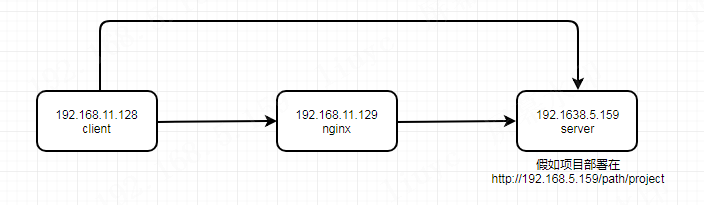
# nginx反向代理个人总结

需求：（1）客户端直接访问服务端能正常访问，（2）同时客户端通过反向代理访问服务端也要正常访问。



对于直接访问（即client直接访问server），web审批项目不需要更改代码。

对于nginx反向代理的情况（即client访问nginx），下面会分路径相同于不相同的情况进行说明。

词语解析：

1、路径相同说明：

假如项目部署在5.100/a/b/c，nginx在6.200中。

server\_name 6.200;

location / {

proxy\_pass http://5.100/;

}

location /a/ {

proxy\_pass http://5.100/a/;

}

location /a/b/ {

proxy\_pass http://5.100/a/b/;

}

location /a/b/c/ {

proxy\_pass http://5.100/a/b/c/;

}

location /a/b/c/ {

proxy\_pass http://5.100;

}

location /b/c/ {

proxy\_pass http://5.100/a;

}

......

上面的情况都是可以通过6.200/a/b/c/和5.100/a/b/c来访问项目，url地址就只有ip不相同（端口也可以不相同），其他uri段必须要一致。这里就把它称为路径相同。

2、举例路径不相同的情况：

location /a/ {

proxy\_pass http://5.100/a/b/c/;

}

这个时候我们通过6.200/a/和5.100/a/b/c可以访问项目，是允许这样配置的。但uri段不一样，此文档把这种情况称为路径不相同。

location /x/ {

proxy\_pass http://5.100/;

}

通过6.200/x/a/b/c与5.100/a/b/c访问，这样也是路径（uri）不相同，但这种后段路径完全相同的可以通过rewrite来处理。

## 1、路径相同的情况

nginx配置如下：

upstream proxyweb {

server 192.168.5.159; #需要反向代理到的web服务器

}

server {

listen 80;

server\_name 192.168.11.129;

location / {

proxy\_pass http://proxyweb;

#Proxy Settings

proxy\_set\_header Host $host:$server\_port;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_next\_upstream error timeout invalid\_header http\_500 http\_502 http\_503 http\_504;

proxy\_max\_temp\_file\_size 0;

proxy\_connect\_timeout 90;

proxy\_send\_timeout 90;

proxy\_read\_timeout 90;

proxy\_buffer\_size 4k;

proxy\_buffers 4 32k;

proxy\_busy\_buffers\_size 64k;

proxy\_temp\_file\_write\_size 64k;

}

location ~\* \.(eot|otf|ttf|woff)$ {

proxy\_pass http://proxyweb;

}

}

几个重要的配置用红色字体表明，然后后边文档都会省略部分代码。配置完成启动nginx后，就能这样访问项目：

http://192.168.11.129/path/project

这个时候nginx把匹配的“/”及后面所有内容“path/project”，一起追加在proxy\_pass后面（[相对路径和绝对路径](https://xiexianbin.cn/nginx/2018/01/31/nginx-proxy_pass)），即

http://192.168.5.159/path/project

这种匹配方式直接访问192.168.11.129，转到的是192.168.5.159，如果5.159是tomcat服务器，就会访问tomcat主页，如果是wamp服务器，就显示wamp主页，很容易理解，所以我们访问192.168.11.129的时候，通过path来寻找project项目：http://192.168.11.129/path/project，所以是可以正常访问。

上述情况产生的问题：当nginx想一个域名和端口，反向代理多个项目的时候，那我们通常不会监听“/”这个路径了，一般是作为没有找到匹配的路径才匹配“/”，如：

location /aProject {}

location /bProject {}

location / {#默认}

配置要修改为：

location /path/project/ {

proxy\_pass http://proxyweb/path/project/;

}

#另一个项目:

location /path/project2/ {

proxy\_pass http://proxyweb/path/project2/;

}

这个时候还是通过同一地址访问：

http://192.168.11.129/path/project 和 http://192.168.5.159/path/project

补充项目的base\_url构造代码

$baseUrl = str\_replace('\\', '/', dirname($\_SERVER['SCRIPT\_NAME']));

