

Nginx基础03：配置文件nginx.conf （Part2）

文章分类: Server; 标签: Nginx; 作者: Hackyle;

更新时间: Mon Jan 16 14:53:54 CST 2023

1. http-server块

1. 公有配置
1. 防盗链
2. 地址重写
3. 流程控制
2. listen: 监听端口
1. 参数
2. 实例
3. server_name: 指定域名
4. location: URL映射
1. URI映射
2. 路径替换
3. 反向代理
4. SSL
5. 访问控制
2. 内置变量

上一篇文章概述与罗列了"全局配置块、events配置块、http全局块"的基本配置与属性，本篇文章将继续深入server块的配置项，以及相关应用。

上篇文章地址: [Nginx基础02：配置文件nginx.conf \(Part1\)](#)

如何使用本篇文章

- 本文作为一篇高度总结和罗列nginx.conf中所有的基础配置项，循规蹈矩地按照文章的**顺序阅读的方式不可取**
- 笔者建议所有读者，**先看目录**，掌握Nginx都有哪些基础的配置块，再想要了解那**一个配置块**时，**再详细去看**
- 作为一篇字典类的文章，建议读者**善用浏览器的全文查找功能**，按Ctrl + F调出查找功能，搜索你感兴趣的关键字，针对性地学习

内容导航

- [http-server块](#)
 - [公有配置](#)
 - [防盗链](#)
 - [地址重写](#)
 - [流程控制](#)
 - [listen: 监听端口](#)
 - [参数](#)
 - [实例](#)
 - [server_name: 指定域名](#)
 - [location: URL映射](#)
 - [URI映射](#)
 - [路径替换](#)
 - [反向代理](#)
 - [SSL](#)
 - [访问控制](#)
- [内置变量](#)

```
1 user nginx;
2 worker_processes auto;
3 error_log /var/log/nginx/error.log notice;
4 pid /var/run/nginx.pid;
5
6
7
8
9 events {
10     worker_connections 1024;
11 }
12
13
14 http {
15     include /etc/nginx/mime.types;
16     default_type application/octet-stream;
17     log_format main '$remote_addr - $remote_user [$time_local] "$request"
18         '$status $body_bytes_sent "$http_referer"
19         "$http_user_agent" "$http_x_forwarded_for"';
20     access_log /var/log/nginx/access.log main;
21     sendfile on;
22     keepalive_timeout 65;
23
24     server {
25         listen 80;
26         server_name localhost;
27
28         location / {
29             root /usr/share/nginx/html;
30             index index.html index.htm;
31         }
32
33         error_page 500 502 503 504 /50x.html;
34         location = /50x.html {
35             root /usr/share/nginx/html;
36         }
37     }
38
39     server {
40         listen 8080;
41         server_name localhost;
42
43         location / {
44             root /usr/share/nginx/html;
45             index index.html index.htm;
46         }
47
48         error_page 500 502 503 504 /50x.html;
49         location = /50x.html {
50             root /usr/share/nginx/html;
51         }
52     }
53 }
```

nginx.conf配置

- 全局块配置
- events {
events块配置
- http块
 - http全局块
 - server块1
 - server全局块
 - location块1
 - location块2
 -
 - server块2

HACKYLE

正式开始前，还是再次强调server块在配置文件中的位置

http-server块

server块和“虚拟主机”的概念有密切联系

- 虚拟主机技术主要应用于HTTP、FTP及EMAIL等多项服务，将一台服务器的某项或者全部服务内容逻辑划分为多个服务单位，对外表现为多个服务器，从而充分利用服务器硬件资源。从用户角度来看，一台虚拟主机和一台独立的硬件主机是完全一样的。
- 在使用Nginx服务器提供Web服务时，利用虚拟主机的技术就可以避免为每一个要运行的网站提供单独的Nginx服务器，也无需为每个网站对应运行一组Nginx进程。虚拟主机技术使得Nginx服务器可以在同一台服务器上只运行一组Nginx进程，就可以运行多个网站。
- 一个http块都可以包含多个server块，而**每个server块就相当于一台虚拟主机**，它内部可有多台主机联合提供服务，一起对外提供在逻辑上关系密切的一组服务（或网站）。

公有配置

error_page指令

- 配置Nginx出现错误时，返回自定义页面以及错误代码，或将浏览器重定向到其他URI。
- 出现404错误时，响应根目录下的html文件：error_page 404 /404.html;

防盗链

valid_referers none | blocked | server_names | string...

