# Nginx

# Nginx如何处理一个请求

## 基于名字的虚拟主机

Nginx首先选定由哪一个*虚拟主机*来处理请求。让我们从一个简单的配置（其中全部3个虚拟主机都在端口\*：80上监听）开始：

server {

listen 80;

server\_name example.org www.example.org;

...

}

server {

listen 80;

server\_name example.net www.example.net;

...

}

server {

listen 80;

server\_name example.com www.example.com;

...

}

在这个配置中，nginx仅仅检查请求的“Host”头以决定该请求应由哪个虚拟主机来处理。如果Host头没有匹配任意一个虚拟主机，或者请求中根本没 有包含Host头，那nginx会将请求分发到定义在此端口上的默认虚拟主机。在以上配置中，第一个被列出的虚拟主机即nginx的默认虚拟主机——这是 nginx的默认行为。而且，可以显式地设置某个主机为默认虚拟主机，即在"listen"指令中设置"default\_server"参数：

server {

listen 80 **default\_server**;

server\_name example.net www.example.net;

...

}

请注意"default\_server"是监听端口的属性，而不是主机名的属性。后面会对此有更多介绍。

## 如何防止处理未定义主机名的请求

如果不允许请求中缺少“Host”头，可以定义如下主机，丢弃这些请求：

server {

listen 80;

server\_name "";

return 444;

}

在这里，我们设置主机名为空字符串以匹配未定义“Host”头的请求，而且返回了一个nginx特有的，非http标准的返回码444，它可以用来关闭连接。

从0.8.48版本开始，这已成为主机名的默认设置，所以可以省略server\_name ""。而之前的版本使用机器的*hostname*作为主机名的默认值。

## 基于域名和IP混合的虚拟主机

下面让我们来看一个复杂点的配置，在这个配置里，有几个虚拟主机在不同的地址上监听：

server {

listen 192.168.1.1:80;

server\_name example.org www.example.org;

...

}

server {

listen 192.168.1.1:80;

server\_name example.net www.example.net;

...

}

server {

listen 192.168.1.2:80;

server\_name example.com www.example.com;

...

}

这个配置中，nginx首先测试请求的IP地址和端口是否匹配某个[server](http://nginx.org/cn/docs/http/ngx_http_core_module.html#server)配置块中的[listen](http://nginx.org/cn/docs/http/ngx_http_core_module.html#listen)指令配置。接着nginx继续测试请求的Host头是否匹配这个[server](http://nginx.org/cn/docs/http/ngx_http_core_module.html#server)块中的某个[server\_name](http://nginx.org/cn/docs/http/ngx_http_core_module.html#server_name)的值。如果主机名没有找到，nginx将把这个请求交给默认虚拟主机处理。例如，一个从192.168.1.1:80端口收到的访问www.example.com的请求将被监听192.168.1.1:80端口的默认虚拟主机处理，本例中就是第一个服务器，因为这个端口上没有定义名为www.example.com的虚拟主机。

默认服务器是监听端口的属性，所以不同的监听端口可以设置不同的默认服务器：

server {

listen 192.168.1.1:80;

server\_name example.org www.example.org;

...

}

server {

listen 192.168.1.1:80 **default\_server**;

server\_name example.net www.example.net;

...

}

server {

listen 192.168.1.2:80 **default\_server**;

server\_name example.com www.example.com;

...

}

## 一个简单PHP站点配置

现在我们来看在一个典型的，简单的PHP站点中，nginx怎样为一个请求选择*location*来处理：

server {

listen 80;

server\_name example.org www.example.org;

root /data/www;

location / {

index index.html index.php;

}

location ~\* \.(gif|jpg|png)$ {

expires 30d;

}

location ~ \.php$ {

fastcgi\_pass localhost:9000;

fastcgi\_param SCRIPT\_FILENAME

$document\_root$fastcgi\_script\_name;

include fastcgi\_params;

