lognovel
榄果

nginx

Nginx日志参考手册

格式、配置和使用

http://lognovel.io

Nginx日志参考手册: 格式、配置和使用

http://lognovel.io
Mr. ChunXin Li

出版日期 October 10th,2021

修订历史		
修订 1.0	2021年10月10日.	<doc@lognovel.com>
加入HTTP/1.1状态码部分		
修订 0.1	2021年8月16日.	<doc@lognovel.com>
文档建立		

内容简介

在这份小手册里，简要介绍了ngnix日志的配置和使用方法。并用几个简单的例子用于说明nginx的日志系统是如何工作的。在附录部分将nginx的所有变量（字段）及http的状态列出，方便读者查阅。

目录

插图清单

表格清单

前言

1. 排版约定

在正文中会有一些诸如程序代码、系统命令或是屏幕输出一类的信息，为了能清晰地展示各种元素，本手册遵守下列排版约定。

1. **字体约定**，在正文中嵌入的一些系统命令、文件名、函数或是参数等使用有别于正文的字体：

   a. 等宽字体：程序片段、正文中出现的配置选项、变量、函数名等，我们采用等宽字体，样式如下：`serverip`。

   b. 等宽微粗字体：表示由用户输入的系统命令。例如，在linux下，我们查网络接口信息的命令：`ifconfig eth0`。

   c. 等宽斜体：表示由用户输入的值或是一些需要设定的参数的值，例如，用户在浏览器里输入的一个url，用如下的样式显示：*`http://lognovel.com`*。

   d. 等宽微粗斜体字体：表示文件名、数据库名及新的术语等。例如，文件的名称用如下的样式来显示：***`myfirst.xml`***。

2. **提示、告诫类信息**（admonitions），有一些内容是正文的补充或是对用户的一种提醒。为了能清晰地展示这部分内容，采用独立的“信息块”来呈现。一共有5种，分别为：

   - 

     提示

     可以使用`apt-get upgrade`来升级您的ubuntu或debian Linux系统。

   - 

     注意

     在运行`apt-get upgrade`之前，需要先运行用`apt-get update`命令。

   - 

     重要

     请及时升级系统补丁程序，这是保证系统处于安全状态的好方法。

   - 

     小心

     如果系统升级还没有完成，请不要重新启动您的电脑。

   - 

     警告

     在系统升级过程中千万不要关闭电源。

3. **屏幕内容围栏及程序代码围栏：**

   - 屏幕内容围栏，有时候我们需要整块显示在屏幕上输出的内容，或者是键入的命令。例如，在linux下，我们使用`ifconfig eth0`命令来查网络接口信息：

```
$ ifconfig eth0
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 98:fa:9b:db:b9:9f  txqueuelen 1000  (以太网)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xea300000-ea320000
```

- 程序代码围栏，对于一些程序代码，使用带有序号的样式来展示，例如，一段php代码：

```
 1 <?php
 2   class SimpleBook {
 3     private $title;
 4
 5     function __construct($title_in) {
 6       $this->title  = $title_in;
 7     }
 8
 9     function getTitle() {return $this->title;}
10   }
11 ?>
```

2. 关于我们

lognovel是一家提供数据分析平台及产品、提供数据分析服务的公司。在机器数据及各种系统日志方面具有丰富的经验。在长时间服务于客户及产品开发的过程中，接触到大量的开源系统，为了能让技术人员更好地开发产品及服务客户，我们将很多开源系统的使用及配置方法写成文档，在内部进行分享。

在这样的过程中，我们从开源社区汲取了丰富的营养，做为对社区的一点微不足道的贡献，我们将这些文档进行整理，并将之回馈给社区。可以从下方的链接来获取这些文档：

https://github.com/lognovel/

如果您对我们有任何建议请让我们知晓，您的任何建议都弥足珍贵，您可以通过如下地址和方式联系我们：

上海榄果数字科技有限公司
莘砖公路668号G60科创大厦A座804
上海，松江
021-67695150
<doc@lognovel.com>
http://lognovel.io

第 1 章 nginx

1. nginx简介

Nginx[engine-x]是一个高性能的HTTP、web代理、邮件代理、通用的TCP/UDP代理服务器软件。最初由俄罗斯人Igor Sysoev开发。很长一段时间以来，很多俄罗斯的站点都在使用它。现在nginx已经在全球范围内得到了广泛的应用。在2019年，nginx的商业化运营被知名的F5收购。从nginx的商业站点我们可以看到它一个简短的历史：https://www.nginx.com/careers/#history-nginx

1. **2004年**：俄罗斯人Igor Sysoev为了解决C10K问题[1]创建了nginx。nginx比较完美的解决了这个问题。得到很多站点的认可。

2. **2011年**：Igor Sysoev联合创办了NGINX公司，支持nginx及nginx-Plus的开发。并为用户提供企业级和专业的服务。从C10问题起步，nginx用了6年的时间，跃升为世界级的服务器软件产品。

3. **2013年**：nginx-plus第一个版本正式发布。这个版本可以为用户的web应用提供高性能的web服务、强大的负载均衡服务、高伸缩性的加速服务。

4. **2016年**：NGINX Inc.的员工超过100人。用他们自己的话说就是：“作为web的心脏，NGINX的很多雇员在应用开发和部署领域都是很有前瞻性头脑的人。”

