**[详解 nginx location ~ .\*\.(js|css)?$ 什么意思？](http://www.cnblogs.com/feiyuanxing/p/4668818.html)**

**http://www.cnblogs.com/feiyuanxing/p/4668818.html**

**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（参考资料而来，还未实际验证，试试就知道了，不必拘泥，仅供参考）：

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：

**1**

**location = / {精确匹配，必须是127.0.0.1/**

**#规则A**

**}**

**location = /login {精确匹配，必须是127.0.0.1/login**

**#规则B**

**}**

**location ^~ /static/ {非精确匹配，并且不区分大小写，比如127.0.0.1/static/js.**

**#规则C**

**}**

**location ~ \.(gif|jpg|png|js|css)$ {区分大小写，以gif,jpg,js结尾**

**#规则D**

**}**

**location ~\* \.png$ {不区分大小写，匹配.png结尾的**

**#规则E**

**}**

**location !~ \.xhtml$ {区分大小写，匹配不已.xhtml结尾的**

**#规则F**

**}**

**location !~\* \.xhtml$ {**

**#规则G**

**}**

**location / {什么都可以**

**#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：**

#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/**

**}**

#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/

}

**未试验过的其他信息：**

三、ReWrite语法   
last – 基本上都用这个Flag。   
break – 中止Rewirte，不在继续匹配   
redirect – 返回临时重定向的HTTP状态302   
permanent – 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