Nginx入门之静态资源与动态访问分离 原

Fraud 发布于 2016/01/08 15:15 字数 1249 阅读 3586 收藏 11 点赞 1 评论 0

Nginx动静分离 Nginx的location配置

在上一篇中，我们配置了

```
location / {
    proxy_pass http://localhost:8080 ;
}
```

这其实就是将Nginx的所有请求都转发给

```
http://localhost:8080
```

如果我们需要将web服务请求转发，而其他的静态资源（比如jpg、css等）不转发直接由nigx处理的话，那效率应该又能提高不少。那么这个其实就是对location进行配置了。下面对网上查阅的关于这个配置的资料做些整理，大家可以自己实践

语法规则： location [=|~|~*|^~] /uri/ { … }

= 开头表示精确匹配

^~ 开头表示uri以某个常规字符串开头，理解为匹配 url路径即可。nginx不对url做编码，因此请求为/static/20%/aa，可以被规则^~ /static/ /aa匹配到（注意是空格）。

~ 开头表示区分大小写的正则匹配

~* 开头表示不区分大小写的正则匹配

!~和!~*分别为区分大小写不匹配及不区分大小写不匹配 的正则

/ 通用匹配，任何请求都会匹配到。

多个location配置的情况下匹配顺序为：

首先匹配 =，其次匹配^~, 其次是按文件中顺序的正则匹配，最后是交给 / 通用匹配。当有匹配成功时候，停止匹配，按当前匹配规则处理请求。

例子，有如下匹配规则：

```
location = / {
   #规则A
}
location = /login {
   #规则B
}
location ^~ /static/ {
   #规则C
}
location ~ \.(gif|jpg|png|js|css)$ {
   #规则D
}
location ~* \.png$ {
   #规则E
}
location !~ \.xhtml$ {
   #规则F
}
location !~* \.xhtml$ {
   #规则G
}
location / {
   #规则H
}
```

那么产生的效果如下：

访问根目录/， 比如http://localhost/ 将匹配规则A

访问 http://localhost/login 将匹配规则B，http://localhost/register 则匹配规则H

访问 http://localhost/static/a.html 将匹配规则C

访问 http://localhost/a.gif, http://localhost/b.jpg 将匹配规则D和规则E，但是规则D顺序优先，规则E不起作用，而 http://localhost/static/c.png 则优先匹配到规则C

访问 http://localhost/a.PNG 则匹配规则E，而不会匹配规则D，因为规则E不区分大小写。

访问 http://localhost/a.xhtml 不会匹配规则F和规则G，http://localhost/a.XHTML不会匹配规则G，因为不区分大小写。规则F，规则G属于排除法，符合匹配规则但是不会匹配到，所以想想看实

际应用中哪里会用到。

访问 http://localhost/category/id/1111 则最终匹配到规则H，因为以上规则都不匹配，这个时候应该是nginx转发请求给后端应用服务器，比如FastCGI（php），tomcat（jsp），nginx作为方向代理服务器存在。

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，官网如是说。
#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://tomcat:8080/index
}
# 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 ^~ /static/ {
    root /webroot/static/;
}
location ~* \.(gif|jpg|jpeg|png|css|js|ico)$ {
    root /webroot/res/;
}
#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/
}
```

三、ReWrite语法

last 一 基本上都用这个Flag。

break 一 中止Rewirte，不在继续匹配

redirect 一 返回临时重定向的HTTP状态302

permanent 一 返回永久重定向的HTTP状态301

1、下面是可以用来判断的表达式：

–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

2、下面是可以用作判断的全局变量

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

$host：localhost

$server_port：88

$request_uri：http://localhost:88/test1/test2/test.php

$document_uri：/test1/test2/test.php

$document_root：D:\nginx/html

$request_filename：D:\nginx/html/test1/test2/test.php

四、Redirect语法

```
server {
    listen 80;
    server_name start.igrow.cn;
    index index.html index.php;
    root html;
    if ($http_host !~ “^star\.igrow\.cn$&quot {
        rewrite ^(.*) http://star.igrow.cn$1 redirect;
    }
}
```

五、防盗链

```
location ~* \.(gif|jpg|swf)$ {
    valid_referers none blocked start.igrow.cn sta.igrow.cn;
    if ($invalid_referer) {
        rewrite ^/ http://$host/logo.png;
    }
}
```

六、根据文件类型设置过期时间

```
location ~* \.(js|css|jpg|jpeg|gif|png|swf)$ {
    if (-f $request_filename) {
        expires 1h;
        break;
    }
}
```

七、禁止访问某个目录

```
location ~* \.(txt|doc)${
    root /data/www/wwwroot/linuxtone/test;
    deny all;
}
```

一些可用的全局变量:

```
$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
```


Fraud