**Nginx Location Rewrite与动静分离**

# 一、Nginx正则及location匹配

## 1、Nginx location规则匹配

^~ 标识符匹配后面跟一个字符串。匹配字符串后将停止对后续的正则表达式进行匹配，如location ^~ /images/，在匹配了/images/这个字符串后就停止对后续的正则匹配

= 精准匹配，如location = /，只会匹配url为/的请求-> <http://www.a.com/>，不匹配默认主页，一般用于跳转

~ 区分大小写的匹配 -> location ~ \.jsp$

~\* 不区分大小写的匹配-> location ~\* \.jsp$

/ 通用匹配, 如果没有其它匹配,任何请求都会被匹配到

## 2、正则表达式

\* 重复前面的字符0次或多次

? 重复前面的字符0次或1次

+ 重复前面的字符1次或多次

. 匹配除换行符以外的任意1个字符

(ac|be) 匹配ac或be

^ 以…开头

$ 以…结尾

{n} 重复前面的字符n次

{n,} 重复前面的字符n次或更多次

{n,m} 重复前面的字符n到m次

[a] 匹配单个字符a

[a-z] 匹配a-z小写字母的任意一个

[^a] 匹配除了a以外的任意字符

[^abc] 匹配除了abc这几个字母以外的任意字符

## 3、Nginx location应用规则

location [=|~|~\*|^~|!~|!~\*|/] /url/{...}

使用字段：server

location参数根据URL的不同需求来进行配置，可以使用字符串与正则表达式匹配，

location ~\* .\*\.jsp$ {

proxy\_pass http://tomcat\_server;

}

**location = / {**

**# 精确匹配 / ，主机名后面不能带任何字符串**

**[ configuration A ]**

**}**

**location / {**

**# 因为所有的地址都以 / 开头，所以这条规则将匹配到所有请求**

**# 但是正则和最长字符串会优先匹配**

**[ configuration B ]**

**}**

**location /documents/ {**

**# 匹配任何以 /documents/ 开头的地址，匹配符合以后，还要继续往下搜索**

**# 只有后面的正则表达式没有匹配到时，这一条才会采用这一条**

**[ configuration C ]**

**}**

**location ~ /documents/abc/ {**

**# 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**

**# 只有后面的正则表达式没有匹配到时，这一条才会采用这一条**

**[ configuration CC ]**

**}**

**location ^~ /images/ {**

**# 匹配任何以 /images/ 开头的地址，匹配符合以后，停止往下搜索正则，采用这一条。**

**[ configuration D ]**

**}**

**location ~\* \.(gif|jpg|png)$ {**

**# 匹配所有以gif,jpg或jpeg 结尾的请求**

**# 然而，所有请求 /images/ 下的图片会被config D 处理，因为 ^~ 到达不了这一条正则**

**[ configuration E ]**

**}**

**location ~ /images/ {**

**# 字符匹配到 /images/，继续往下，会发现 ^~ 存在**

**[ configuration F ]**

**}**

**location /images/abc {**

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

**# F与G的放置顺序是没有关系的**

**[ configuration G ]**

**}**

**location ~ /images/abc/ {**

**# 只有去掉config D 才有效：先最长匹配config G 开头的地址，继续往下搜索，匹配到这一条正则，采用**

**[ configuration H ]**

### 匹配顺序优先级：

(location =) > (location 完整路径) > (location ^~ 路径) > (location ~,~\* 正则顺序) > (location 部分起始路径) > (/)

按照上面的location匹配，分析以下案例：

/ ->config A

精确完全匹配，即使/index.html也匹配不了

/downloads/ ->config B

匹配B以后，往下没有任何匹配，采用B

/images/1.gif -> config D

匹配到B，往下匹配到D，停止往下

/images/abc/def ->config D

最长匹配到G，往下匹配D，停止往下

你可以看到 任何以/images/开头的都会匹配到D并停止，FG写在这里是没有任何意义的，H是永远轮不到的，这里只是为了说明匹配顺序

/documents/index.html ->config C

匹配到C，往下没有任何匹配，采用C

/documents/linux.jpg -> configuration E

匹配到C，往下正则匹配到E

需要在E的路径下建立documents目录并在此目录下有linux.png才可以访问成功

/Error.Jpg 匹配到B，往下匹配到E，因为E用了~\*不区分大小写的匹配，所以可以访问成功（确保目录下有Error.Jpg文件）,在linux浏览器中访问。