5. **2017年**：NGINX收获了1000个客户。

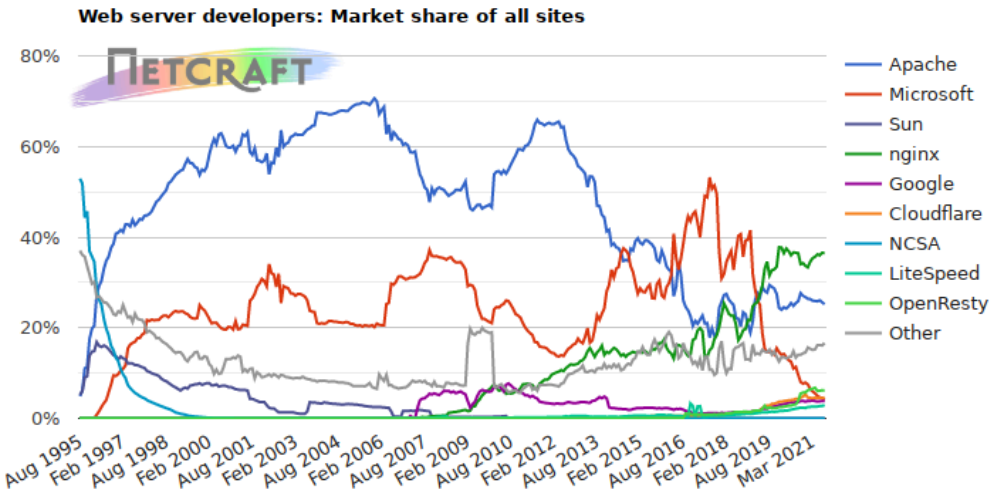
6. **2018年**：超过450M个web站点在使用nginx。

7. **2019年**：F5收购NGINX Inc.。

根据netcraft的调查，截至到2021年8月，nginx在整个Internet的使用量已经占到了22.38%的份额。而且还在不断上升。

[1]C10K是关于如何解决10000个并发连接的问题，可以参考http://www.kegel.com/c10k.html

图 1.1. web服务器



可以参考如下的链接：

https://news.netcraft.com/archives/2021/08/25/august-2021-web-server-survey.html

2. nginx日志简介

nginx日志主要分为两种：access_log(访问日志)和error_log(错误日志)。

1. access_log

   访问日志(access_log)主要记录客户端的请求。通过access_log指令来实现。客户端向Nginx服务器发起的每一次请求都记录在这里。客户端IP、浏览器信息、referer、请求处理时间及请求URL等都可以在访问日志中得到。具体要记录哪些信息及记录的格式、排列顺序等，可以通过log_format指令进行定制。

2. error_log

   错误日志(error_log)记录了访问及系统的错误信息，可以帮助我们定位错误的原因。错误日志在Nginx中是通过error_log指令实现的。该指令记录服务器和请求处理过程中的错误信息。与access_log不同的是，error_log的日志不能使用log_format指令进行定制。

另外，在access_log和log_format中使用了很多变量，详细的变量信息可以参考Nginx官方文档。nginx变量参考站点 [https://nginx.org/en/docs/varindex.html]

第 2 章 配置

1. 将日志记录到syslog

Nginx使用error_log和access_log两个指令来将日志记录到日志文件中。在配置日志的过程中，会用到下面的几个参数：

1. server=address ：定义syslog服务器的地址，这个地址可以是一个域名或是IP地址，可以带一个端口号。如果端口没有指定，那就会使用UDP的514端口向syslog服务器发送日志信息。如果域名被解析为多个IP地址，那第一个被解析出来的IP地址会作为syslog服务器的地址。或者，也可以使用“UNIX-domain socket”的方式来指定。

2. facility=string ：设置syslog消息的facility（日志设施）。根据RFC3164的定义，Facility是 kern、user、mail、daemon、auth、intern、lpr、news、uucp、clock、authpriv、ftp、ntp, audit、alert、cron、local0 . . local7中的一个，默认值是 local7。

3. severity=string ：这个参数在error日志消息中是由nginx来决定的，不采用syslog的定义方式。severity这个参数仅适用于access_log。

   这个参数用于定义access_log消息的“严重（severity）”级别，根据RFC3164的定义，一共有8各级别，分别是：Emergency、Alert、Critical、Error、Warning、notice、Informational和Debug。

4. tag=string ：设置日志消息的标签。默认是“nginx”。

5. nohostname ：禁止将“hostname”这个字段添加到日志消息的头部。

下面的代码展示了将日志记录到syslog的配置：

```
error_log syslog:server=192.168.1.1 debug;

access_log syslog:server=unix:/var/log/nginx.sock,nohostname;
access_log syslog:server=[2001:db8::1]:12345,facility=local7,tag=nginx,severity=info combined;
```

2. access_log：访问日志配置

Nginx中主要通过三个指令来配置访问日志（access_log）：

1. log_format：定义日志的具体格式、日志条目中包含哪些信息、排列的顺序等。

2. access_log：指定访问日志的存储位置、使用哪种日志格式等。

3. open_log_file_cache：此指令来设置缓存，提升性能。

其中，log_format、access_log和open_log_file_cache位于Nginx文档的“Module ngx_http_log_module”部分。

2.1. log_format

log_format用来定义日志的具体格式、日志条目中包含哪些信息、排列的顺序等。一次定义，多次使用。它的父节点或上下文（context）是http。

log_format的配置格式如下：

```
log_format name [escape=default|json|none] string ...;
```

其中，*name*是要定义的日志格式的名称。这个名称会在access_log指令中被引用。有一个名字称为combined的“格式”已经被系统定义了。默认情况下，就是使用这个。combined中包含的变量及排列顺序如下：

```
log_format combined '$remote_addr - $remote_user [$time_local] '
                    '"$request" $status $body_bytes_sent '
                    '"$http_referer" "$http_user_agent"';
```

escape参数用来设置日志中变量的json或default字符转义（escaping）。默认情况下使用的是default。用Nginx官方的解释就是：

- default：对default来说，字符“”、“\”及其他值小于32大于126的字符，被转义为“\xXX”，如果变量的值没有找到，那将会使用连字符“-”来代替。
- json：对json来说，所有不允许出现在json字符串[1]里的字符都会被转义。

  字符“”和“\” 被转义为“\”和“\\”，值小于32的字符转义为“\n” “\r” “\t” “\b” “\f”或“\u00XX”。
- none：none就是禁用转义。

简尔言之，escape的目的就是将日志中很多变量值里的字符进行转义后再进行存储的。具体有哪些变量，在文后会详细的列出。

2.2. access_log

access_log指令是设置日志写入时的路径、格式及缓存相关配置的。它的设置语法如下：

```
access_log path [format [buffer=size] [gzip[=level]] [flush=time] [if=condition]];
access_log off;
```