- 功能：控制是否需要检验referer，设定校验referer时的校验值
- 应用场景：防盗链
- 参数
 - none：如果Header中的Referer为空，允许访问
 - blocked：在Header中的Referer不为空，但是该值被防火墙或代理进行伪装过，如不带"http://"、"https://"等协议头的资源允许访问。
 - server_names：指定具体的域名或者IP。也就是说Request中的Referer必须为这里指定的参数，才让访问
 - string：可以支持正则表达式和*的字符串。如果是正则表达式，需要以`~`开头表示
- 位置：server、location
- **防盗链实现原理**：将Request中的Referer与valid_referers设定的值进行比对，如果匹配到了就将\$invalid_referer变量置0，如果没有匹配到，则将\$invalid_referer变量置为1，**通过if语句判定**不符合条件的响应403

```

1  server {
2      listen 443 ssl http2;
3      listen [::]:443 ssl http2;
4      server_name res.hackyle.com;
5
6      ssl_certificate "/etc/nginx/cert/res.hackyle.com.pem";
7      ssl_certificate_key "/etc/nginx/cert/res.hackyle.com.key";
8      ssl_session_cache shared:SSL:1m;
9      ssl_session_timeout 10m;
10     ssl_ciphers HIGH:!aNULL:!MD5;
11     ssl_prefer_server_ciphers on;
12
13     #防盗链：验证所有请求中的Referer是否来自*.hackyle.com，否则响应403
14     valid_referers blocked *.hackyle.com;
15     if ($invalid_referer){
16         return 403;
17     }
18
19     #客户端：增删改查的接口
20     #对外提供静态资源的地址：https://res.hackyle.com/桶名/年份/月份/uuid.文件拓展名
21     location / {
22         proxy_pass http://localhost:9000/;
23     }
24 }
```

地址重写

rewrite regex replacement [flag];

- **地址重写 (Rewrite) :** 等价于请求重定向; **地址转发 (Forward) :** 等价于请求转发
- 功能: 将请求中的regex替换为replacement
- 参数
 - regex: 匹配URI中将要被替换的内容
 - replacement: 替换成谁。如果该字符串是以"http://"或者"https://"开头的, 则不会继续向下对URI进行其他处理, 而是直接返回重写后的URI给客户端。
 - flag: 用来设置rewrite对URI的处理行为
 - last:
 - break
 - redirect
 - permanent
 - 位置: server、location、if

rewrite_log on | off (默认值) ;

- 功能: 开启后, URL重写的相关日志将以notice级别输出到error_log指令配置的日志文件汇总。
- 位置: http、server、location、if

流程控制**set \$variable value;**

- variable: 变量的名称, 用"\$"作为变量的第一个字符, 且不能与Nginx服务器预设的全局变量同名。
- value: 变量的值, 可以是字符串、其他变量或者变量的组合等。
- 位置: server、location、if