/documents/abc/ ->config CC

匹配到C，往下正则顺序匹配到CC

### 实际使用建议

所以实际使用中，个人觉得至少有三个匹配规则定义，如下：

#直接匹配网站根，通过域名访问网站首页比较频繁，使用这个会加速处理。这里是直接转发给后端应用服务器了，也可以是一个静态首页。

# 第一个必选规则

location = / {

proxy\_pass http://192.168.200.112;

}

# 第二个必选规则是处理静态文件请求，这是nginx作为http服务器的强项

# 有两种配置模式，目录匹配或后缀匹配,任选其一或搭配使用

location ^~ /documents/ {

root /benet/;

}

location ~\* \.(gif|jpg|jpeg|png|css|js|ico)$ {

root /benet/images;

}

location ~\* \.php$ {

proxy\_pass http://192.168.200.112;

}

#第三个规则就是通用规则，用来转发动态请求到后端应用服务器，非静态文件请求就默认是动态请求，自己根据实际把握。

#毕竟目前的一些框架的流行，带.php,.jsp后缀的情况很少了

location / {

proxy\_pass http://tomcat:8080;

}

# 二、Nginx Rewrite规则

Nginx Rewrite功能是使用nginx提供的全局变量或自己设置的变量，结合正则表达式和标志位实现URL重写以及重定向功能。Rewrite指令只能放在server{},location{},if{}中，并且只能对域名后边的除去传递的参数外的字符串起作用。

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，如果需要Nginx的Rewrite功能，在编译安装Nginx之前，必须安装PCRE库。

例如：<http://www.crushlinux.com/a/we/index.php?id=1&u=admin只对URL中的/a/we/index.php>等字符串起作用。

URL是Uniform Resource Location的缩写，译为“统一资源定位符”。

## 语法：

rewrite 正则表达式 更换目标 [标志位];

rewrite和location功能有点像，都能实现跳转。主要区别在于rewrite是在同一域名内更改获取资源的路径，而location是对路径做控制访问或反向代理，可以使用proxy\_pass代理到其他机器。很多情况下rewrite也会写在location里，它们的执行顺序是：

1. 执行server块的rewrite指令
2. 执行location匹配
3. 执行选定的location中的rewrite指令

## if指令与全局变量

### if判断指令

语法为if(condition){...}，对给定的条件condition进行判断。如果为真，大括号内的rewrite指令将被执行，if条件(conditon)可以是如下任何内容：