1. path指的是日志的存储路径，如：*/var/log/nginx/access.log*
2. format指的是日志存储使用的格式，就是前面使用log_format指令定义好的格式，如果在这里没有使用，那么就使用系统预定义好的格式combined。

[1]关于json的相关知识请参考：https://datatracker.ietf.org/doc/html/rfc8259/

3. buffer设置日志缓冲区的大小。当缓冲区的日志数据超出该设定值时，缓冲区的日志数据就会被写到磁盘文件上。默认情况下，这个数值是64K。
4. gzip设置缓冲区日志数据的压缩级别，在缓冲区日志数据被写入到磁盘文件之前会被压缩。压缩的级别为1到9，数字约大，压缩越大。默认级别为1。
5. flush设置日志缓冲区刷新的时间间隔，当时间超过这个值时，缓冲区数据就会被写到磁盘文件。
6. if根据条件设置是否记录日志。当if所包含的条件为0或空时，日志不会被记录。

```
http {

    map $status $normal {
        ~^2  1;
        default 0;
    }
    map $status $abnormal {
        ~^2  0;
        default 1;
    }
    map $remote_addr $islocal {
        ~^127  1;
        default 0;}

    server {

        access_log logs/access.log combined if=$normal;
        access_log logs/access_abnormal.log combined if=$abnormal;
        access_log logs/access_local.log combined if=$islocal;

    }
}
```

7. off取消当前级别上的日志记录。

2.3. open_log_file_cache

open_log_file_cache默认配置下，Nginx每次将缓冲区（buffer） 里的日志数据写入到磁盘中时，都需要打开相应的日志文件，并获得该文件的句柄，利用该句柄向文件中写入数据，写入完成后关闭该句柄。该指令把句柄存储在缓存中，从而提升写入的效率。它的设置语法如下：

```
open_log_file_cache max=N [inactive=time] [min_uses=N] [valid=time];
open_log_file_cache off;
```

1. max设置缓存中最多容纳的文件描述符数量，如果被占满，采用LRU算法将描述符关闭。
2. inactive设置缓存存活时间，默认是10s。
3. min_uses在inactive时间段内，日志文件最少使用几次，该日志文件句柄记入缓存，默认是1次。
4. valid设置多久对日志文件名进行检查，看是否发生变化，默认是60s。
5. off不使用缓存。默认为off。

3. error_log：错误日志配置

error_log位于Nginx文档的“Core functionality”部分。

当Nginx的配置或者在运行时出现错误时，所产生的错误信息将会被记录到“错误日志”中。“错误日志”通过error_log指令来进行配置。与access_log不同的是，日志的格式不支持自定义。具体格式如下：

```
error_log file [level];
```

1. file指定错误日志的存放文件，如：*/var/log/error_log*。
2. [level]level表示日志等级，日志等级分为debug, info, notice, warn, error, crit, alert, emerg。可以看到其取值范围是按紧急程度从低到高排列的。debug的输出最为详细。只有日志的错误级别等于或高于level指定的值才会写入错误日志中。error是实际生产环境中常用的输出级别。默认值是error。

```
Default:
error_log logs/error.log error;
Context: main, http, mail, stream, server, location
```

需要注意的是：error_log off并不能关闭错误日志，而是会将错误日志记录到一个文件名为off的文件中。正确的关闭错误日志记录功能的方法如下：

```
error_log /dev/null;
```

第 3 章 配置案例及日志解析

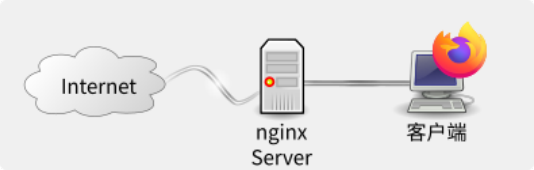
1. 环境说明

为了能更清晰地描述，我们在局域网中设立一台nginx服务器，我们称为Server，其内网IP地址是192.168.31.2/24，外网IP地址是192.168.1.2/24。我们在其上安装Debian Linux（V12）。使用如下命令安装nginx：

```
apt-get install nginx-full
```

安装结束后，nginx的配置文件位于*/etc/nginx*目录下，日志文件默认位于*/var/log/nginx*目录下。

图 3.1. nginx服务器网络拓扑图



2. 案例

2.1. 默认情况

在默认情况下，我们不使用log_format来命名和定义日志格式，使用nginx默认的combined格式。在前文中我们已经提到combined的格式如下：

```
log_format combined '$remote_addr - $remote_user [$time_local] '
                    '"$request" $status $body_bytes_sent '
                    '"$http_referer" "$http_user_agent"';
```

我们在*/etc/nginx/nginx.conf*中加入如下一行：

```
access_log /var/log/nginx/access.log;
```

接下来，使用/etc/init.d/nginx restart重新启动nginx的web服务。然后通过一台客户端的Chrome浏览器访问Server：*http://192.168.31.2/lognovel.html*，那么在Server的*/var/log/nginx/access.log*中，会出现如下一行记录：

```
192.168.31.7 - - [11/Oct/2021:09:07:43 +0000]
      "GET /lognovel.html HTTP/1.1" 200 418
      "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
           Chrome/94.0.4606.71 Safari/537.36"
```

上面的记录实际上是在一行里，为了描述方便，我们把它分为四行。根据combined的描述，很容易将这段日志解析出来：

1. 第一行：192.168.31.7是$remote_addr；第一个“-”是定义中固定的；第二个是$remote_user，因为这里$remote_user为空，所以被转义为“-”。
2. 第二行："GET /lognovel.html HTTP/1.1"是"$request"；200是$status[1]，200代表“ok”；418是$body_bytes_sent，返回给客户端的响应体的字节数（即不含响应头）
3. 第三行和第四行：“-”是指$http_referer，这里为空，所以使用“-”进行了转意。剩下的内容全部代表了客户端浏览器的信息。

