第1章 目录索引-下载服务

0.官方地址

```
http://nginx.org/en/docs/http/ngx_http_autoindex_module.html
```

1.应用场景

使用nginx作为简易的文件下载服务器

2.参数说明

参数解释:

```
Syntax: autoindex on | off;
Default: autoindex off;
Context: http, server, location

# autoindex 常用参数
autoindex_exact_size off;
默认为 on， 显示出文件的确切大小，单位是 bytes。
修改为 off，显示出文件的大概大小，单位是 kB 或者 MB 或者 GB。

autoindex_localtime on;
默认为 off，显示的文件时间为 GMT 时间。
修改为 on， 显示的文件时间为文件的服务器时间。

charset utf-8,gbk;
默认中文目录乱码，添加上解决乱码
```

3.配置文件

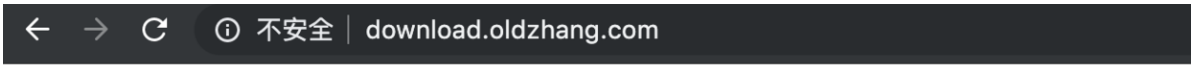
```
cat >/etc/nginx/conf.d/download.conf <<EOF
server {
    listen 80;
    server_name download.oldzhang.com;
    location / {
        root /usr/share/nginx/html/download;
        charset utf-8,gbk;
        autoindex on;
        autoindex_localtime on;
        autoindex_exact_size off;
    }
}
EOF
```

4.创建测试数据

```
touch /usr/share/nginx/html/download/{1..10}.txt
```

5.访问页面

download.oldzhang.com



Index of /

../		
1.txt	30-Jul-2019 18:03	0
10.txt	30-Jul-2019 18:03	0
2.txt	30-Jul-2019 18:03	0
3.txt	30-Jul-2019 18:03	0
4.txt	30-Jul-2019 18:03	0
5.txt	30-Jul-2019 18:03	0
6.txt	30-Jul-2019 18:03	0
7.txt	30-Jul-2019 18:03	0
8.txt	30-Jul-2019 18:03	0
9.txt	30-Jul-2019 18:03	0

第2章 状态监控

0.官方地址

```
http://nginx.org/en/docs/http/ngx_http_stub_status_module.html
```

1.状态字段解释

```
Active connections # 当前活动的连接数
accepts # 当前的总连接数 TCP
handled # 成功的连接数 TCP
requests # 总的 http 请求数
Reading # 请求
Writing # 响应
Waiting # 等待的请求数，开启了 keepalive
# 注意，一次 TCP 的连接，可以发起多次 http 的请求，如下参数可配置进行验证
keepalive_timeout 0; # 类似于关闭长连接
keepalive_timeout 65; # 65s 没有活动则断开连接
```

2.配置文件

```
cat > /etc/nginx/conf.d/status.conf <<EOF
server {
   listen 80;
   server_name  status.oldzhang.com;
   stub_status on;
   access_log off;
}
EOF
```

3.访问测试

```
[root@web01 ~]# curl status.oldboy.com
Active connections: 1
server accepts handled requests
 4 4 6
Reading: 0 Writing: 1 Waiting: 0
```

第3章 Nginx 基于IP的访问控制

0.官方地址

```
http://nginx.org/en/docs/http/ngx_http_access_module.html
```

1.配置语法

```
#允许配置语法
Syntax: allow address | CIDR | unix: | all;
Default: —
Context: http, server, location, limit_except

#拒绝配置语法
Syntax: deny address | CIDR | unix: | all;
Default: —
Context: http, server, location, limit_except\
```

2.配置案例1: 拒绝windows访问www域名

```
cat > /etc/nginx/conf.d/01-www.conf <<EOF
server   {
    listen       80;
    server_name  www.oldzhang.com;
    location / {
        root   /usr/share/nginx/html/www;
        index  index.html index.htm;
        deny 10.0.0.1;
        allow all;
    }
}
EOF
```

3.windows测试访问

www.oldboy.com

403 Forbidden

nginx/1.16.0

4.使用虚拟机测试访问

```
[root@web01 ~]# curl www.oldzhang.com
www
```

5.配置案例2: 只允许windows访问，其他全部拒绝

```
cat >/etc/nginx/conf.d/01-www.conf<<EOF
server   {
    listen       80;
    server_name  www.oldzhang.com;
    location / {
        root   /usr/share/nginx/html/www;
        index  index.html index.htm;
        allow 10.0.0.1;
        deny all;
    }
}
EOF
```

6.windows测试访问

www.oldboy.com

www

7.使用虚拟机测试访问

```
[root@web01 ~]# curl www.oldzhang.com
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.16.0</center>
</body>
</html>
```

第4章 Nginx 基于用户认证的访问控制

0.官方配置

```
http://nginx.org/en/docs/http/ngx_http_auth_basic_module.html
```

1.配置语法

```
#访问提示字符串
Syntax: auth_basic string| off;
Default: auth_basic off;
Context: http, server, location, limit_except

#账户密码文件
Syntax: auth_basic_user_file file;
Default: -
Context: http, server, location, limit_except
```

2.配置文件

需要安装 httpd-tools，该包中携带了 htpasswd 命令

```
[root@web01 ~]# yum install httpd-tools -y
```

创建新的密码文件, -c 创建新文件 -b 允许命令行输入密码

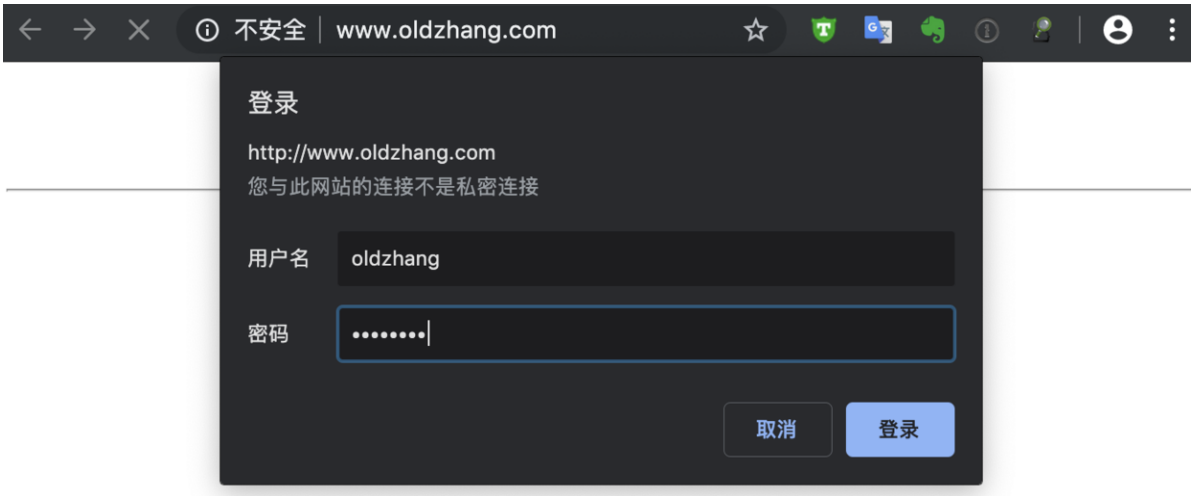
```
[root@web01 ~]# htpasswd -b -c /etc/nginx/auth_conf oldzhang oldzhang
Adding password for user oldzhang
```