}

}

首先，nginx使用前缀匹配找出最准确的location，这一步nginx会忽略location在配置文件出现的顺序。上面的配置中，唯一的前缀匹配location是"/"， 而且因为它可以匹配任意的请求，所以被作为最后一个选择。接着，nginx继续按照配置中的顺序依次匹配正则表达式的location，匹配到第一个正则 表达式后停止搜索。匹配到的location将被使用。如果没有匹配到正则表达式的location，则使用刚刚找到的最准确的前缀匹配的 location。

请注意所有location匹配测试只使用请求的URI部分，而不使用参数部分。这是因为写参数的方法很多，比如：

/index.php?user=john&page=1

/index.php?page=1&user=john

除此以外，任何人在请求串中都可以随意添加字符串：

/index.php?page=1&something+else&user=john

现在让我们来看使用上面的配置，请求是怎样被处理的：

* 请求"/logo.gif"首先匹配上location "/"，然后匹配上正则表达式"\.(gif|jpg|png)$"。因此，它将被后者处理。根据"root /data/www"指令，nginx将请求映射到文件/data/www/logo.gif"，并发送这个文件到客户端。
* 请求"/index.php"首先也匹配上location "/"，然后匹配上正则表达式"\.(php)$"。 因此，它将被后者处理，进而被发送到监听在localhost:9000的FastCGI服务器。[fastcgi\_param](http://nginx.org/cn/docs/http/ngx_http_fastcgi_module.html#fastcgi_param)指令将FastCGI的参数SCRIPT\_FILENAME的值设置为"/data/www/index.php"，接着FastCGI服务器执行这个文件。变量$document\_root等于[root](http://nginx.org/cn/docs/http/ngx_http_core_module.html#root)指令设置的值，变量$fastcgi\_script\_name的值是请求的uri，"/index.php"。
* 请求"/about.html"仅能匹配上location "/"，因此，它将使用此location进行处理。根据"root /data/www"指令，nginx将请求映射到文件"/data/www/about.html"，并发送这个文件到客户端。
* 请求"/"的处理更为复杂。它仅能匹配上location "/"，因此，它将使用此location进行处理。然后，[index](http://nginx.org/cn/docs/http/ngx_http_index_module.html#index)指令使用它的参数和"root /data/www"指令所组成的文件路径来检测对应的文件是否存在。如果文件/data/www/index.html不存在，而/data/www/index.php存在，此指令将执行一次内部重定向到"/index.php"，接着nginx将重新寻找匹配"/index.php"的location，就好像这次请求是从客户端发过来一样。正如我们之前看到的那样，这个重定向的请求最终交给FastCGI服务器来处理。

# nginx Windows版使用说明

nginx的Windows版本使用原生Win32 API（非Cygwin模拟层）。当前nginx/Windows只使用*select*作为通知方法，所以不要期待它有很高的性能和扩展性。鉴于这点和一些已知问题，nginx/Windows目前还处于*beta*阶段。nginx/Windows和Unix版本相比，功能几乎已经齐全，除了XSLT过滤器、图像过滤器、GeoIP模块和嵌入Perl语言支持以外。

安装nginx/Windows，需要[下载](http://nginx.org/cn/download.html)最新的1.5.13开发版本，因为开发分支上包含了所有已知的问题修复，尤其是针对Windows版本的问题修复。解压缩下载得到的zip文件，进入nginx-1.5.13目录，运行nginx。下面给出一个在C盘根目录下安装的例子：

cd c:\

unzip nginx-1.5.13.zip

cd nginx-1.5.13

start nginx

可以在命令行运行tasklist命令来查看nginx进程：

C:\nginx-1.5.13>tasklist /fi "imagename eq nginx.exe"

Image Name PID Session Name Session# Mem Usage

=============== ======== ============== ========== ============

nginx.exe 652 Console 0 2 780 K

nginx.exe 1332 Console 0 3 112 K

其中一个是主进程，另一个是工作进程。如果nginx没有启动，请查看logs\error.log文件以寻找失败原因。如果日志文件不存在，那失败原因会记录在Windows事件日志中。如果某次请求没有展示预想的页面，而是展示了错误页面，也请查看logs\error.log文件。

nginx/Windows使用工作目录作为前缀将配置文件中设置的相对目录补齐。就上面安装的例子而言，工作目录应该是C:\nginx-1.5.13\（工作目录基本上与运行文件所在的目录相同）。配置文件中的目录请使用“/”，而不是“\”做目录分隔：

access\_log logs/site.log;

root C:/web/html;

nginx/Windows作为标准控制台应用运行，而不是系统服务。可以用下面的命令控制：

|  |  |
| --- | --- |
| nginx -s stop | 快速退出 |
| nginx -s quit | 优雅退出 |
| nginx -s reload | 更换配置，启动新的工作进程，优雅的关闭以往的工作进程 |
| nginx -s reopen | 重新打开日志文件 |