if (condition){...}

- 语法要求: if与左括号之间存在一个空格
- 位置: server、location
- condition的写法
 1. 变量名: 对应的值为空或者是0, if都判断为false, 其他条件为true。
 2. "="和"!=": 满足条件为true, 不满足为false。例如: if (\$request_method = POST){ return 405; }
 3. 正则表达式: 匹配成功返回true, 否则返回false。变量与正则表达式之间使用"~","~*","!~","!~*"来连接。
 - "~"代表匹配正则表达式过程中区分大小写,
 - "~*"代表匹配正则表达式过程中不区分大小写
 - "!~"和"!~*"刚好和上面取相反值, 如果匹配上返回false,匹配不上返回true
 - 例子:


```
if ($http_user_agent ~ MSIE){

    # $http_user_agent的值中是否包含MSIE字符串, 如果包含返回true

}
```
 - 注意: 正则表达式字符串一般不需要加引号, 但是如果字符串中包含"}"或者是";"等字符时, 就需要把引号加上。
- 4. 文件是否存在:


```
if (-f $request_filename){

    #判断请求的文件是否存在。文件存在时返回true

}

if (!-f $request_filename){

    #判断请求的文件是否不存在。文件不存在且文件目录存在时返回true, 其他情况返回false

}
```
- 5. 判断请求的目录是否存在使用"-d"和"!-d"
 1. 当使用"-d"时, 如果请求的目录存在, if返回true, 如果目录不存在则返回false
 2. 当使用"!-d"时, 如果请求的目录不存在但上级目录存在则返回true, 该目录和它上级目录都不存在则返回false,如果请求目录存在也返回
- 6. 判断请求的目录或者文件是否存在使用"-e"和"!-e"
 1. 当使用"-e",如果请求的目录或者文件存在时, if返回true,否则返回
 2. 当使用"!-e",如果请求的文件和文件所在路径上的目录都不存在返回true,否则返回false

7. 判断请求的文件是否可执行使用"-x"和"!-x"

1. 当使用"-x",如果请求的文件可执行, if返回true,否则返回false
2. 当使用"!-x",如果请求文件不可执行, 返回true,否则返回false

break;

中断当前作用域break后面的指令, 即位于它前面的指令配置生效, 位于后面的指令配置无效

位置: server、location、if

return code [text] | code URL | URL;

- 功能: 立即响应给客户端, 其后面的配置都将失效
- 参数
 - code: 状态代码
 - text: 响应体内容, 支持变量的使用
 - URL: 响应URL地址, 客户端收到后会重定向
- 位置: server、location、if

listen: 监听端口**Listen指令: 指定本个server所使用的端口 (监听端口)**

```

1  listen address[:port]
2  [default_server]
3  [ssl]
4  [http2 | spdy]
5  [proxy_protocol]
6  [setfib=number]
7  [fastopen=number]
8  [backlog=number]
9  [rcvbuf=size]
10 [sndbuf=size]
11 [accept_filter=filter]
12 [deferred]
13 [bind]
14 [ipv6only=on|off]
15 [reuseport]
16 [so_keepalive=on|off|[keepidle]:[keepintvl]:[keepcnt]];
17
18 listen port
19 [default_server]
20 [ssl]
21 [http2 | spdy]
22 [proxy_protocol]
23 [setfib=number]
24 [fastopen=number]
25 [backlog=number]
26 [rcvbuf=size]
27 [sndbuf=size]
28 [accept_filter=filter]
29 [deferred]
30 [bind]
31 [ipv6only=on|off]
32 [reuseport]
33 [so_keepalive=on|off|[keepidle]:[keepintvl]:[keepcnt]];
34
35 listen unix:path
36 [default_server]
37 [ssl]
38 [http2|spdy]
39 [proxy_protocol]

```

```

40 | [backlog=number]
41 | [rcvbuf=size]
42 | [sndbuf=size]
43 | [accept_filter=filter]
44 | [deferred]
45 | [bind]
46 | [so_keepalive=on|off|[keepidle]:[keepintvl]:[keepcnt]];