nginx 配置调用

```
cat >/etc/nginx/conf.d/01-www.conf <EOF
server   {
    listen       80;
    server_name  www.oldzhang.com;
    location / {
        auth_basic "Auth access Blog Input your Passwd!";
        auth_basic_user_file auth_conf;
        root   /usr/share/nginx/html/www;
        index  index.html index.htm;
    }
}
EOF
```

3.访问测试

www.oldzhang.com



第5章 Nginx 请求限制

0.官方地址

```
http://nginx.org/en/docs/http/ngx_http_limit_req_module.html
```

1.配置语法

```
#模块名 ngx_http_limit_req_module
Syntax: limit_req_zone key zone=name:size rate=rate;
Default: —
Context: http

#引用限速模块
Syntax: limit_conn zone number [burst=number] [nodelay];
Default: —
Context: http, server, location
```

参数解释:

```
定义一条规则
limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;
limit_req_zone $binary_remote_addr zone=two:10m rate=2r/s;

limit_req_zone                  #引用限速模块
$binary_remote_addr             #判定条件，每个请求的IP
zone=one:10m                    #定义一个zone名称
rate=1r/s;                      #限制速度，1秒1次

引用一条限速规则
limit_req zone=two burst=5 nodelay;

limit_req                       #引用限速规则语法
zone=one                        #引用哪一条规则
burst=5                         #令牌桶，允许排队的数量
nodelay;                        #如果不希望在请求被限制时延迟过多的请求，则应使用参数nodelay
```

2.配置文件

```
cat >/etc/nginx/conf.d/01-www.conf <<EOF
limit_req_zone $binary_remote_addr zone=req_zone:10m rate=1r/s;
server   {
    listen       80;
    server_name  www.oldzhang.com;
    limit_req zone=req_zone burst=3 nodelay;
    access_log  /var/log/nginx/www.access.log  main;
    location / {
        root   /usr/share/nginx/html/www;
        index  index.html index.htm;
    }
}
EOF
```

3.访问测试

ab压测

```
yum install httpd-tools -y
ab -n 20 -c 2 http://www.oldzhang.com/
```

for循环

```
一秒1次
for i in {1..10000};do curl -I 10.0.0.7;sleep 1;done

一秒2次
for i in {1..10000};do curl -I 10.0.0.7;sleep 0.5;done

一秒5次
for i in {1..10000};do curl -I 10.0.0.7;sleep 0.2;done
```

4.查看访问日志

```
tail -f /var/log/nginx/www.access.log
```

5.查看错误日志

```
tail -f /var/log/nginx/error.log
```

6.为什么限制请求的效果更好

我们先来回顾一下 http 协议的连接与请求，首先 HTTP 是建立在 TCP 基础之上, 在完成 HTTP 请求需要先建立TCP 三次握手（称为 TCP 连接），在连接的基础上在完成 HTTP 的请求。
所以多个 HTTP 请求可以建立在一次 TCP 连接之上, 那么我们对请求的精度限制，当然比对一个连接的限制会更加的有效，因为同一时刻只允许一个 TCP 连接进入, 但是同一时刻多个 HTTP 请求可以通过一个 TCP 连接进入。所以针对 HTTP 的请求限制才是比较优的解决方案。

第6章 Nginx 内置变量

1.Nginx内置变量介绍

变量介绍：

```
nginx包含了很多内置的变量，这些变量可以用来定义日志的格式或者跳转时作为判断条件。
```

官方地址：

```
http://nginx.org/en/docs/varindex.html
```

2.Nginx内置变量

```

```

```
$args                       #存放URL中的参数，例如：http://www.oldboyedu.com/index.php?id=526195417&partener=search 则args变量结果为：id=526195417&partener=search
$document_root              #当前请求资源的根目录，例如:/code/blog/
$document_uri               #只返回用户请求中的URI，不包含参数，例如：http://www.oldboyedu.com/index.php
$remote_addr                #记录客户端 IP 地址，注意，是公网IP
$remote_user                #记录经过用户认证模块认证后的客户端用户名
$host                       #请求的host名称
$request                    #记录请求的方法以及请求的 http 协议
$request_method             #请求资源的方法，例如GET/PUT/POST
$scheme                     #请求的协议，例如http或https
$status                     #记录请求状态码(用于定位错误信息)
$server_addr                #Nginx服务器的IP地址
$server_name                #Nginx服务器的主机名
$server_port                #Nginx服务器的端口号
$body_bytes_sent            #发送给客户端的资源字节数，不包括响应头的大小
$bytes_sent                 #发送给客户端的总字节数
$http_referer               #记录从哪个页面链接访问过来的
$http_user_agent            #记录客户端浏览器相关信息
$http_x_forwarded_for       #记录客户端 IP 地址
$request_length             #请求的长度（包括请求行， 请求头和请求正文）
$request_filename           #请求的资源在系统中的路径，例如：/code/blog/index.html
$request_time               #请求花费的时间，单位为秒，精度毫秒
# 注:如果 Nginx 位于负载均衡器， nginx 反向代理之后， web 服务器无法直接获取到客 户端真实的 IP 地址。
# $remote_addr 获取的是反向代理的 IP 地址。 反向代理服务器在转发请求的 http 头信息中，
# 增加 X-Forwarded-For 信息，用来记录客户端 IP 地址和客户端请求的服务器地址。
```

