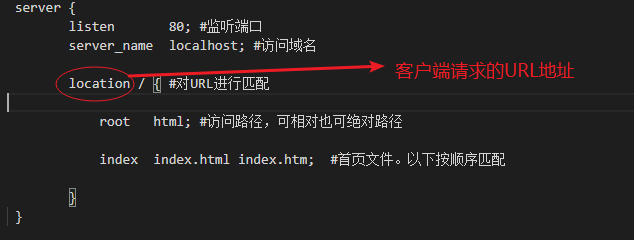
**指令-热启动**Nginx重新读取配置的命令 : nginx -s reload

## location语法详解 (#对请求URL地址进行匹配)



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

= 开头表示精确匹配

^~ 开头表uri以某个常规字符串开头，理解为匹配 url路径。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）。以xx开头

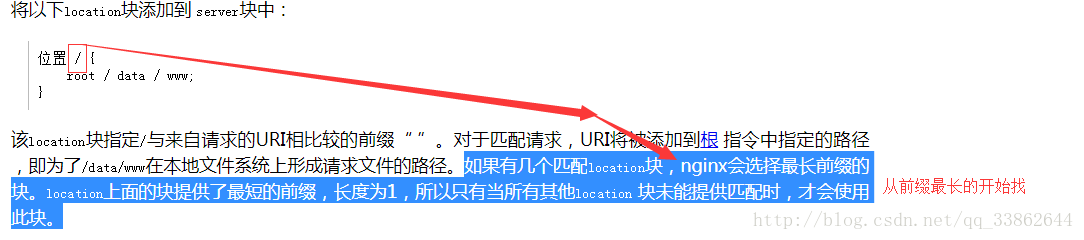
~ 开头表示区分大小写的正则匹配                     以xx结尾

~\* 开头表示不区分大小写的正则匹配                以xx结尾

!~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，仅供参考）：



首先精确匹配 =-》其次以xx开头匹配^~-》然后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。

### 实例:有如下匹配规则：

location = / {

   #规则A    #访问根目录/， 比如http://localhost/

}

location = /login {

   #规则B #例如访问 http://localhost/login

}

location ^~ /static/ {

   #规则C ##例如访问 http://localhost/static/a.html

}

location ~ \.(gif|jpg|png|js|css)$ {

   #规则D，注意：是根据括号内的大小写进行匹配。括号内全是小写，只匹配小写

}

location ~\* \.png$ {

   #规则E

}

location !~ \.xhtml$ {

   #规则F

}

location !~\* \.xhtml$ {

   #规则G

}

location / {

   #规则H #例如访问http://localhost/register

}

那么产生的效果如下：

1) 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

2) 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。

3) 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!）。规则F，规则G属于排除法，符合匹配规则也不会匹配到，所以想想看实际应用中哪里会用到。

4) 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/ {                              //以xx开头

    root /webroot/static/;

}

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

    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

location / {

    proxy\_pass http://tomcat:8080/

}