当表达式只是一个变量时，如果其值为空或任何以0开头的字符串时都会当作条件为false

直接比较变量和内容时，使用=或!=

-f和!-f 用来判断是否存在文件

-d和!-d 用来判断是否存在目录

-e和!-e 用来判断是否存在文件或目录

-x和!-x 用来判断文件是否可执行

例如： <http://www.a.com/msie/a/b/test.html>

**if ($http\_user\_agent ~ MSIE) {**

**rewrite ^(.\*)$ /msie/$1;**

**}**

**//如果UA包含"MSIE"，rewrite请求到/msie/目录下**

### Nginx变量

下面是可以用作if判断的变量

$args： 记录请求行中的参数

$content\_length： 记录请求头中的Content-length字段。

$document\_root： 记录当前请求在root指令中指定的值，即网站的根目录路径。

$host ： 记录请求主机头字段，否则为服务器名称。

$http\_user\_agent： 记录用于记录客户端浏览器的相关信息

$http\_cookie： 记录客户端cookie信息

$limit\_rate： 记录可以限制连接速率。

$request\_uri： 记录完整的url

$request： 记录客户端请求的动作和http版本信息，通常为GET或POST。

$request\_filename： 记录当前请求的文件路径，由root或alias指令与URI请求生成。

$scheme ： 记录HTTP方法（如http，https）。

$server\_addr： 记录服务器地址，在完成一次系统调用后可以确定这个值。

$server\_name： 记录服务器名称。

$server\_port： 记录请求到达服务器的端口号。

$remote\_addr： 记录远程客户端的IP地址。

$remote\_port： 记录远程客户端的端口。

$remote\_user 记录远程客户端用户名称

$time\_local 记录访问时间及时区

$status 记录请求的状态，例如成功时为200，页面找不到时为404

$http\_referer 记录是从哪个页面链接访问过来的

$invalid\_referer： valid\_referers列表中没有指定的Referer头的值

**URL分析：**

**http://localhost:88/test1/test2/test.php**

**$host：localhost**

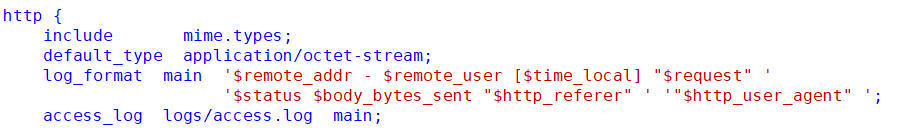
**$server\_port：88**

**$request\_uri：http://localhost:88/test1/test2/test.php**

**$document\_root：/var/www/html**

**$request\_filename：/var/www/html/test1/test2/test.php**

**Nginx日志案例：**



**log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '**

**'$status $body\_bytes\_sent "$http\_referer" ' '"$http\_user\_agent" ';**

**日志格式中字段对应的是：main是日志格式名称（具体使用什么日志格式使用名字来调用）**

**IP地址-用户名 [本地时间] “请求” 状态码 主体内容大小 “访问来源链接” ”浏览器信息”**

**查看/usr/local/nginx/logs/access.log日志文件，可以看到日志按定义的格式显示**

**192.168.200.1 - - [10/Mar/2022:21:26:50 +0800] "GET /index.php HTTP/1.1" 200 43811 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36"**

**192.168.200.1 - - [10/Mar/2022:21:26:50 +0800] "GET /index.php?=PHPE9568F34-D428-11d2-A769-00AA001ACF42 HTTP/1.1" 200 2524 "http://192.168.200.111/index.php" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36"**