第7章 Nginx 添加第三方模块

1.第三方模块介绍

```
nginx除了核心模块和官方的模块以外，还有很多第三方的模块，需要注意的是：添加新模块需要重新编译安装源码包
```

2.添加第三方模块实战 - echo模块

模块开源地址：

```
https://github.com/openresty/echo-nginx-module
```

编译安装步骤：

```
yum install openssl-devel pcre-devel gcc -y
groupadd www -g 1000
useradd www -s /sbin/nologin -M -u 1000 -g 1000
mkdir /data/soft -p
cd /data/soft/
wget http://nginx.org/download/nginx-1.19.0.tar.gz
tar zxvf nginx-1.19.0.tar.gz
cd nginx-1.19.0/
```

```
./configure \
--user=www \
--group=www \
--prefix=/opt/nginx-1.19.0 \
--with-http_stub_status_module \
--with-http_ssl_module \
--with-pcre
make && make install
cd /opt/
ln -s /opt/nginx-1.19.0 nginx
cp /opt/nginx/sbin/nginx /usr/sbin/
nginx -V
mkdir -p /opt/nginx/pid
mkdir -p /opt/nginx/conf.d
cat > /opt/nginx/conf/nginx.conf << 'EOF'
user  www;
worker_processes  1;

error_log  logs/error.log warn;
pid        pid/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  logs/access.log  main;
    sendfile    on;
    keepalive_timeout  65;
    include /opt/nginx/conf.d/*.conf;
}
EOF
nginx -t
```

添加第三方模块：

```
cd /data/soft/
yum install git -y
git clone https://github.com/openresty/echo-nginx-module.git
ll -d /data/soft/echo-nginx-module/
cd /data/soft/nginx-1.19.0
./configure \
--user=www \
--group=www \
--prefix=/opt/nginx-1.19.0 \
--with-http_stub_status_module \
--with-http_ssl_module \
--with-pcre \
--add-module=/data/soft/echo-nginx-module
make && make install
```

```
\cp /opt/nginx/sbin/nginx /usr/sbin/
nginx -V
```

创建子配置文件：

```
cat > /opt/nginx/conf.d/echo.conf << 'EOF'
server   {
    listen       80;
    server_name  www.echo.com;
    location / {
        echo "day day up!";
        echo $remote_addr;
        echo $http_user_agent;
    }
}
EOF
nginx -t
nginx -s reload
```

验证结果：

```
[root@web-7 ~]# curl 127.0.0.1
day day up!
127.0.0.1
curl/7.29.0
```

第8章 Nginx location匹配

使用 Nginx location 可以控制访问网站的路径, 但一个 server 可以有多个 location 配置, 多个 location 的优先级该如何区分

0.官方网址

```
http://nginx.org/en/docs/http/ngx_http_core_module.html#location
```

1.location语法介绍

```
Syntax: location [ = | ~ | ~* | ^~ ] uri { ... }
location @name { ... }
Default:    —
Context: server, location
```

2.location语法优先级

匹配符	匹配规则	优先级
=	精确匹配	1
^~	以某个字符串开头	2
~	区分大小写的正则匹配	3
~*	不区分大小写的正则匹配	4
!~	区分大小写不匹配的正则	5
!~*	不区分大小写不匹配的正则	6
/	通用匹配，任何请求都会匹配到	7

3.配置location匹配规则实战

```
cat >/etc/nginx/conf.d/01-www.conf << 'EOF'
server {
    listen       80;
    server_name  www.oldzhang.com;
    root   /usr/share/nginx/html/www;
    location / {
       return 200  "location / \n";
    }
    location = / {
        return 200 "location = \n";
    }

    location /documents/ {
        return 200 "location /documents/ \n";
    }
    location ^~ /images/ {
        return 200 "location ^~ /images/ \n";
    }
    location ~* \.(gif|jpg|jpeg)$ {
        return 200 "location ~* \.(gif|jpg|jpeg) \n";
    }
    access_log off;
}
EOF
```

4.测试location匹配规则

```
#精确匹配=/
[root@web01 ~]# curl www.oldzhang.com
location =

#没有满足的请求，所以匹配了/
[root@web01 ~]# curl www.oldzhang.com/oldzhang.html
location /

#匹配了/documents
[root@web01 ~]# curl www.oldzhang.com/documents/oldboy.html
location /documents/
```

```
#没有满足的条件，匹配/
[root@web01 ~]# curl www.oldzhang.com/oldboy/documents/oldboy.html
location /

#正则匹配了文件名
[root@web01 ~]# curl www.oldzhang.com/oldboy.jpg
location ~* \.(gif|jpg|jpeg)

#~*匹配正则不区分大小写优先于/documents
[root@web01 ~]# curl www.oldzhang.com/documents/oldboy.jpg
location ~* \.(gif|jpg|jpeg)

#^~优先匹配于~*
[root@web01 ~]# curl www.oldzhang.com/images/oldboy.jpg
location ^~ /images/
```

第9章 Nginx rewrite跳转

0.官方地址

```
https://nginx.org/en/docs/http/ngx_http_rewrite_module.html
```

1.ngx_http_rewrite_module模块介绍

```
在实际工作中，我们经常会遇到需要修改用户的URL的请求，例如如果用户访问http那就强制跳转到https，
或者判断用户浏览器类型，并跳转到不同的站点。
在Nginx中，跳转使用的是ngx_http_rewrite_module模块

ngx_http_rewrite_module 模块包含了多个指令，分别为：
break           #中断配置
if              #请求判断
set             #设置变量
return          #返回状态或URL
rewrite         #对用户请求的URL或URI进行重写
```

2.if指令

2.1 官方文档：

```
http://nginx.org/en/docs/http/ngx_http_rewrite_module.html#if
```

2.2 指令说明：

```
if用于条件判断，并且可以根据判断的结果执行不同的配置，if可以配置在server或location中。
```

2.3 指令语法：

```
if (匹配条件) {
    执行的动作
}
```

2.4 匹配规则：

```
=                   #比较变量和字符串是否相等，相等则为true，不相等则为false
!=                  #比较变量和字符串是否不相等，不相等则为true，不相等则为false

~                   #区分大小写的正则匹配，匹配上则为true，不匹配则为false
!~                  #区分大小写的正则匹配，不匹配则为true，匹配则为false

~*                  #不区分大小写的正则匹配，匹配上则为true，不匹配则为false
!~*                 #不区分大小写的正则匹配，不匹配则为true，匹配则为false

-f和!-f           #判断请求的文件是否存在
-d和!-d           #判断请求的目录是否存在
-e和!-e           #判断请求的文件/目录/软链接是否存在
-x和!-x           #判断请求的文件是否具有可执行权限
```

2.5 创建测试配置文件：

```
cat >/opt/nginx/conf.d/if.conf << 'EOF'
server {
    listen       8080;
    server_name  www.oldzhang.com;
    root   /usr/share/nginx/html/www;

    location /mall {
        #客户端类型完全匹配iphone
        if ($http_user_agent = iphone) {
           echo "agent is iphone";
        }

        #客户端类型区分大小写匹配
        if ($http_user_agent ~ Android) {
           echo "agent is Android";
        }

        #客户端类型不区分大小写匹配
        if ($http_user_agent ~* Chrome) {
           echo "agent is Chrome";
        }

        #如果不是GET则输出
        if ($request_method != GET){
           echo "method is not GET";
        }

        #如果是IE浏览器，直接返回404
        if ($http_user_agent ~* IE){
           return 404;
        }
```

```
32
33          #都没匹配上则执行下面语句
34          echo "not match";
35          echo "agent ==> $http_user_agent";
36          echo "request ==> $request_method";
37      }
38  }
39  EOF
```