$baseUrl = trim($baseUrl, '/');

$baseUrl .= '/';

$proto = is\_https() ? 'https://' : 'http://';

$config['base\_url'] = $proto.$\_SERVER['HTTP\_HOST'] . '/' . $baseUrl;

这时$config['base\_url'] = http://192.168.11.129/path/project，访问静态资源static就会拼接在base\_url后面。能正常请求。

**总结：**以上在路径相同的情况下，不用修改web审批的代码也能满足需求的两种途径访问项目

## 2、路径不相同的情况

如：

location /path {

proxy\_pass http://proxyweb/path/project/;

}

这个时候我们可以通过192.168.11.129/path或者192.168.5.159/path/project访问项目，但对于前者，会发现css、js等都加载不到，通过控制台会看到请求的地址为192.168.11.129/path/project/static/css/bootstrap.css。这个地址是由第1点中提到的base\_url构造代码中拼接出来的。

然后nginx把

192.168.11.129/path/project/static/css/bootstrap.css

转发到

192.168.5.159/path/project//project/static/css/bootstrap.css

就会找不到文件。程序就无法正常运行。

可以在apache 的access log文件中看到



一般如果proxy\_pass最后有/，那么location建议最后也带有/，否则就会出现 // 的情况

对于以上情况，代理配置没有什么可以改的，能改的只有base\_url的构造代码，但还是没想到怎样在

访问 http://192.168.11.129 然后 base\_url = http://192.168.11.129；

访问http://192.168.5.159/path/project然后base\_url = http://192.168.5.159/path/project。

即$proto.$\_SERVER['HTTP\_HOST'] . '/' . $baseUrl;在什么时候需要拼接$baseUrl，在什么时候不拼接，后端完全不知道。

## 3、路径不相同的解决方法

增加header作为后台接收参数。

location /IPG/ {

proxy\_pass http://proxyweb/path/project/;

# （1）加到 $\_SERVER['HTTP\_IPG\_PATH'] 中，自动加HTTP

# 测试过 HTTP-IPG-PATH，key会变为 HTTP\_HTTP\_IPG\_PATH

# 应该是php自动加的，网上看到的NGINX获取php的header都是要去掉HTTP\_字符串

# （2）也可以getallheaders()['IPG-PATH']获取

set $IPG\_PATH /IPG/;

proxy\_set\_header IPG-PATH $IPG\_PATH;

}

通过上面的配置，php的$\_SERVER会多一个HTTP\_IPG\_PATH的key，就可以通过$\_SERVER['HTTP\_IPG\_PATH']来获取。那么我们可以把base\_url的构造代码改为：

$baseUrl = str\_replace('\\', '/', dirname($\_SERVER['SCRIPT\_NAME']));

$baseUrl = trim($baseUrl, '/');

$baseUrl .= '/';

$proto = is\_https() ? 'https://' : 'http://';

$config['base\_url'] = $proto.$\_SERVER['HTTP\_HOST'] . '/' . $baseUrl;

// 经过nginx的时候，在nginx中定义了一个path，不经过nginx就不存在

// 模仿baseUrl处理斜杠

if(isset($\_SERVER['HTTP\_IPG\_PATH'])){

$nginxUrl = trim($\_SERVER['HTTP\_IPG\_PATH'], '/');

$nginxUrl .= '/';

$config['base\_url'] = $proto.$\_SERVER['HTTP\_HOST'] . '/' . $nginxUrl;

}

这个时候，我们访问192.168.11.129/IPG会显示project项目，查看base\_url()为http://192.168.11.129/IPG，访问192.168.5.159/path/project时base\_url()为http://192.168.5.159/path/project。

这样http://192.168.11.129/IPG请求资源的路径会为http://192.168.11.129/IPG/static/....css，经过nginx也能正常转发。

注意配置时IPG-PATH的路径要对应项目路径，可以有很多的配置方式，再举一例：

location /IPG/ {

proxy\_set\_header IPG-PATH /IPG/path/project/;

proxy\_pass http://proxyweb/;

}

**总结：**对于这种方法，我个人理解是与X-Forwarded-For、X-Real-IP等同一个意思，只是X-Forwarded-For、X-Real-IP它们写入标准中，然后每个系统都按标准名称来获取它们的值，那我们也可以理解为IPG-PATH就是此系统的标准，就等于没有配置X-Forwarded-For无法获取真实ip，没有配置IPG-PATH就无法得到正确的请求路径。