```

参数

address: 监听请求来的IP地址

- 如果是IPv6的地址，需要使用中括号 “[]” 括起来，比如[fe80::1]等。
- 也即本个server只针对指定请求IP的访问。

port: 端口号

- 如果只定义了IP地址没有定义端口号，就使用80端口。
- 要是没配置listen指令，且Nginx以超级用户权限运行，则使用:80，否则使用:8000。
- 多个虚拟主机可以同时监听同一个端口，但是server_name需要设置成不一样；

default_server: 默认server

- 如果没有找到address:port，则使用本个配置指定的地址。

backlog=number

- 设置监听函数listen()最多允许多少网络连接同时处于挂起状态
- 在FreeBSD中默认为-1，其他平台默认为511。

accept_filter=filter: 设置监听端口对请求的过滤，被过滤的内容不能被接收和处理。

- 本指令只在FreeBSD和NetBSD 5.0+平台下有效。
- filter可以设置为dataready或httpready，具体参阅Nginx的官方文档。

bind: 标识符

- 使用独立的bind()处理此address:port;
- 一般情况下，对于端口相同而IP地址不同的多个连接，Nginx服务器将只使用一个监听命令，并使用bind()处理端口相同的所有连接。

实例

- listen 127.0.0.1:8000; #只监听来自0.0.1这个IP，请求8000端口的请求
- listen localhost:8000; #和上面效果一样
- listen 127.0.0.1; #只监听来自0.0.1这个IP，请求80端口的请求（不指定端口，默认80）
- listen 8000; #监听来自所有IP请求8000端口的请求
- listen *:8000; #和上面效果一样

server_name: 指定域名

功能: 用于配置虚拟主机的名称

语法: server_name name...;

实例: server_name myserver.com www.myserver.com

name: 域名，多个用空格分割

- name可以使用通配符 “*”，注意的是通配符不能出现在域名的中间，只能出现在首段或尾段
 - *.baidu.com: 表示百度下的所有子域
 - *: 所有顶级域名下，名为“baidu”的域
 - 错误的配置: *.cn www.itheima.c*
- name可以使用正则: 例如: server_name ~^www\.(w+)\.com\$;

name匹配优先级:

1. 准确匹配server_name
2. 通配符在开始时匹配server_name成功
3. 通配符在结尾时匹配server_name成功
4. 正则表达式匹配server_name成功

location: URL映射

功能: 映射URL请求(支持正则)到具体的页面、处理器上

```

1 | 语法: location [ = | ~ | ~* | ^~ ] uri {
2 |     root 请求访问的根目录(使用绝对路径);
3 |     index 设置网站的首页;
4 | }
```

URI映射

- “=” : 进行普通字符精确匹配, 也就是完全匹配。如果已经匹配成功, 就停止继续向下搜索并立即处理此请求。
- “^~” : 前缀匹配。如果匹配成功, 则不再匹配其他location。
- “~” : 表示执行一个正则匹配, 区分大小写。
- “~*” : 表示执行一个正则匹配, 不区分大小写。

匹配的优先级:

- 等号类型(=)的优先级最高。一旦匹配成功, 则不再查找其他匹配项, 停止搜索。
- ^~类型表达式, 不属于正则表达式。一旦匹配成功, 则不再查找其他匹配项, 停止搜索。
- 正则表达式类型(~ ~*)的优先级次之。如果有多个location的正则能匹配的话, 则使用正则表达式最长的那个。
- 常规字符串匹配类型。按前缀匹配。
- / 通用匹配, 如果没有匹配到, 就匹配通用的

前置参数 (“=”) 示例

```

1 | server {
2 |     listen 80;
3 |     server_name 127.0.0.1;
4 |     location =/abc {
5 |         ...
6 |     }
7 | }
8 | 可以匹配到
9 | http://192.168.200.133/abc
10 | http://192.168.200.133/abc?p1=TOM
11 | 匹配不到
12 | http://192.168.200.133/abc/
13 | http://192.168.200.133/abcdef
```

前置参数 (“~”) 示例

```

1 | server {
2 |     listen 80;
3 |     server_name 127.0.0.1;
4 |     location ~^/abc\w${ #红色部分是正则, ^表示一行的开始, $表示一行的结束, /表示请求中的
5 |         default_type text/plain;
6 |         return 200 "access success";
7 |     }
8 | }
9 | }
10 | server {
11 |     listen 80;
12 |     server_name 127.0.0.1;
13 |     location ~*^/abc\w${
14 |         default_type text/plain;
```

```

15 |         return 200 "access success";
16 |     }
    | }