2.6 测试方法：

1.默认访问

```
1  [root@web-7 ~]# curl 127.0.0.1:8080/mall
2  not match
3  agent ==> curl/7.29.0
4  request ==> GET
```

2.测试完全匹配=

```
1  [root@web-7 ~]# curl -A "iphone" 127.0.0.1:8080/mall
2  agent is iphone
3
4  [root@web-7 ~]# curl -A "iphoneeee" 127.0.0.1:8080/mall
5  not match
6  agent ==> iphoneeee
7  request ==> GET
8
9  [root@web-7 ~]# curl -A "Iphone" 127.0.0.1:8080/mall
10 not match
11 agent ==> Iphone
12 request ==> GET
```

3.测试区分大小写匹配~

```
1  [root@web-7 ~]# curl -A "Android" 127.0.0.1:8080/mall
2  agent is Android
3
4  [root@web-7 ~]# curl -A "android" 127.0.0.1:8080/mall
5  not match
6  agent ==> android
7  request ==> GET
8
9  [root@web-7 ~]# curl -A "Androidd" 127.0.0.1:8080/mall
10 agent is Android
11
12 [root@web-7 ~]# curl -A "AAndroidD" 127.0.0.1:8080/mall
13 agent is Android
14
15 [root@web-7 ~]# curl -A "AndXroid" 127.0.0.1:8080/mall
16 not match
17 agent ==> AndXroid
18 request ==> GET
```

4.测试不区分大小写~*

```
[root@web-7 ~]# curl -A "Chrome" 127.0.0.1:8080/mall
agent is Chrome

[root@web-7 ~]# curl -A "chrome" 127.0.0.1:8080/mall
agent is Chrome

[root@web-7 ~]# curl -A "ChRoMe" 127.0.0.1:8080/mall
agent is Chrome

[root@web-7 ~]# curl -A "AaChRoMe" 127.0.0.1:8080/mall
agent is Chrome

[root@web-7 ~]# curl -A "AaChRoMeDd" 127.0.0.1:8080/mall
agent is Chrome
```

3.return指令

3.1 官方文档

```
http://nginx.org/en/docs/http/ngx_http_rewrite_module.html#return
```

3.2 指令说明

```
1.return的作用是当匹配上之后直接返回响应状态码或者重写URL或者返回状态码的同时显示文本。
2.return后的指令不会在执行。
```

3.3 指令语法

```
reutrn可以作用于 server, location, if
return code [text];
return code URL;
return URL;
```

3.4 实验配置

```
server {
    listen       8090;
    server_name  www.oldzhang.com;
    root   /usr/share/nginx/html/www;

    location / {
        echo "this is index";
    }

    location /mall {
        #客户端类型完全匹配iphone
        if ($http_user_agent = iphone) {
           return 200 "agent is iphone \n";
        }

        #客户端类型区分大小写匹配
        if ($http_user_agent ~ Android) {
           return 200 "agent is Android \n";
```

```
        }

        #客户端类型不区分大小写匹配
        if ($http_user_agent ~* Chrome) {
           return 200 "agent is Chrome \n";
        }

        #如果是IE浏览器，直接301跳转到/dog
        if ($http_user_agent ~* IE){
           return 301 http://www.oldzhang.com:8090/dog;
        }

        #都没匹配上则跳转到首页
        return 301 http://www.oldzhang.com:8090/;
    }

    location /dog {
        return 444 "wang wang! \n";
    }
}
```

3.5 测试方法

1.不指定客户端跳转到首页

```
[root@web-7 ~]# curl -L http://127.0.0.1:8090/mall
this is index
```

2.指定客户端跳转

```
[root@web-7 ~]# curl -L -A "iphone" http://127.0.0.1:8090/mall
agent is iphone

[root@web-7 ~]# curl -L -A "Chrome" http://127.0.0.1:8090/mall
agent is Chrome
```

3.指定客户端为IE，跳转到另一个页面

```
[root@web-7 ~]# curl -L -A "IE" http://127.0.0.1:8090/mall
wang wang!
```

4.set指令

4.1 官方文档

```
https://nginx.org/en/docs/http/ngx_http_rewrite_module.html#set
```

4.2 指令说明

```
为指定变量设置一个值。该值可以包含文本，变量及其组合
```

4.3 指令语法

```
set可以作用于 server, location, if
set 变量名 变量值;
```

4.4 实验配置

```
server {
    listen       8000;
    server_name  www.oldzhang.com;
    root   /usr/share/nginx/html/www;
    set $nb_agent $http_user_agent;

    location /{
        if ($nb_agent ~* IE){
           return 301 http://www.oldzhang.com:8090/dog;
        }
        return 200 "this is index \n";
    }

    location /dog {
        return 444 "wang wang! \n";
    }
}
```

4.5 测试方法

1.默认访问会跳转到首页

```
[root@web-7 ~]# curl 127.0.0.1:8000/
this is index
```

2.指定客户端类型为IE会跳转到dog

```
[root@web-7 ~]# curl -L -A "IE" 127.0.0.1:8000/
wang wang!
```

5.break指令

5.1 官方文档

```

```

5.2 指令说明

```
终止break所处的作用域的配置，执行完break后，其所在的区域后面的ngx_http_rewrite_module模块的指令不在执行。
注意，只是不执行ngx_http_rewrite_module模块的指令，其他指令不受影响
```

5.3 指令语法

```
break;
可以作用于：server, location, if
```

5.4 实验配置

```
cat > /opt/nginx/conf.d/www.conf << 'EOF'
server   {
    listen       80;
    server_name  www.oldboyedu.com;
    location / {
        set $name  oldboyedu;
        echo $name;
        break;
        set $name  szoldboy;
        echo $name;
    }
}
EOF
```

5.5实验结果

```
[root@web-7 ~]# curl www.oldboyedu.com
oldboyedu
oldboyedu
```

6.rewrite指令--重点掌握

6.1 官方文档

```
https://nginx.org/en/docs/http/ngx_http_rewrite_module.html#rewrite
```

6.2 指令说明

```
1.rewrite指令可以基于用户请求的RUI通过正则表达式的匹配来进行改写。
2.rewrite指令可以存在多条，并且按照次序依次执行
3.rewrite指令可以根据flag标志符对指令进一步处理
4.如果替换字符串以“http://”、“https://”或“$scheme”开头，则处理将停止，并将重定向返回到客户端。
```