2.2. 使用json转义存储日志

前面的两种情况使用的都是escape=default，下面我们再看一个escape=json的例子。我们仍然使用combined里的8个变量，在*/etc/nginx/nginx.conf*中添加如下的行：

```
log_format simplejson escape=json '$remote_addr - $remote_user [$time_local] '
                                  '"$request" $status $body_bytes_sent '
                                  '"$http_referer" "$http_user_agent"';
access_log /var/log/nginx/access.log simplejson;
```

在这个配置中，我们新增加了一个“日志格式”：simplejson，然后在access_log指令中引用它。它所使用的变量及排列顺序与combined完全一致。接下来，我们在客户端再次访问*http://192.168.31.2/lognovel.html*，然后看*/var/log/nginx/access.log*会出现什么？

```
192.168.31.7 -  [13/Oct/2021:03:42:58 +0000]
      "GET /lognovel.html HTTP/1.1" 200 418
      "" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
          Chrome/94.0.4606.71 Safari/537.36"
```

与escape=default比较，在escape=json情况下，$remote_user和$http_referer没有转义为“-”。而是使用了“空”。关于json字符的相关标准可以参考：https://datatracker.ietf.org/doc/html/rfc8259/

2.3. 自定义日志格式

combined中使用了8个字段（变量）来记录日志信息，接下来我们使用自定义的方式来记录日志信息，为了表述方便，我们只使用三个字段：时间戳$time_iso8601、客户端地址$remote_addr、客户端端口$remote_port以及请求信息$request，并且这四个字段使用“：”隔开。我们在*/etc/nginx/nginx.conf*文件中添加如下两行：

```
log_format simplelog '$time_iso8601 : $remote_addr : $remote_port : $request';
access_log /var/log/nginx/access.log simplelog;
```

我们在配置文件中做了两件事情：

1. 定义simplelog日志格式，只记录四个字段：$time_local、$remote_addr、$remote_port以及$request，并且这三个字段使用“：”隔开。

[1]更多状态码请参考：RFC2616 [https://datatracker.ietf.org/doc/html/rfc2616/]

2. 使用access_log指令来使用simplelog这个日志格式。

接下来通过客户端的Chrome浏览器再次访问Server：*http://192.168.31.2/lognovel.html*，那么在Server的*/var/log/nginx/access.log*中，会出现如下一行记录：

```
2021-10-12T07:23:47+00:00 : 192.168.31.7 : 56008 : GET /lognovel.html HTTP/1.1
```

可以看出，nginx已经按照要求记录了日志。从这条记录里，顺便还可以比较一下两个不同的时间戳变量$time_local与$time_iso8601的区别。

2.4. 使用完整的json格式

为方便后期的处理，也可以将日志存储为完全的json格式，如下，我们在配置文件中定义一个新的格式jsonlog：

```
log_format jsonlog  escape=json '{"@timestamp":"$time_iso8601",'
                '"server_addr":"$server_addr",'
                '"remote_addr":"$remote_addr",'
                '"host":"$host",'
                '"uri":"$uri",'
                '"body_bytes_sent":$body_bytes_sent,'
                '"bytes_sent":$body_bytes_sent,'
                '"upstream_response_time":$upstream_response_time,'
                '"request":"$request",'
                '"request_length":$request_length,'
                '"request_time":$request_time,'
                '"status":"$status",'
                '"http_referer":"$http_referer",'
                '"http_x_forwarded_for":"$http_x_forwarded_for",'
                '"http_user_agent":"$http_user_agent"'
                '}';

access_log /var/log/nginx/access.log jsonlog;
```

在这个配置中，我们新增加了一个“日志格式”：jsonlog，然后在access_log指令中引用它。接下来，我们在客户端再次访问*http://192.168.31.2/lognovel.html*，然后看*/var/log/nginx/access.log*会出现什么？

```
{"@timestamp":"2021-10-18T03:05:19+00:00",
 "server_addr":"192.168.31.2",
 "remote_addr":"192.168.31.7",
 "host":"192.168.31.2",
 "uri":"/lognovel.html",
 "body_bytes_sent":0,
 "bytes_sent":0,"upstream_response_time":,
 "request":"GET /lognovel.html HTTP/1.1",
 "request_length":543,
 "request_time":0.000,
 "status":"304",
 "http_referer":"",
 "http_x_forwarded_for":"",
 "http_user_agent":"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
                  Chrome/94.0.4606.81 Safari/537.36"
}
```

nginx已经生成了一条完整的json日志记录。

附录 A. 变量

nginx中有很多“嵌入式变量Embedded Variables”。nginx是模块化的，不同的模块中有不同的一些变量。大部分都在*ngx_http_core_module*模块中。这些变量有时候会作为判断条件出现在配置文件中、有时会作为占位符出现在配置文件中、有时会作为一个被记录的日志字段。所有的变量可以参考如下链接：https://nginx.org/en/docs/varindex.html

为方便查阅，我们对所有的变量按照所属模块，做了一个简单的梳理，并用表格的形式进行了汇总。

1. ngx_http_core_module中的变量

nginx所有的变量是与apache httpd web server的变量是匹配或一致的。

https://nginx.org/en/docs/http/ngx_http_core_module.html#variables

表 A.1. ngx_http_core_module中的变量

序号	名称	描述
1	$arg_name	argument name in the request line
2	$args	arguments in the request line
3	$binary_remote_addr	client address in a binary form, value’s length is always 4 bytes for IPv4 addresses or 16 bytes for IPv6 addresses
4	$body_bytes_sent	number of bytes sent to a client, not counting the response header; this variable is compatible with the “%B” parameter of the mod_log_config Apache module
5	$bytes_sent	number of bytes sent to a client (1.3.8, 1.2.5)
6	$connection	connection serial number (1.3.8, 1.2.5)
7	$connection_requests	current number of requests made through a connection (1.3.8, 1.2.5)
8	$connection_time	connection time in seconds with a milliseconds resolution (1.19.10)
9	$content_length	“Content-Length” request header field
10	$content_type	“Content-Type” request header field
11	$cookie_name	the name cookie
12	$document_root	root or alias directive’s value for the current request
13	$document_uri	same as $uri
14	$host	in this order of precedence: host name from the request line, or host name from the “Host” request header field, or the server name matching a request
15	$hostname	host name
16	$http_name	arbitrary request header field; the last part of a variable name is the field name converted to lower case with dashes replaced by underscores
17	$https	“on” if connection operates in SSL mode, or an empty string otherwise