```

前置参数 (“^~”) 示例

```

1 | server {
2 |     listen 80;
3 |     server_name 127.0.0.1;
4 |     location ^~/abc{
5 |         default_type text/plain;
6 |         return 200 "access success";
7 |     }
8 | }

```

路径替换

例子:

```

1 | location /img/ {
2 |     alias /usr/local/image/; #将"/img/"替换为"/usr/local/image/"
3 |     #例如: 请求/img/aa.jpg, 在本机中将会被替换为/usr/local/image/aa.jpg
4 | }

```

指定根目录: root path;

例子:

```

1 | location /img/ {
2 |     root /usr/data/; #请求/img/aa.jpg, 将会拼接为/usr/data/img/aa.jpg
3 | }

```

反向代理

Reverse Proxy: 用Nginx来接收internet上的连接请求, 然后将请求转发给内部网络上的服务器去具体处理

proxy_pass URL;

- 设置被代理服务器地址, 可以是主机名称、IP地址加端口号形式。
- URL为要设置的被代理服务器地址, 包含传输协议('http','https://')、主机名称或IP地址加端口号、URI等要素。
- 位置: location

proxy_set_header field value;

- 更改Nginx服务器接收到的客户端请求的请求头信息, 然后将新的请求头发送给代理的服务器
- 默认值proxy_set_header Host \$proxy_host; proxy_set_header Connection close;
- 位置: http、server、location
- 例子:

```

server {
    listen 8080;
    server_name localhost;
    location /server {
        proxy_pass http://192.168.200.146:8080/;
        proxy_set_header username TOM;
    }
}

```

proxy_redirect redirect replacement; | proxy_redirect default; | proxy_redirect off;

- 重置头信息中的"Location"和"Refresh"的值

- 参数
 - redirect: 目标, Location的值; replacement: 要替换的值
 - default: 将location块的uri变量作为replacement, 将proxy_pass变量作为redirect进行替换
 - off: 关闭proxy_redirect
- 默认: proxy_redirect default;
- 位置: http、server、location

```
server {
    listen 8081;
    server_name localhost;
    location / {
        proxy_pass http://192.168.200.146:8081/;
        proxy_redirect http://192.168.200.146 http://192.168.200.133;
    }
}
```

proxy_buffering on (默认值) |off;

- 开启或者关闭代理服务器的缓冲区
- 位置: http、server、location

proxy_buffers number size;

- 指定单个连接从代理服务器读取响应的缓存区的个数和大小
- 默认值: proxy_buffers 8 4k | 8K;(与系统平台有关)
- number:缓冲区的个数
- size:每个缓冲区的大小, 缓冲区的总大小就是number*size
- 位置: http、server、location

proxy_buffer_size size;

- 设置从被代理服务器获取的第一部分响应数据的大小。保持与proxy_buffers中的size一致即可, 当然也可以更小。
- 默认值: proxy_buffer_size 4k | 8k;(与系统平台有关)
- 位置: http、server、location

proxy_busy_buffers_size size;

- 限制同时处于BUSY状态的缓冲总大小。
- 默认值proxy_busy_buffers_size 8k | 16K;
- 位置http、server、location

proxy_temp_path path;

- 当缓冲区存满后, 仍未被Nginx服务器完全接受, 响应数据就会被临时存放在磁盘文件上, 该指令设置文件路径
- 位置http、server、location

proxy_temp_file_write_size size;

- 用来设置磁盘上缓冲文件的大小
- 默认值: proxy_temp_file_write_size 8K|16K;
- 位置http、server、location

SSL

HTTPS是一种通过计算机网络进行安全通信的传输协议。它经由HTTP进行通信, 利用SSL/TLS建立全通信, 加密数据包, 确保数据的安全性。

SSL(Secure Sockets Layer)安全套接层

TLS(Transport Layer Security)传输层安全

ssl_certificate file;

