</code>。
<code>return</code> 指令放在 <code>server</code> 或 <code>location</code> 上文中。语法很简单：
<pre><code class="highlight-chroma">return code [text];
return code URL;
return URL;
</code></pre>
将客户重定向到一个新域名的示例：
<pre><code class="highlight-chroma">server {
    listen 80;
    listen 443 ssl;
    server_name www.old-name.com;
    return 301 $scheme://www.new-name.com$request_uri;
}
</code></pre>
上面代码中，listen 指令表明 server 块同时用于 HTTP 和 HTTPS 流量。<code>server_name</code> 指令匹配包含域名 ‘www.old-name.com’ 的请求。<code>return</code> 指令告诉 Nginx 停止处理请求，直接返回 <code>301 (Moved Permanently)</code> 代码和指定的重写过的 URL 到客户端。<code>$scheme</code> 是协议（HTTP 或 HTTPS），<code>$request_uri</code> 是包含参数的完整的 URI。
对于 3xx 系列响应码，<code>url</code> 参数定义了新的（重写过的）URL：
<pre><code class="highlight-chroma">return (301 | 302 | 303 | 307) url;
</code></pre>
对于其他响应码，可以选择定义一个出现在响应正文中的文本字符串（HTTP 代码的标准文本，如 404 的 Not Found，仍包含在标题中）。文本可以包含 NGINX 变量。
<pre><code class="highlight-chroma"><span class="highlight

cl">return (1xx | 2xx | 4xx | 5xx) ["text"];
</code></pre>
例如，在拒绝没有有效身份验证令牌的请求时，此指令可能适用：
<pre><code class="highlight-chroma"><span class="highlight
cl">return 401 "Access denied because token is expired or invalid";
</code></pre>
通过 error_page 指令，可以为每个 HTTP 代码返回一个完整的自定义 HTML 页面，也可以更
响应代码或执行重定向。
<h2 id="rewrite-指令">rewrite 指令</h2>
<a href="https://ld246.com/forward?goto=https%3A%2F%2Fwww.nginx.com%2Fblog%
Fcreating-nginx-rewrite-rules%2F" target="_blank" rel="nofollow ugc">NGINX Rewrite 规则
方文档
<a href="https://ld246.com/forward?goto=http%3A%2F%2Fnginx.org%2Fen%2Fdocs%2Fhtt
%2Fngx_http_rewrite_module.html" target="_blank" rel="nofollow ugc">Rewrite 模块手册</a

<a href="https://ld246.com/forward?goto=http%3A%2F%2Fwww.w3.org%2FProtocols%2Frfc
616%2Frfc2616-sec10.html" target="_blank" rel="nofollow ugc">HTTP 响应码
rewrite 规则会改变部分或整个用户请求中的 URL，主要有两个用途：

通知客户端，请求的资源已经换地方了。例如网站改版后添加了 www 前缀，通过 rewrite 规则
以将所有请求导向新站点。
控制 Nginx 中的处理流程。例如当需要动态生成内容时，将请求转发到应用程序服务器。<code
try_files</code> 指令经常用于这个目的。

但是，如果需要测试 URL 之间更复杂的区别，或者要从原始 URL 中捕获的元素没有对应的 NGI
X 变量，或者更改或添加路径中的元素（例如各大 PHP 框架常用的 <code>index.php</code> 入
文件），该怎么办？ 可以使用 <code>rewrite</code> 指令。
<code>rewrite</code> 指令放在 <code>server</code> 或 <code>location</code> 上
文中。语法很简单：
<pre><code class="highlight-chroma"><span class="highlight
cl">rewrite regex URL [flag];
</code></pre>
第一个参数 regex 是正则表达式。
<h3 id="flag-标志位">flag 标志位</h3>
flag 支持以下 4 个选项：

last：停止处理当前的 ngx_http_rewrite_module 指令集，并开始对匹配更改后的 URI 的新 loca
ion 进行搜索（再从 server 走一遍匹配流程）。此时对于当前 <code>server</code> 或 <code>lo
ation</code> 上下文，不再处理 ngx_http_rewrite_module 重写模块的指令。
break：停止处理当前的 ngx_http_rewrite_module 指令集
redirect：返回包含 302 代码的临时重定向，在替换字符串不以“http://”， “https://”或“$s
heme”开头时使用
permanent：返回包含 301 代码的永久重定向。

last 和 break 的区别及共同处：

last 重写 url 后，会再从 server 走一遍匹配流程，而 break 终止重写后的匹配
break 和 last 都能阻止后面的 rewrite 指令再次执行