6.3 指令语法

```
rewrite regex replacement [flag];
rewrite 匹配规则 重写指令 [标志符];

rewrite指令可以作用于server, location, if
```

6.4 flag标识符说明

flag作用说明：

```
redirect        返回302临时重定向，浏览器地址会显示跳转后的URL地址，浏览器不会缓存DNS记录
permanent       返回301临时重定向，浏览器地址会显示跳转后的URL地址，浏览器会缓存DNS记录

last            本条规则匹配完成后，继续向下匹配新的location URI规则
break           本条规则匹配完成即终止，不再匹配后面的任何规则
```

使用场景区分：

```
1.使用redirect和permanent参数，浏览器会显示跳转后的URL，服务端将改写后的URL返回给客户端，由客户端重新发起新的请求。
2.使用last和break，浏览器显示的还是原来的URL，由服务器内部完成跳转。
```

last和break区别：

```
last标记在本条规则匹配完成后，对其所在的server标签重新发起修改后的URL请求，再次匹配location。
break标记则会在本条规则匹配完成后，终止匹配，不会在匹配后面的规则。
```

6.4 redirect和permanent实验

6.4.1 permanent 301 永久跳转 实验

permanent 301跳转需求：

```
用户访问 www.oldboy.com 跳转到 blog.oldboy.com
```

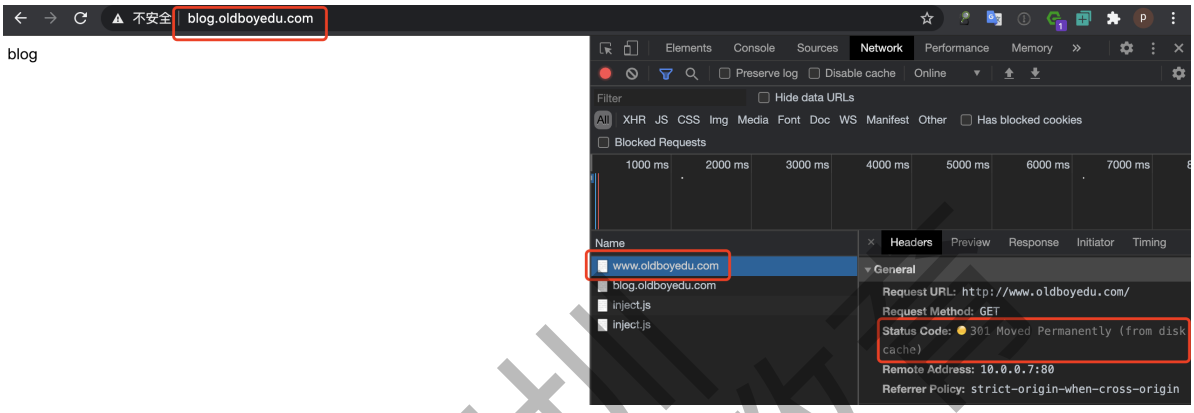
permanent 301配置：

```
cat > /opt/nginx/conf.d/www.conf << 'EOF'
server   {
    listen       80;
    server_name  www.oldboyedu.com;
    location / {
        root   /code/www;
        index  index.html;
        rewrite / http://blog.oldboyedu.com permanent;
    }
}
EOF

cat > /opt/nginx/conf.d/blog.conf << 'EOF'
server   {
    listen       80;
    server_name  blog.oldboyedu.com;
    location / {
        root   /code/blog;
        index  index.html;
    }
}
```

```
EOF

mkdir /code/{www,blog} -p
echo www > /code/www/index.html
echo blog > /code/blog/index.html

nginx -t
nginx -s reload
```

permanent 301测试：

```
浏览输入 www.oldboyedu.com 后会跳转到 blog.oldboyedu.com
headers头部信息显示from disk cache
```



6.4.2 redirect 302 临时跳转 实验

redirect 302跳转需求：

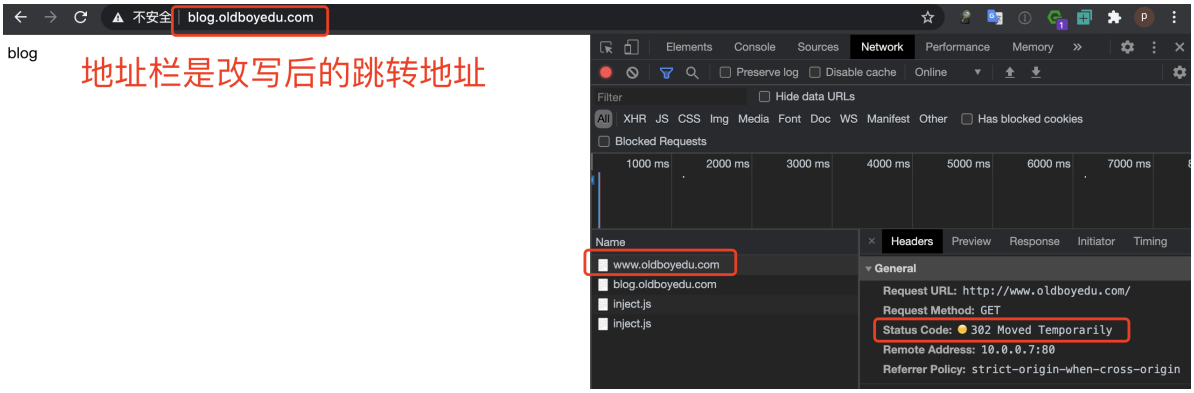
```
用户访问www.oldboy.com 跳转到 blog.oldboy.com
```

redirect 302配置：

```
cat > /opt/nginx/conf.d/www.conf << 'EOF'
server   {
    listen       80;
    server_name  www.oldboyedu.com;
    location / {
        root   /code/www;
        index  index.html;
        rewrite / http://blog.oldboyedu.com redirect;
    }
}
EOF
```

redirect 302测试：

```
浏览输入 www.oldboyedu.com 后会跳转到 blog.oldboyedu.com
headers头部信息没有 from disk cache ，因为302跳转浏览器不缓存DNS记录
```



6.5 break和last实验

6.5.1 last实验

last 实验需求：

```
访问AAA 跳转到BBB 然后再跳转到CCC
```

last 配置文件：

```
cat > /opt/nginx/conf.d/www.conf << 'EOF'
server   {
    listen       80;
    server_name  www.oldboyedu.com;
    location /CCC {
        return 200 "CCCC \n";
    }

    location /BBB {
        rewrite ^/BBB/(.*)$ /CCC/$1 last;
    }

    location /AAA {
        rewrite ^/AAA/(.*)$ /BBB/$1 last;
    }
}
EOF
```

last 测试

```
[root@web-7 ~]# curl -L http://www.oldboyedu.com/AAA/index.html
CCCC
```

