今天深入研究了下nginx的[location](http://nginx.org/en/docs/http/ngx_http_core_module.html#location)的用法，已经一些需要注意的细节，现在做一个归纳总结，以备后面查询。

**location匹配的是nginx的哪个变量？**

$request\_uri

**location的匹配种类有哪些?**

格式 location [ 空格 | = | ~ | ~\* | !~ | !~\* ] /uri/ {}

# 精确匹配: 相等(=)

# 字符串匹配: 字符串匹配(空格) 匹配开头(^~)

# 正则匹配: 区分大小写匹配(~) 不区分大小写匹配(~\*) 区分大小写不匹配(!~) 不区分大小写不匹配(!~\*)

**location搜索优先级优先级如何?**

精确匹配 > 字符串匹配( 长 > 短 [ 注: ^~ 匹配则停止匹配 ]) > 正则匹配( 上 > 下 )

# 精确匹配只能命中一个

# 字符串匹配使用匹配最长的最为匹配结果

# 正则匹配按照location定义的顺序进行匹配，先定义具有高优先级

*特别注意：* **字符串匹配优先搜索，但是只是记录下最长的匹配 ( *如果 ^~ 是最长的匹配，则会直接命中，停止搜索正则* )，然后继续搜索正则匹配，如果有正则匹配，则命中正则匹配，如果没有正则匹配，则命中最长的字符串匹配.**

**举例说明**

( 这里使用了 [echo-nginx-module](https://github.com/openresty/echo-nginx-module) 模块，方便做输出测试 )

精确匹配

location = /images/test.png {

echo 'config1';

}

location /images/test.png {

echo 'config2';

}

location \/images\/test\.png$ {

echo 'config3';

}

如果此时请求 <http://127.0.0.1/images/test>.png 会输出什么呢？

输出 config1, 毋容置疑，精确匹配优先级最高！

精确匹配的特殊情况

location = / {

index index.html;

}

location / {

echo 'config2';

}

此时是输入http://127.0.0.1 会输出什么呢？

是输出 config2, 怎么精确匹配的优先级不灵了呢？

是这样的，精确匹配还是起作用了，请求目录(非具体文件)，nginx会将请求内部定向到index文件，

既此时真正的请求是http://127.0.0.1/index.html, 这是 config2则被命中!

所以精确匹配不要用来匹配 /

字符串搜索与正则搜索

location /images/test.png {

echo 'config1';

}

location ^~ /images/ {

echo 'config2';

}

location ~ \/images\/test\.png$ {

echo 'config3';

}

location ~ \/images\/ {

echo 'config4';

}

如果此时请求 <http://127.0.0.1/images/test>.png 会输出什么呢？

当然是 config3，正则命中

(虽然 config1 为最长匹配的字符串，此时只做记录，后面还要搜索正则匹配，则config3正则匹配命中),

仔细观察可以发现config4也被匹配成功了，但是正则的匹配顺序是按照location的定义顺序匹配的，所以config3命中.

字符串匹配优先级的提升( ^~ ) location /images/ {     echo 'config1'; } location ^~ /images/test.png {     echo 'config2'; } location ~ /images/test\.png$ {     echo 'config3'; } location ~ \/images\/ {     echo 'config4'; } 如果此时请求 <http://127.0.0.1/images/test>.png 会输出什么呢？ 当然是config2, 首部匹配命中 (因为字符串匹配是优先搜索的，此时发现config2 为最长的字符串匹配且为^~匹配方式，所以停止搜索正则，直接命中！)