防止盗链

**localtion ~\*\.(gif|jpg|png|swf|flv)${**

**valid\_referers none blocked www.test.com \*.test.com;**

**if($invalid\_referer){**

**rewrite ^/(.\*)** [**http://www.test.com/block.html**](http://www.test.com/block.html)**;**

**}**

**}**

实现域名跳转

所有对www.360buy.com的访问，rewrite到www.jd.com

**server {**

**listen 80;**

**server\_name www.jd.com;**

**charset utf-8;**

**root html;**

**index index.html index.htm;**

**if ($host = "www.360buy.com") {**

**rewrite ^(.\*)$ http://www.jd.com/$1;**

**}**

**}**

# 三、Nginx静动分离

**实验环境：两台服务器**

**192.168.200.111 Nginx服务器**

**192.168.200.112 LAMP服务器**

## LAMP rpm方式搭建(112上配置)

**[root@localhost ~]# yum -y install httpd mariadb mariadb-devel mariadb-server php**

**[root@localhost ~]# systemctl start httpd**

**[root@localhost ~]# systemctl start mariadb**

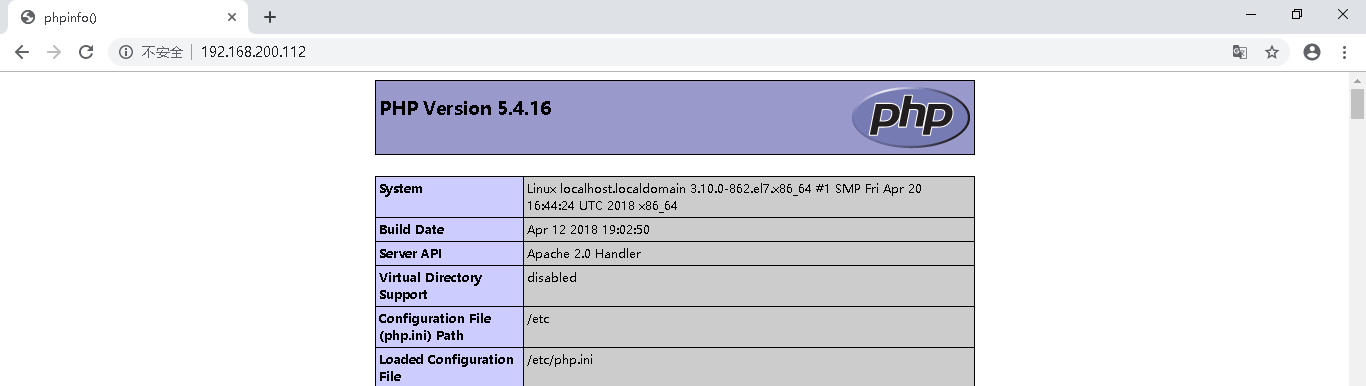
**[root@localhost ~]# cat /var/www/html/index.php**

**<?php**

**phpinfo();**

**?>**

**宿主机测试：**



## 配置Nginx动静分离（111上配置）

### 修改nginx.conf主配置文件

**location ~\* \.(png|bmp|jpg)$ {**

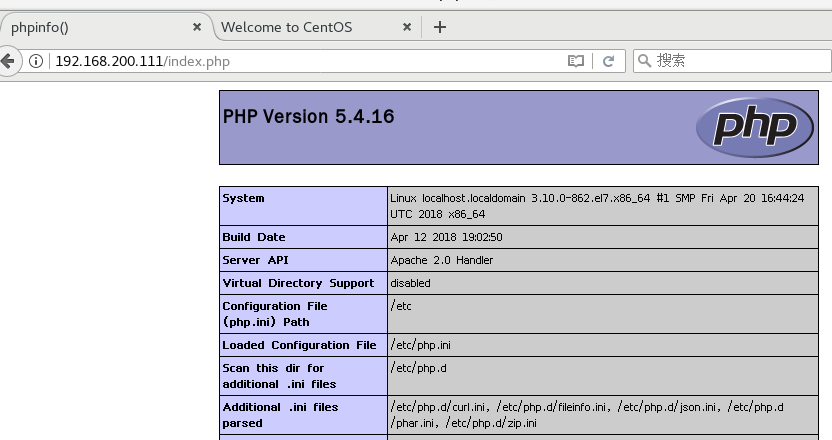
**root /usr/local/nginx/html/images;**

**}**

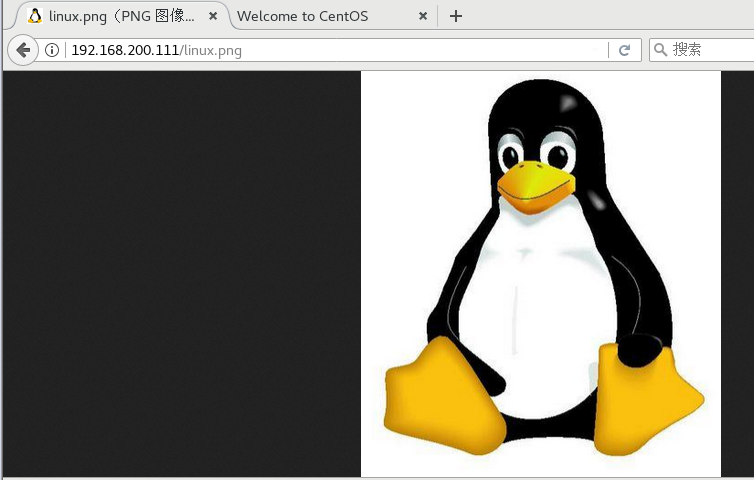
**location ~\* \.php$ {**

**proxy\_pass http://192.168.200.112;**

**}**



访问192.168.200.111的php页面时跳转到了192.168.200.112



访问图片时仍然在本地服务器上