序号	名称	描述
18	$is_args	“?” if a request line has arguments, or an empty string otherwise
19	$limit_rate	setting this variable enables response rate limiting; see limit_rate
20	$msec	current time in seconds with the milliseconds resolution (1.3.9, 1.2.6)
21	$nginx_version	nginx version
22	$pid	PID of the worker process
23	$pipe	“p” if request was pipelined, “.” otherwise (1.3.12, 1.2.7)
24	$proxy_protocol_addr	client address from the PROXY protocol header (1.5.12) The PROXY protocol must be previously enabled by setting the proxy_protocol parameter in the listen directive.
25	$proxy_protocol_port	client port from the PROXY protocol header (1.11.0) The PROXY protocol must be previously enabled by setting the proxy_protocol parameter in the listen directive.
26	$proxy_protocol_server_addr	server address from the PROXY protocol header (1.17.6) The PROXY protocol must be previously enabled by setting the proxy_protocol parameter in the listen directive.
27	$proxy_protocol_server_port	server port from the PROXY protocol header (1.17.6) The PROXY protocol must be previously enabled by setting the proxy_protocol parameter in the listen directive.
28	$query_string	same as $args
29	$realpath_root	an absolute pathname corresponding to the root or alias directive’s value for the current request, with all symbolic links resolved to real paths
30	$remote_addr	client address
31	$remote_port	client port
32	$remote_user	user name supplied with the Basic authentication
33	$request	full original request line
34	$request_body	request body The variable’s value is made available in locations processed by the proxy_pass, fastcgi_pass, uwsgi_pass, and scgi_pass directives when the request body was read to a memory buffer.
35	$request_body_file	name of a temporary file with the request body At the end of processing, the file needs to be removed. To always write the request body to a file, client_body_in_file_only needs to be enabled. When the name of a temporary file is passed in a proxied request or in a request to a FastCGI/uwsgi/SCGI server, passing the request body should be disabled by the proxy_pass_request_body off, fastcgi_pass_request_body off, uwsgi_pass_request_body off, or scgi_pass_request_body off directives, respectively.

序号	名称	描述
36	$request_completion	“OK” if a request has completed, or an empty string otherwise
37	$request_method	request method, usually “GET” or “POST”
38	$request_time	request processing time in seconds with a milliseconds resolution (1.3.9, 1.2.6); time elapsed since the first bytes were read from the client
39	$request_uri	full original request URI (with arguments)
40	$scheme	request scheme, “http” or “https”
41	$sent_http_name	arbitrary response header field; the last part of a variable name is the field name converted to lower case with dashes replaced by underscores
42	$sent_trailer_name	arbitrary field sent at the end of the response (1.13.2); the last part of a variable name is the field name converted to lower case with dashes replaced by underscores
43	$server_addr	an address of the server which accepted a request Computing a value of this variable usually requires one system call. To avoid a system call, the listen directives must specify addresses and use the bind parameter.
44	$server_name	name of the server which accepted a request
45	$server_port	port of the server which accepted a request
46	$server_protocol	request protocol, usually “HTTP/1.0”, “HTTP/1.1”, or “HTTP/2.0”
47	$status	response status (1.3.2, 1.2.2)
48	$tcpinfo_rtt, $tcpinfo_rttvar, $tcpinfo_snd_cwnd, $tcpinfo_rcv_space	information about the client TCP connection; available on systems that support the TCP_INFO socket option
49	$time_iso8601	local time in the ISO 8601 standard format (1.3.12, 1.2.7)
50	$time_local	local time in the Common Log Format (1.3.12, 1.2.7)
51	$uri	current URI in request, normalized The value of $uri may change during request processing, e.g. when doing internal redirects, or when using index files.
52	$http_user_agent	The User-Agent request header is a characteristic string that lets servers and network peers identify the application, operating system, vendor, and/or version of the requesting user agent.
53	$http_cookie	web cookie

2. ngx_http_auth_jwt_module中的变量

https://nginx.org/en/docs/http/ngx_http_auth_jwt_module.html#variables

表 A.2. ngx_http_auth_jwt_module中的变量

序号	名称	描述
1	$jwt_header_name	returns the value of a specified JOSE header
2	$jwt_claim_name	returns the value of a specified JWT claim
3	$jwt_payload	returns the decrypted top-level payload of nested or encrypted tokens (1.21.2). For nested tokens returns the enclosed JWS token. For encrypted tokens returns JSON with claims.

3. ngx_http_fastcgi_module中的变量

https://nginx.org/en/docs/http/ngx_http_fastcgi_module.html#variables

表 A.3. ngx_http_fastcgi_module中的变量

序号	名称	描述
1	$fastcgi_script_name	request URI or, if a URI ends with a slash, request URI with an index file name configured by the fastcgi_index directive appended to it. This variable can be used to set the SCRIPT_FILENAME and PATH_TRANSLATED parameters that determine the script name in PHP. For example, for the “/info/” request with the following directives
2	$fastcgi_path_info	the value of the second capture set by the fastcgi_split_path_info directive. This variable can be used to set the PATH_INFO parameter.

4. ngx_http_gzip_module中的变量

https://nginx.org/en/docs/http/ngx_http_gzip_module.html#variables

表 A.4. ngx_http_gzip_module中的变量

序号	名称	描述
1	$gzip_ratio	achieved compression ratio, computed as the ratio between the original and compressed response sizes.