- 为当前这个虚拟主机指定一个带有PEM格式证书的证书。
- 位置: http、server

ssl_certificate_key file;

- 指定PEM secret key文件的路径

- 位置: http、server

ssl_session_cache off | none | [builtin[:size]] [shared:name:size]

- 配置用于SSL会话的缓存
- 默认值: ssl_session_cache none;
- 参数
 - off:禁用会话缓存, 客户端不得重复使用会话
 - none:禁止使用会话缓存, 客户端可以重复使用, 但是并没有在缓存中存储会话参数
 - builtin:内置OpenSSL缓存, 仅在一个工作进程中使用。
 - shared:所有工作进程之间共享缓存, 缓存的相关信息用name和size来指定
- 位置: http、server

ssl_session_timeout time;

- 开启SSL会话功能后, 设置客户端能够反复使用储存在缓存中的会话参数时间。
- 默认值: ssl_session_timeout 5m;
- 位置: http、server

ssl_ciphers ciphers;

- 指出允许的密码, 密码指定为OpenSSL支持的格式
- 默认值: ssl_ciphers HIGH:!aNULL:!MD5;
- 位置: http、server

ssl_prefer_server_ciphers on | off (默认值) ;

- 指定是否服务器密码优先客户端密码
- 位置: http、server

访问控制

允许访问: allow [address | CIDR | all]

- 使用字段: http, server, location, limit_except

禁止访问: deny all/IP网段;

- 使用字段: http, server, location, limit_except

例子:

```
1 | location / {
2 |     deny 10.1.100.100; #不允许该IP访问
3 |     allow 10.1.200.0/24; #允许该网段访问
4 |     allow 192.168.1.0/16; #允许该网段访问
5 |     deny all; #除了allow允许的网段, 其他请求IP都禁止访问
6 | }
```

内置变量

nginx的配置文件中可以使用的内置变量以美元符\$开始, 也有人叫全局变量。其中, 部分预定义的变量的值是可以改变的。

\$args:

- 这个变量等于请求行中的参数, 同\$query_string。
- 例如: /aa/bb?name=kyle&age=22中的 "name=kyle&age=22"

\$content_length : 请求头中的Content-length字段。

\$content_type : 请求头中的Content-Type字段。

\$document_root : 当前请求在root指令中指定的值。

\$host : 请求主机头字段, 否则为服务器名称。

\$http_user_agent : 客户端agent信息

\$http_cookie：客户端cookie信息

\$limit_rate：这个变量可以限制连接速率。

\$request_method：客户端请求的动作，通常为GET或POST。

\$remote_addr：客户端的IP地址。

\$remote_port：客户端的端口。

\$remote_user：已经经过Auth Basic Module验证的用户名。

\$request_filename：当前请求的文件路径，由root或alias指令与URI请求生成。

\$scheme：HTTP方法（如http，https）。

\$server_protocol：请求使用的协议，通常是HTTP/1.0或HTTP/1.1。

\$server_addr：服务器地址，在完成一次系统调用后可以确定这个值。

\$server_name：服务器名称。

\$server_port：请求到达服务器的端口号。

\$request_uri：包含请求参数的原始URI，不包含主机名，如："/foo/bar.php?arg=baz"。

\$uri：不带请求参数的当前URI，\$uri不包含主机名，如"/foo/bar.html"。

\$document_uri：与\$uri相同

版权声明：非明确标注皆为原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上本文链接及此声明。
原文链接：<https://blog.hackyle.com/article/server/nginx-conf-part2>

留下你的评论

Name:

Email:

Link:

File Edit View Format Tools Table Help

↶ ↷ **B** *I* U ~~S~~                

Input comment, please

p

0 words 

SUBMIT

RESET

© Copy Right: 2022 HACKYLE. All Rights Reserved

Designed and Created by HACKYLE SHAW