6.5.2 break实验

break 实验需求：

```
访问AAA后直接访问跳转后的资源，不再匹配BBB和CCC的location
```

break 配置文件：

```
cat > /opt/nginx/conf.d/www.conf << 'EOF'
server   {
```

```
    listen       80;
    server_name  www.oldboyedu.com;
    location /CCC {
        return 200 "CCCC \n";
    }

    location /BBB {
        rewrite ^/BBB/(.*)$ /CCC/$1 last;
    }

    location /AAA {
        root /code/www;
        index index.html;
        rewrite ^/AAA/(.*)$ /BBB/$1 break;
    }
}
EOF
```

创建测试文件：

```
[root@web-7 ~]# mkdir /code/www/AAA
[root@web-7 ~]# mkdir /code/www/BBB
[root@web-7 ~]# mkdir /code/www/CCC
[root@web-7 ~]# echo AAA > /code/www/AAA/index.html
[root@web-7 ~]# echo BBB > /code/www/BBB/index.html
[root@web-7 ~]# echo CCC > /code/www/CCC/index.html
[root@web-7 ~]# tree /code/www/
/code/www/
├── AAA
|   └── index.html
├── BBB
|   └── index.html
├── CCC
|   └── index.html
└── index.html
```

测试访问效果：

```
[root@web-7 ~]# curl -L http://www.oldboyedu.com/AAA/index.html
BBB
```

6.6 生产跳转场景

6.6.1 URL跳转

需求：

```
用户访问 www.oldboyedu.com/blog 跳转到 blog.oldboyedu.com/
```

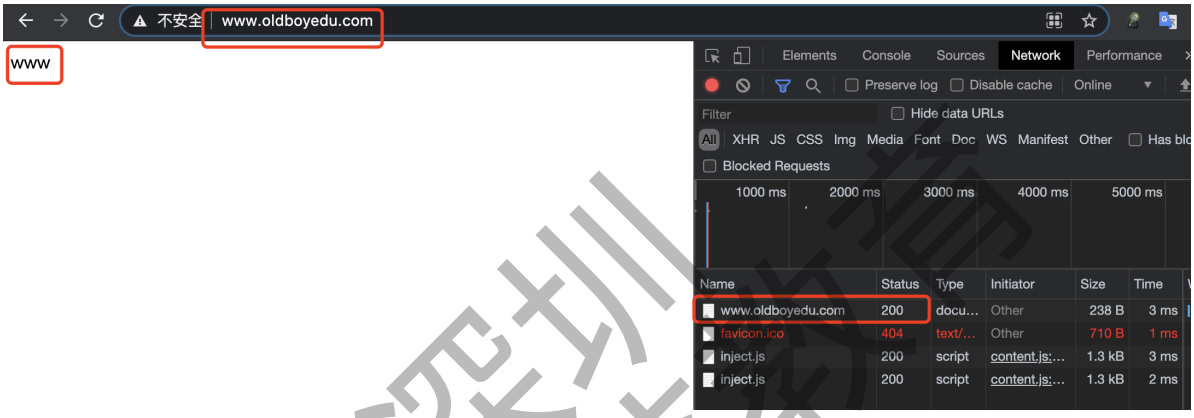
实验配置：

```
cat > /opt/nginx/conf.d/www.conf << 'EOF'
server   {
    listen       80;
    server_name  www.oldboyedu.com;
```

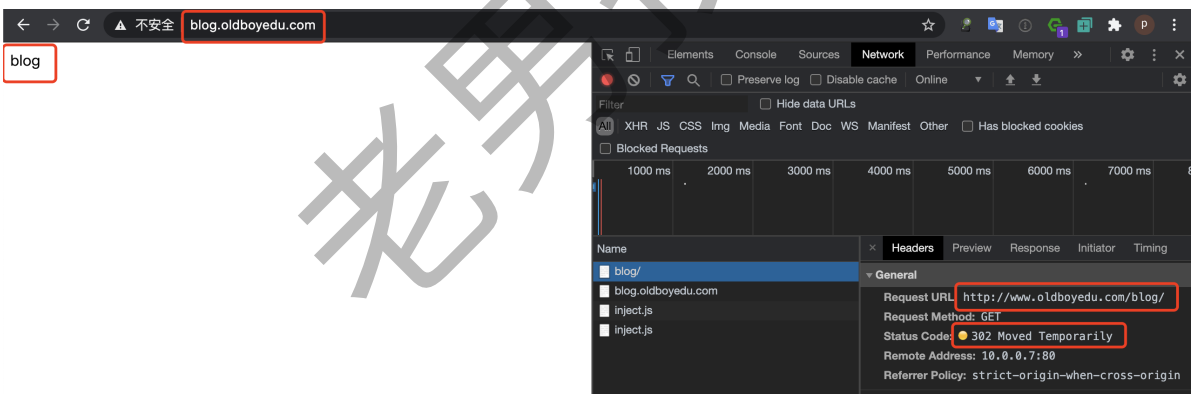
```
    location / {
        root /code/www;
        index index.html;
    }

    location /blog/ {
        rewrite ^/blog/(.*)$ http://blog.oldboyedu.com/$1 redirect;
    }
}
EOF

nginx -t
nginx -s reload
```