5. ngx_http_limit_conn_module中的变量

https://nginx.org/en/docs/http/ngx_http_limit_conn_module.html#variables

表 A.5. ngx_http_limit_conn_module中的变量

序号	名称	描述
1	$limit_conn_status	keeps the result of limiting the number of connections (1.17.6): PASSED, REJECTED, or REJECTED_DRY_RUN

6. ngx_http_limit_req_module中的变量

https://nginx.org/en/docs/http/ngx_http_limit_req_module.html#variables

表 A.6. ngx_http_limit_req_module中的变量

序号	名称	描述
1	$limit_req_status	keeps the result of limiting the request processing rate (1.17.6): PASSED, DELAYED, REJECTED, DELAYED_DRY_RUN, or REJECTED_DRY_RUN

7. ngx_http_memcached_module中的变量

https://nginx.org/en/docs/http/ngx_http_memcached_module.html#variables

表 A.7. ngx_http_memcached_module中的变量

序号	名称	描述
1	$memcached_key	Defines a key for obtaining response from a memcached server.

8. ngx_http_proxy_module中的变量

https://nginx.org/en/docs/http/ngx_http_proxy_module.html

表 A.8. ngx_http_proxy_module中的变量

序号	名称	描述
1	$proxy_host	name and port of a proxied server as specified in the proxy_pass directive;
2	$proxy_port	port of a proxied server as specified in the proxy_pass directive, or the protocol’s default port;
3	$proxy_add_x_forwarded_for	the “X-Forwarded-For” client request header field with the $remote_addr variable appended to it, separated by a comma. If the “X-Forwarded-For” field is not present in the client request header, the $proxy_add_x_forwarded_for variable is equal to the $remote_addr variable.

9. ngx_http_realip_module中的变量

https://nginx.org/en/docs/http/ngx_http_realip_module.html#variables

表 A.9. ngx_http_realip_module中的变量

序号	名称	描述
1	$realip_remote_addr	keeps the original client address (1.9.7)
2	$realip_remote_port	keeps the original client port (1.11.0)

10. ngx_http_referer_module中的变量

https://nginx.org/en/docs/http/ngx_http_referer_module.html#variables

表 A.10. ngx_http_referer_module中的变量

序号	名称	描述
1	$invalid_referer	Empty string, if the “Referer” request header field value is considered valid, otherwise “1”.

11. ngx_http_secure_link_module中的变量

https://nginx.org/en/docs/http/ngx_http_secure_link_module.html#variables

表 A.11. ngx_http_secure_link_module中的变量

序号	名称	描述
1	$secure_link	The status of a link check. The specific value depends on the selected operation mode.
2	$secure_link_expires	The lifetime of a link passed in a request; intended to be used only in the secure_link_md5 directive.

12. ngx_http_session_log_module中的变量

https://nginx.org/en/docs/http/ngx_http_session_log_module.html#variables

表 A.12. ngx_http_session_log_module中的变量

序号	名称	描述
1	$session_log_id	current session ID;
2	$session_log_binary_id	current session ID in binary form (16 bytes).

13. ngx_http_slice_module中的变量

https://nginx.org/en/docs/http/ngx_http_slice_module.html#variables

表 A.13. ngx_http_slice_module中的变量

序号	名称	描述
1	$slice_range	the current slice range in HTTP byte range format, for example, bytes=0-1048575.

14. ngx_http_spdy_module中的变量

https://nginx.org/en/docs/http/ngx_http_spdy_module.html#variables

表 A.14. ngx_http_spdy_module中的变量

序号	名称	描述
1	$spdy	SPDY protocol version for SPDY connections, or an empty string otherwise;
2	$spdy_request_priority	request priority for SPDY connections, or an empty string otherwise.

15. ngx_http_ssi_module中的变量

https://nginx.org/en/docs/http/ngx_http_ssi_module.html#variables

表 A.15. ngx_http_ssi_module中的变量

序号	名称	描述
1	$date_local	current time in the local time zone. The format is set by the config command with the timefmt parameter

序号	名称	描述
2	$date_gmt	current time in GMT. The format is set by the config command with the timefmt parameter.

16. ngx_http_ssl_module中的变量

https://nginx.org/en/docs/http/ngx_http_ssl_module.html#variables

表 A.16. ngx_http_ssl_module中的变量

序号	名称	描述
1	$ssl_cipher	returns the name of the cipher used for an established SSL connection;
2	$ssl_ciphers	returns the list of ciphers supported by the client (1.11.7). Known ciphers are listed by names, unknown are shown in hexadecimal
3	$ssl_client_escaped_cert	returns the client certificate in the PEM format (urlencoded) for an established SSL connection (1.13.5);
4	$ssl_client_cert	returns the client certificate in the PEM format for an established SSL connection, with each line except the first prepended with the tab character; this is intended for the use in the proxy_set_header directive;
5	$ssl_client_fingerprint	returns the SHA1 fingerprint of the client certificate for an established SSL connection (1.7.1);
6	$ssl_client_i_dn	returns the “issuer DN” string of the client certificate for an established SSL connection according to RFC 2253 (1.11.6);
7	$ssl_client_i_dn_legacy	returns the “issuer DN” string of the client certificate for an established SSL connection;
8	$ssl_client_raw_cert	returns the client certificate in the PEM format for an established SSL connection;
9	$ssl_client_s_dn	returns the “subject DN” string of the client certificate for an established SSL connection according to RFC 2253 (1.11.6);
10	$ssl_client_s_dn_legacy	returns the “subject DN” string of the client certificate for an established SSL connection;
11	$ssl_client_serial	returns the serial number of the client certificate for an established SSL connection;
12	$ssl_client_v_end	returns the end date of the client certificate (1.11.7);
13	$ssl_client_v_remain	returns the number of days until the client certificate expires (1.11.7);
14	$ssl_client_v_start	returns the start date of the client certificate (1.11.7);
15	$ssl_client_verify	returns the result of client certificate verification: “SUCCESS”, “FAILED:reason”, and “NONE” if a certificate was not present;

序号	名称	描述
16	$ssl_curves	returns the list of curves supported by the client (1.11.7). Known curves are listed by names, unknown are shown in hexadecimal,
17	$ssl_early_data	returns “1” if TLS 1.3 early data is used and the handshake is not complete, otherwise “” (1.15.3).
18	$ssl_protocol	returns the protocol of an established SSL connection;
19	$ssl_server_name	returns the server name requested through SNI (1.7.0);
20	$ssl_session_id	returns the session identifier of an established SSL connection;
21	$ssl_session_reused	returns “r” if an SSL session was reused, or “.” otherwise (1.5.11).