## 已知问题

* 虽然可以启动若干工作进程运行，实际上只有一个进程在处理请求所有请求。
* 一个工作进程只能处理不超过1024个并发连接。
* 缓存和其他需要共享内存支持的模块在Windows Vista及后续版本的操作系统中无法工作，因为在这些操作系统中，地址空间的布局是随机的。

## 日后可能加强的功能

* 作为系统服务运行。
* 使用“I/O完成端口”作为事件模型。
* 使用单工作进程多线程的模型。

# 配置文件中的计量单位

容量可以用千字节(k，K)和兆字节(m，M)来描述，比如“8k”，“1m”。 如果没有指定单位，容量以字节为单位。

时间可以用分钟、小时、天等来描述：

|  |  |
| --- | --- |
| s | 秒 |
| m | 分钟 |
| h | 小时 |
| d | 天 |
| w | 周 |
| M | 月，30天 |
| y | 年，365天 |

比如，“1h 30m”，“1y 6M”。 另外，在一些指令中，描述时间可以精确到毫秒精度(ms)。 如果没有指定单位，时间以秒为单位。

# 调试日志

要开启调试日志，首先需要在配置nginx时打开调试功能，然后编译：

./configure --with-debug ...

然后在配置文件中设置error\_log的级别为debug：

error\_log /path/to/log debug;

nginx的windows二进制版本总是将调试日志开启的，因此只需要设置debug的日志级别即可。

注意，重新定义错误日志时，如过没有指定debug级别，调试日志会被屏蔽。下面的例子里，在[server](http://nginx.org/cn/docs/http/ngx_http_core_module.html#server)层中重新定义的日志就屏蔽了这个虚拟主机的调试日志：

error\_log /path/to/log debug;

http {

server {

error\_log /path/to/log;

...

为了避免这个问题，注释这行重新定义日志的配置，或者也给日志指定debug级别：

error\_log /path/to/log debug;

http {

server {

error\_log /path/to/log debug;

...

另外，也可以只针对[选定的客户端地址](http://nginx.org/cn/docs/ngx_core_module.html#debug_connection)开启调试日志：

error\_log /path/to/log;

events {

debug\_connection 192.168.1.1;

debug\_connection 192.168.10.0/24;

}

# 虚拟主机名

虚拟主机名使用[server\_name](http://nginx.org/cn/docs/http/ngx_http_core_module.html#server_name)指令定义，用于决定由某台[虚拟主机](http://nginx.org/cn/docs/http/ngx_http_core_module.html#server)来处理请求。具体请参考《[nginx如何处理一个请求](http://nginx.org/cn/docs/http/request_processing.html)》。虚拟主机名可以使用确切的名字，通配符，或者是正则表达式来定义：

server {

listen 80;

server\_name example.org www.example.org;

...

}

server {

listen 80;

server\_name \*.example.org;

...

}

server {

listen 80;

server\_name mail.\*;

...

}

server {

listen 80;

server\_name ~^(?<user>.+)\.example\.net$;

...

}

nginx以名字查找虚拟主机时，如果名字可以匹配多于一个主机名定义，比如同时匹配了通配符的名字和正则表达式的名字，那么nginx按照下面的优先级别进行查找，并选中第一个匹配的虚拟主机：

 确切的名字；

最长的以星号起始的通配符名字：\*.example.org；

最长的以星号结束的通配符名字：mail.\*；

第一个匹配的正则表达式名字（按在配置文件中出现的顺序）。

# 下载:

<http://nginx.org/en/download.html>

源代码:

http://hg.nginx.org/nginx/file/fd722b890eab/src