跳转结果：

正常访问由 location提供服务



访问blog会被跳转到blog.oldboyedu.com



6.6.2 http跳转https

需求：

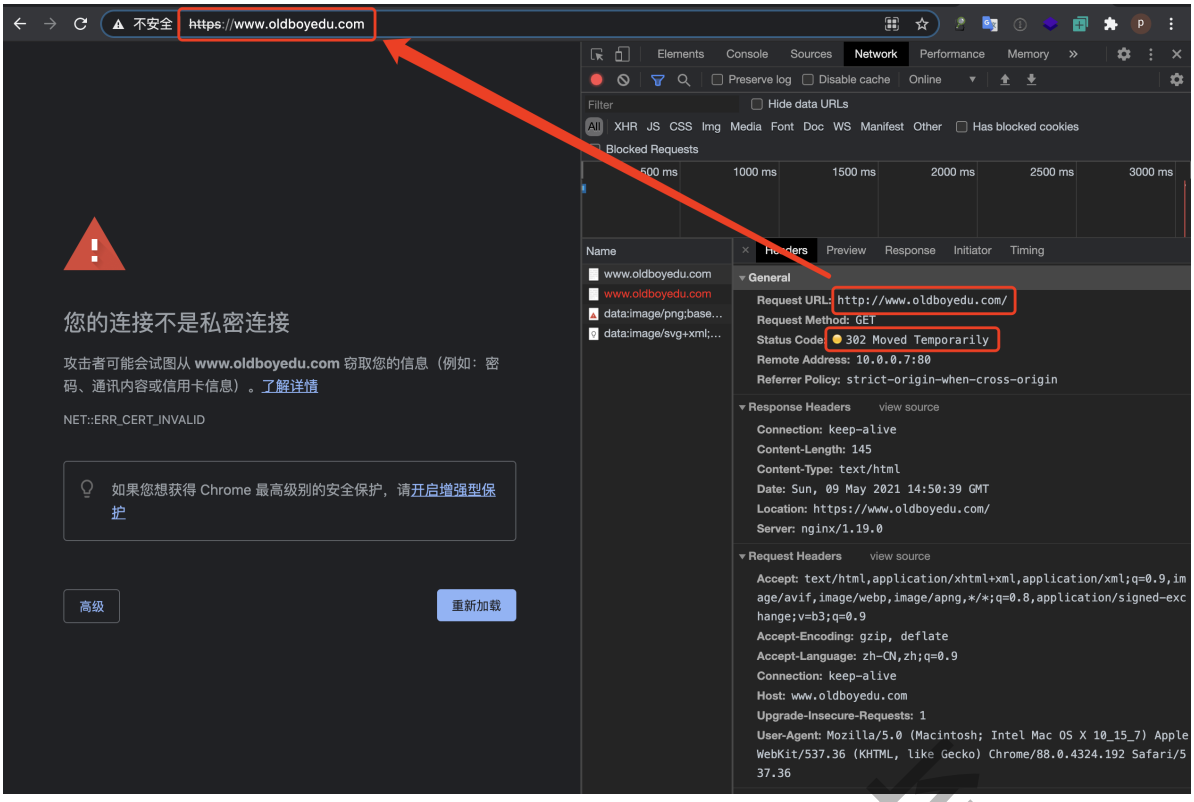
```
用户访问 http://www.oldboyedu.com
跳转到   https://www.oldboyedu.com
```

配置：

```
#生成证书并拷贝到指定位置
mkdir /opt/nginx/ssl_key
cd /opt/nginx/ssl_key
```

```
openssl genrsa -idea -out server.key 2048
输入密码:

openssl req -days 36500 -x509 -sha256 -nodes -newkey rsa:2048 -keyout server.key -out server.crt
-----------------
CN
SZ
SZ
linux6
SA
linux6
linux6@qq.com
-----------------

#检查证书文件
[root@web-7 ~]# ll /opt/nginx/ssl_key/
总用量 8
-rw-r--r-- 1 root root 1367 5月   9 22:47 server.crt
-rw-r--r-- 1 root root 1708 5月   9 22:47 server.key

#创建配置文件
cat > /opt/nginx/conf.d/www.conf << 'EOF'
server {
   listen 80;
   server_name  www.oldboyedu.com;
   rewrite ^(.*) https://$server_name$1 redirect;
}

server {
    listen 443 ssl;
    server_name www.oldboyedu.com;
    ssl_certificate /opt/nginx/ssl_key/server.crt;
    ssl_certificate_key /opt/nginx/ssl_key/server.key;
    location / {
        root /code/www;
        index index.html;
    }
}
EOF

#测试并重启
nginx -t
nginx -s stop
nginx
```

访问测试:



6.6.3 URL拼接跳转

需求:

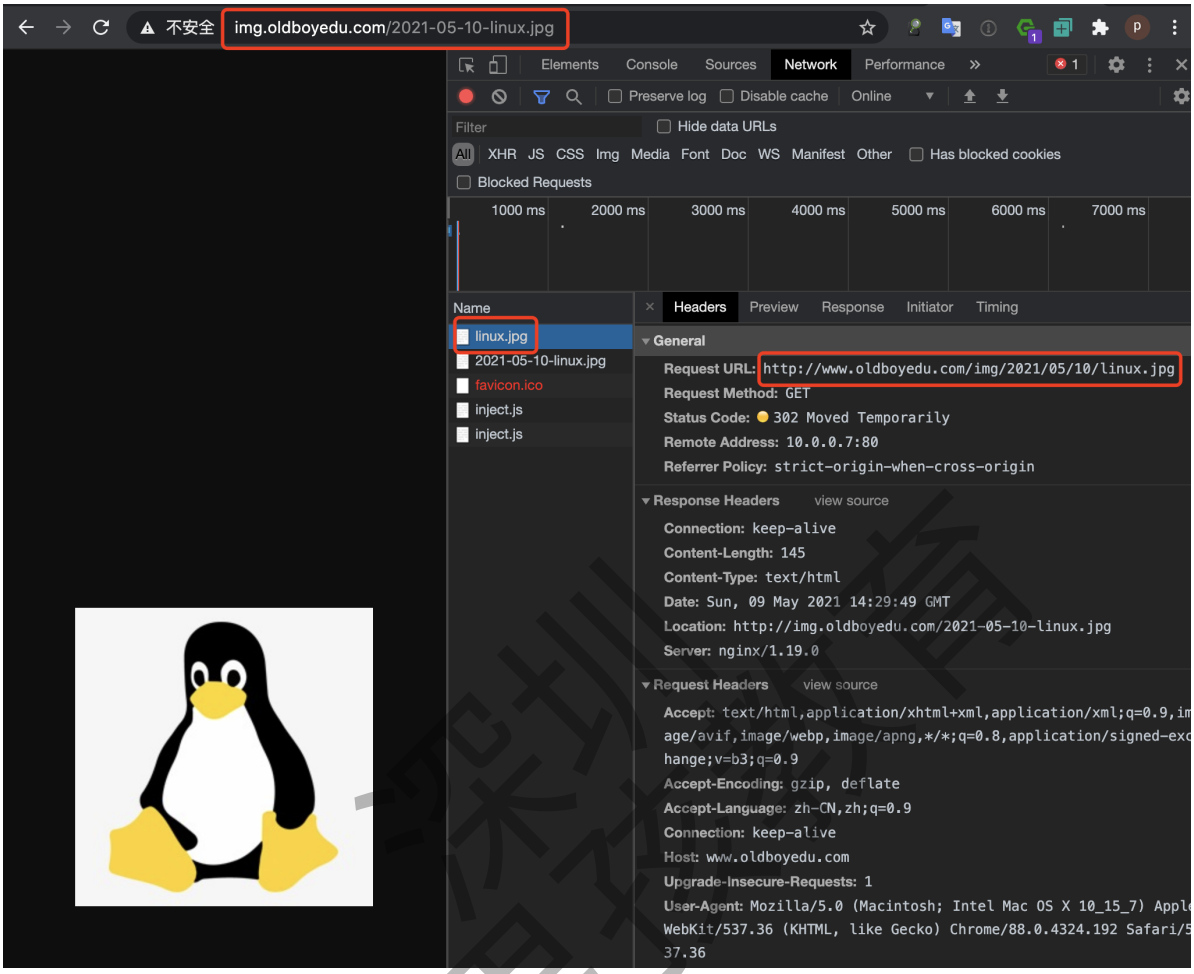
```
用户访问 www.oldboyedu.com/img/2021/05/10/linux.jpg
跳转到   img.oldboyedu.com/2021-05-10-linux.jpg
```