17. ngx_http_stub_status_module中的变量

https://nginx.org/en/docs/http/ngx_http_stub_status_module.html#variables

表 A.17. ngx_http_stub_status_module中的变量

序号	名称	描述
1	$connections_active	The current number of active client connections including Waiting connections.
2	$connections_reading	The current number of connections where nginx is reading the request header.
3	$connections_writing	The current number of connections where nginx is writing the response back to the client.
4	$connections_waiting	The current number of idle client connections waiting for a request.

18. ngx_http_upstream_module中的变量

https://nginx.org/en/docs/http/ngx_http_upstream_module.html#variables

表 A.18. ngx_http_upstream_module中的变量

序号	名称	描述
1	$upstream_addr	keeps the IP address and port, or the path to the UNIX-domain socket of the upstream server. If several servers were contacted during request processing, their addresses are separated by commas, e.g. “192.168.1.1:80, 192.168.1.2:80, unix:/tmp/sock”. If an internal redirect from one server group to another happens, initiated by “X-Accel-Redirect” or error_page, then the server addresses from different groups are separated by colons, e.g. “192.168.1.1:80, 192.168.1.2:80, unix:/tmp/sock : 192.168.10.1:80, 192.168.10.2:80”. If a server cannot be selected, the variable keeps the name of the server group.

序号	名称	描述
2	$upstream_bytes_received	number of bytes received from an upstream server (1.11.4). Values from several connections are separated by commas and colons like addresses in the $upstream_addr variable.
3	$upstream_bytes_sent	number of bytes sent to an upstream server (1.15.8). Values from several connections are separated by commas and colons like addresses in the $upstream_addr variable.
4	$upstream_cache_status	keeps the status of accessing a response cache (0.8.3). The status can be either “MISS”, “BYPASS”, “EXPIRED”, “STALE”, “UPDATING”, “REVALIDATED”, or “HIT”.
5	$upstream_connect_time	keeps time spent on establishing a connection with the upstream server (1.9.1); the time is kept in seconds with millisecond resolution. In case of SSL, includes time spent on handshake. Times of several connections are separated by commas and colons like addresses in the $upstream_addr variable.
6	$upstream_cookie_name	cookie with the specified name sent by the upstream server in the “Set-Cookie” response header field (1.7.1). Only the cookies from the response of the last server are saved.
7	$upstream_header_time	keeps time spent on receiving the response header from the upstream server (1.7.10); the time is kept in seconds with millisecond resolution. Times of several responses are separated by commas and colons like addresses in the $upstream_addr variable.
8	$upstream_http_name	keep server response header fields. For example, the “Server” response header field is available through the $upstream_http_server variable. The rules of converting header field names to variable names are the same as for the variables that start with the “$http_” prefix. Only the header fields from the response of the last server are saved.
9	$upstream_queue_time	keeps time the request spent in the upstream queue (1.13.9); the time is kept in seconds with millisecond resolution. Times of several responses are separated by commas and colons like addresses in the $upstream_addr variable.
10	$upstream_response_length	keeps the length of the response obtained from the upstream server (0.7.27); the length is kept in bytes. Lengths of several responses are separated by commas and colons like addresses in the $upstream_addr variable.
11	$upstream_response_time	keeps time spent on receiving the response from the upstream server; the time is kept in seconds with millisecond resolution. Times of several responses are separated by commas and colons like addresses in the $upstream_addr variable.

序号	名称	描述
12	$upstream_status	keeps status code of the response obtained from the upstream server. Status codes of several responses are separated by commas and colons like addresses in the $upstream_addr variable. If a server cannot be selected, the variable keeps the 502 (Bad Gateway) status code.
13	$upstream_trailer_name	keeps fields from the end of the response obtained from the upstream server (1.13.10).

19. ngx_http_userid_module中的变量

https://nginx.org/en/docs/http/ngx_http_userid_module.html#variables

表 A.19. ngx_http_userid_module中的变量

序号	名称	描述
1	$uid_got	The cookie name and received client identifier.
2	$uid_reset	If the variable is set to a non-empty string that is not “0”, the client identifiers are reset. The special value “log” additionally leads to the output of messages about the reset identifiers to the error_log.
3	$uid_set	The cookie name and sent client identifier.

20. ngx_http_v2_module中的变量

https://nginx.org/en/docs/http/ngx_http_v2_module.html#variables

表 A.20. ngx_http_v2_module中的变量

序号	名称	描述
1	$http2	negotiated protocol identifier: “h2” for HTTP/2 over TLS, “h2c” for HTTP/2 over cleartext TCP, or an empty string otherwise.

21. ngx_stream_core_module中的变量

https://nginx.org/en/docs/stream/ngx_stream_core_module.html#variables

表 A.21. ngx_stream_core_module中的变量

序号	名称	描述
1	$binary_remote_addr	client address in a binary form, value’s length is always 4 bytes for IPv4 addresses or 16 bytes for IPv6 addresses
2	$bytes_received	number of bytes received from a client (1.11.4)
3	$bytes_sent	number of bytes sent to a client
4	$connection	connection serial number
5	$hostname	host name
6	$msec	current time in seconds with the milliseconds resolution
7	$nginx_version	nginx version
8	$pid	PID of the worker process