<code>rewrite</code> 指令只能返回代码 301 或 302。要返回其他代码，需要在 <
ode>rewrite</code> 指令后面包含 <code>return</code> 指令。
<code>rewrite</code> 指令不一定会暂停 NGINX 对请求的处理，因为它不一定会发送重定
到客户端。除非明确指出（使用 flag 或 URL 的语法）你希望 NGINX 停止处理或发送重定向，否则
将在整个配置中运行，查找在重写模块中定义的指令（break、if、return、rewrite 和 set），并按

序处理。如果重写的 URL 与 Rewrite 模块中的后续指令匹配，NGINX 会对重写的 URL 执行指定的
作（通常会重新写入）。</p>
<p>这是复杂的地方，必须仔细计划指令顺序以获得期望的结果。例如，如果原始 location 块和其
的 NGINX 重写规则与重写的 URL 匹配，NGINX 可以进入一个循环，Nginx 默认限制循序最大 10
。</p>
<h3 id="示例-">示例</h3>
<p>下面是使用 <code>rewrite</code> 指令的 NGINX 重写规则的示例。它匹配以字符串 <code
/download</code> 开头的 URL，然后用 <code>/mp3/</code> 替换在路径稍后的某个位置包
的 <code>/media/</code> 或 <code>/audio/</code> 目录，并添加适当的文件扩展名 <code>
mp3</code> 或 <code>.ra</code>。<code>$1</code> 和 <code>$2</code> 变量捕获不
的路径元素。例如，<code>/download/cdn-west/media/file1</code> 变为 <code>/download
cdn-west/mp3/file1.mp3</code>。如果文件名上有扩展名（例如.flv），表达式会将其剥离并用.m
3 替换。</p>
<pre><code class="highlight-chroma"><span class="highlight
cl">server {
   # ...
   rewrite ^(/down
oad/.*)/media/(\w+)\.?.*$ $1/mp3/$2.mp3 last;
   rewrite ^(/down
oad/.*)/audio/(\w+)\.?.*$ $1/mp3/$2.ra last;
   return 403;
   # ...
}
</code></pre>
<p>可以将 flag 添加到重写指令来控制处理流程。示例中的 last 告诉 NGINX 跳过当前服务器或位
块中的任何后续 ngx_http_rewrite_module 重写模块的指令，并开始搜索与重写的 URL 匹配的新位
。</p>
<p>这个例子中的最后一个 <code>return</code> 指令意味着如果 URL 不匹配任何一个 <code>r
write</code> 指令，将返回给客户端 403 代码。</p>
<h2 id="try-files-指令">try_files 指令</h2>
<p><code>try_files</code> 指令也放在 <code>server</code> 或 <code>location</code>
下文中。语法很简单：</p>
<pre><code class="highlight-chroma"><span class="highlight
cl">try_files file ... uri;
</code></pre>
<p><code>try_files</code> 指令的参数是一个或多个文件或目录的列表，以及最后面的 URI 参
。</p>
<p>Nginx 会按顺序检查文件及目录是否存在（根据 <code>root</code> 和 <code>alias</cod
> 指令设置的参数构造完整的文件路径），并用找到的第一个文件提供服务。在元素名后面添加斜杠
code>/</code> 表示这个是目录。如果文件和目录都不存在，Nginx 会执行内部重定向，跳转到命
的最后一个 uri 参数定义的 URI 中。</p>
<p>要想 <code>try_files</code> 指令工作，必须定义一个 location 块捕捉内部重定向。最后一
参数可以是命名过的 location，由初始符号（@）指示。</p>
<p><code>try_files</code> 指令通常使用 <code>$uri</code> 变量，表示 URL 中域名之后的
分。</p>
<p>下面示例中，如果客户端请求的文件不存在，Nginx 会响应一个默认的 GIF 文件。假设客户请
“<a href="https://ld246.com/forward?goto=http%3A%2F%2Fwww.domain.com%2Fimages
2Fimage1.gif" target="_blank" rel="nofollow ugc">http://www.domain.com/images/image1.g
f”，Nginx 会首先通过用于这个 location 的 <code>root</code> 和 <code>alias</code>
指令，在本地目录中查找这个文件。如果“image1.gif”文件不存在，Nginx 会查找“image1.gif/
目录，如果都不存在，会重定向到“/images/default.gif”。这个值精确匹配后面的 location 指令
因此处理过程停止，Nginx 返回这个文件，并标注其缓存 30 秒。</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">location /images/ {

```
    try_files $uri $uri/ /images/default.gif;
}

location = /image/default.gif {
    expires 30s;
}
```

原文链接：Nginx 配置 location 以及 return、rewrite 和 try_files 指令