增加header不是好方法，因为nginx不一定是此系统的后端人员来编写，这样来制定约束不太好，但承认是目前想到的最好解决方法。可能还会有其他的方法，暂时未发现。

附一个同样的问题回答也被采纳的方法，但我未能正确实现，而且只是对于带后缀的静态资源，对于控制器请求应该还是解决不了。

<http://www.imooc.com/wenda/detail/324286>

## 4、重定向和转发

以上三大点的基本问题在于后端php生成的base\_url带有部署时的路径，那么在nginx知道的前提下，对带有项目路径的再重定向一次，主要是通过return、proxy\_pass、rewrite方法实现。

### 4.1、return方法

set $query "";

if ($query\_string ~\* ^.+$) {

set $query "?${query\_string}"; #保留url后面的?xx=xx参数

}

location ~ ^/path/project/(.\*) {

return 307 /IPG/$1$query; #307保留原来的请求方式

}

location /IPG/ {

proxy\_pass http://proxyweb/path/project/;

}

例如请求主页的时候，

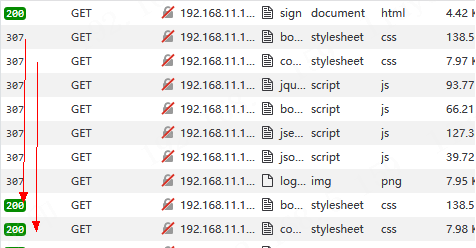
http://192.168.11.129/IPG/

返回的页面的base\_url为

http://192.168.11.129/path/project/

然后浏览器基于base\_url，拼接请求地址，就会继续向nginx发起请求，nginx匹配到path/project的时候，通过307方式重定向到/IPG/$1$query。这样就能在不修改项目代码的前提下使程序正常运行。

产生的问题：



每一个请求都会变为请求两次。

### 4.2、proxy\_pass方法

set $query "";

if ($query\_string ~\* ^.+$) {

set $query "?${query\_string}"; #保留url后面的?xx=xx参数

}

location ~ ^/path/project/(.\*) {

proxy\_pass http://192.168.11.129:80/IPG/$1$query;

}

location /IPG/ {

proxy\_pass http://proxyweb/path/project/;

}

这个方法就只会请求一次，但使用proxy\_pass无法改变地址栏的url，即

192.168.11.129/IPG/ 访问项目

然后点击登录并且成功之后，后台PHP CI框架会执行header("Location:xxxUrl");浏览器地址栏就会变为

192.168.11.129/path/project/console/main

但正常应该为

192.168.11.129/IPG/console/main

### 4.3、rewrite方法

rewrite 有4种参数方式

#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

#redirect #返回302临时重定向，浏览器地址会显示跳转后的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

set $query "";

if ($query\_string ~\* ^.+$) {

set $query "?${query\_string}"; #保留url后面的?xx=xx参数

}

rewrite ^/path/project/(.\*) /IPG/$1 permanent;

location /IPG/ {

proxy\_pass http://proxyweb/path/project/;

}

如果使用前两个，都能正常进行post请求，但浏览器地址栏仍是CI框架设置的地址（/path/project/xxx）。

如果使用后两个，地址栏是会变为想要的/IPG/xxx，但实现的重定向会把post请求变成get请求，导致执行post方法会有问题，get方法正常。



**总结：**以上4.1能解决问题，4.2和4.3未能解决问题，网上有proxy\_pass与rewrite混合使用的情况，我试过实现出来的效果与4.1是相同的，即能改变浏览器地址栏的就必须使用到重定向就必然会是两次请求。

## 5、duanqiang提供的方法

duanqiang提供的方法是使用ngx\_http\_substitutions\_filter\_module模块的subs\_filter方法，这个功能是对返回来的文档进行字符串的替换。

location /IPG/ {

proxy\_pass http://proxyweb/path/project/;

subs\_filter path/project IPG;

}

如请求主页的时候返回的html中base标签本应是：

http://192.168.11.129/path/project/

然后通过替换之后，页面显示的是

http://192.168.11.129/IPG/

这样改变，我们发起请求会是往/IPG/拼接，没有问题，但后台始终是/path/project/，会导致redirect的时候地址栏改变为ip/path/project/xxx，也就是现在的效果与4.2是相同的。