配置:

```
cat > /opt/nginx/conf.d/www.conf << 'EOF'
server   {
    listen       80;
    server_name  www.oldboyedu.com;
    location / {
        root /code/www;
        index index.html;
    }

    location /img/ {
        rewrite ^/img/(\d+)/(\d+)/(\d+)/(.*).jpg$
http://img.oldboyedu.com/$1-$2-$3-$4.jpg redirect;
    }
}
EOF

cat > /opt/nginx/conf.d/img.conf << 'EOF'
server   {
    listen       80;
    server_name  img.oldboyedu.com;
    location / {
        root /code/img;
        index index.html;
    }
}
```

```
EOF

nginx -t
nginx -s reload
```

测试效果：



6.6.4 带参数跳转

此类跳转比较特殊，有两种需求

需求1:

```
访问 www.oldboyedu.com/img/2021/05/10/linux.jpg
跳转到 img.oldboyedu.com/index.php?name=2021-05-10-linux.jpg
```

需求2:

```
访问    www.oldboyedu.com/index.php?name=2021-05-10-linux.jpg
跳转到 img.oldboyedu.com/img/2021/05/10/linux.jpg
```

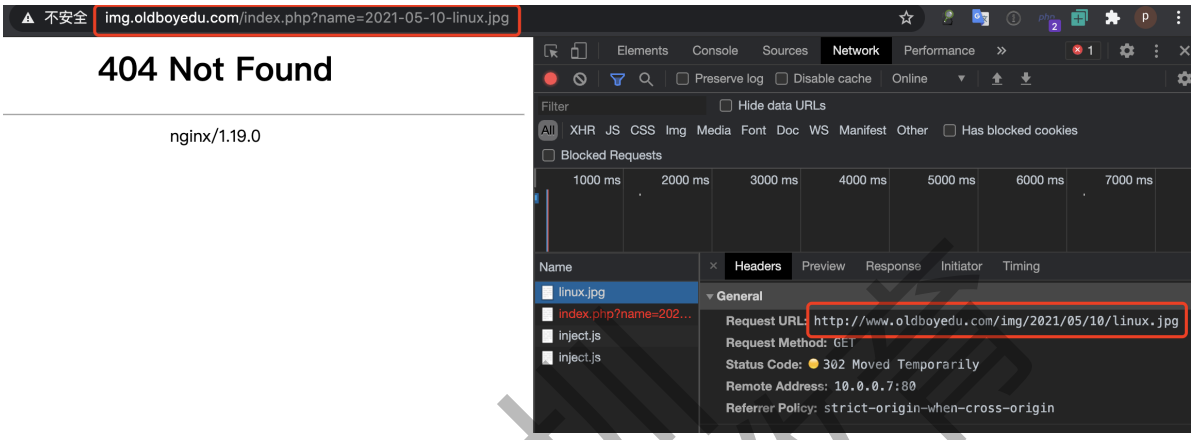
其中需求2比较难实现。

需求1配置：

```
cat > /opt/nginx/conf.d/www.conf << 'EOF'
server   {
    listen       80;
    server_name  www.oldboyedu.com;
```

```
    location / {
        root /code/www;
        index index.html;
    }

    location /img/ {
        rewrite ^/img/(\d+)/(\d+)/(\d+)/(.*).jpg$
http://img.oldboyedu.com/index.php?name=$1-$2-$3-$4.jpg redirect;
    }
}
EOF
```

测试效果：



需求2配置：

```

```

第10章 制作内网YUM仓库

1.为什么需要私有YUM仓库

```
1.下载速度慢
2.需要有外网
3.有些Base源和epel源软件没有，需要单独创建下载源
```

2.需要的软件

```
createrepo
nginx
```

3.配置nginx索引模块

```
cat >/etc/nginx/conf.d/index.conf <<EOF
server {
    listen       80;
    server_name  yum.mysun.com;
    location / {
        autoindex on;
        autoindex_exact_size off;
        autoindex_localtime on;
```

```
        autoindex_format html;
        charset utf-8,gbk;
        root   /data/yum;
        index  index.html index.htm;
    }
}
EOF
```

4.安装createrepo

```
yum install createrepo -y
```

5.准备软件仓库目录并下载需要的软件

```
yum install --downloadonly --downloaddir=/data/yum nginx screen vim tree -y
```

6.生成yum元数据

```
createrepo /data/yum
```

7.客户端生成本地源

```
cat >/etc/yum.repos.d/local.repo <<EOF
[local]
name=local
enable=1
gpgcheck=0
baseurl=http://10.0.0.61
```

8.客户端测试安装

```
yum makecache
yum search nginx
yum install nginx
```

9.更新软件包的操作步骤：

第一种方法：真实下载

1.打开yum缓存

```
[root@web01 /data/yum]# grep "keepcache" /etc/yum.conf
keepcache=1
```

2.清空原来的缓存

```
yum clean all
```

3.下载软件

```
yum remove php-mysql-5.4 php php-fpm php-common
rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
rpm -Uvh https://mirror.webtatic.com/yum/el7/webtatic-release.rpm
yum install php71w php71w-cli php71w-common php71w-devel php71w-embedded php71w-gd php71w-mcrypt php71w-mbstring php71w-pdo php71w-xml php71w-fpm php71w-mysqlnd php71w-opcache -y
```

4.移动已经缓存下来的rpm包到yum仓库目录

```
find /var/cache/yum/ -type f -name "*.rpm"|xargs mv -t /data/yum/
```

5.生成新的yum元数据

```
createrepo --update /data/yum/
```

第二种方法：只下载不安装

```
yum install --downloadonly --downloaddir=/data/yum php71w php71w-cli php71w-common php71w-devel php71w-embedded php71w-gd php71w-mcrypt php71w-mbstring php71w-pdo php71w-xml php71w-fpm php71w-mysqlnd php71w-opcache
```