序号	名称	描述
9	$protocol	protocol used to communicate with the client: TCP or UDP (1.11.4)
10	$proxy_protocol_addr	client address from the PROXY protocol header (1.11.4). The PROXY protocol must be previously enabled by setting the proxy_protocol parameter in the listen directive.
11	$proxy_protocol_port	client port from the PROXY protocol header (1.11.4) .The PROXY protocol must be previously enabled by setting the proxy_protocol parameter in the listen directive.
12	$proxy_protocol_server_addr	server address from the PROXY protocol header (1.17.6) .The PROXY protocol must be previously enabled by setting the proxy_protocol parameter in the listen directive.
13	$proxy_protocol_server_port	server port from the PROXY protocol header (1.17.6) .The PROXY protocol must be previously enabled by setting the proxy_protocol parameter in the listen directive.
14	$remote_addr	client address
15	$remote_port	client port
16	$server_addr	an address of the server which accepted a connection .Computing a value of this variable usually requires one system call. To avoid a system call, the listen directives must specify addresses and use the bind parameter.
17	$server_port	port of the server which accepted a connection
18	$session_time	session duration in seconds with a milliseconds resolution (1.11.4);
19	$status	session status (1.11.4), can be one of the following: 200 session completed successfully 400 client data could not be parsed, for example, the PROXY protocol header 403 access forbidden, for example, when access is limited for certain client addresses 500 internal server error 502 bad gateway, for example, if an upstream server could not be selected or reached. 503 service unavailable, for example, when access is limited by the number of connections
20	$time_iso8601	local time in the ISO 8601 standard format
21	$time_local	local time in the Common Log Format

22. ngx_stream_limit_conn_module中的变量

https://nginx.org/en/docs/stream/ngx_stream_limit_conn_module.html#variables

表 A.22. ngx_stream_limit_conn_module中的变量

序号	名称	描述
1	$limit_conn_status	keeps the result of limiting the number of connections (1.17.6): PASSED, REJECTED, or REJECTED_DRY_RUN

23. ngx_stream_realip_module中的变量

https://nginx.org/en/docs/stream/ngx_stream_realip_module.html#variables

表 A.23. ngx_stream_realip_module中的变量

序号	名称	描述
1	$realip_remote_addr	keeps the original client address
1	$realip_remote_port	keeps the original client port

24. ngx_stream_ssl_module中的变量

https://nginx.org/en/docs/stream/ngx_stream_ssl_module.html#variables

表 A.24. ngx_stream_ssl_module中的变量

序号	名称	描述
1	$ssl_cipher	returns the name of the cipher used for an established SSL connection;
2	$ssl_ciphers	returns the list of ciphers supported by the client (1.11.7). Known ciphers are listed by names, unknown are shown in hexadecimal,
3	$ssl_client_cert	returns the client certificate in the PEM format for an established SSL connection, with each line except the first prepended with the tab character (1.11.8);
4	$ssl_client_fingerprint	returns the SHA1 fingerprint of the client certificate for an established SSL connection (1.11.8);
5	$ssl_client_i_dn	returns the “issuer DN” string of the client certificate for an established SSL connection according to RFC 2253 (1.11.8);
6	$ssl_client_raw_cert	returns the client certificate in the PEM format for an established SSL connection (1.11.8);
7	$ssl_client_s_dn	returns the “subject DN” string of the client certificate for an established SSL connection according to RFC 2253 (1.11.8);
8	$ssl_client_serial	returns the serial number of the client certificate for an established SSL connection (1.11.8);
9	$ssl_client_v_end	returns the end date of the client certificate (1.11.8);
10	$ssl_client_v_remain $ssl_client_v_remain	returns the number of days until the client certificate expires (1.11.8);
11	$ssl_client_v_start	returns the start date of the client certificate (1.11.8);
12	$ssl_client_verify	returns the result of client certificate verification (1.11.8): “SUCCESS”, “FAILED:reason”, and “NONE” if a certificate was not present;
13	$ssl_curves	returns the list of curves supported by the client (1.11.7). Known curves are listed by names, unknown are shown in hexadecimal,
14	$ssl_protocol	returns the protocol of an established SSL connection;

序号	名称	描述
15	$ssl_server_name	returns the server name requested through SNI;
16	$ssl_session_id	returns the session identifier of an established SSL connection;
17	$ssl_session_reused	returns “r” if an SSL session was reused, or “.” otherwise.

25. ngx_stream_ssl_preread_module中的变量

https://nginx.org/en/docs/stream/ngx_stream_ssl_preread_module.html#variables

表 A.25. ngx_stream_ssl_preread_module中的变量

序号	名称	描述
1	$ssl_preread_protocol	the highest SSL protocol version supported by the client (1.15.2)
2	$ssl_preread_server_name	server name requested through SNI
3	$ssl_preread_alpn_protocols	list of protocols advertised by the client through ALPN (1.13.10). The values are separated by commas.

26. ngx_stream_upstream_module中的变量

https://nginx.org/en/docs/stream/ngx_stream_upstream_module.html#variables

表 A.26. ngx_stream_upstream_module中的变量

序号	名称	描述
1	$upstream_addr	keeps the IP address and port, or the path to the UNIX-domain socket of the upstream server (1.11.4). If several servers were contacted during proxying, their addresses are separated by commas, e.g. “192.168.1.1:12345, 192.168.1.2:12345, unix:/tmp/sock”. If a server cannot be selected, the variable keeps the name of the server group.
2	$upstream_bytes_received	number of bytes received from an upstream server (1.11.4). Values from several connections are separated by commas like addresses in the $upstream_addr variable.
3	$upstream_bytes_sent	number of bytes sent to an upstream server (1.11.4). Values from several connections are separated by commas like addresses in the $upstream_addr variable.
4	$upstream_connect_time	time to connect to the upstream server (1.11.4); the time is kept in seconds with millisecond resolution. Times of several connections are separated by commas like addresses in the $upstream_addr variable.
5	$upstream_first_byte_time	time to receive the first byte of data (1.11.4); the time is kept in seconds with millisecond resolution. Times of several connections are separated by commas like addresses in the $upstream_addr variable.

序号	名称	描述
6	$upstream_session_time	session duration in seconds with millisecond resolution (1.11.4). Times of several connections are separated by commas like addresses in the $upstream_addr variable.

附录 B. HTTP/1.1状态码

响应状态，也就是nginx的$status变量。关于响应状态，最著名的就是“404”,代表Not Found，用于说明服务器无法找到所请求的URL。

关于状态码更详细的信息可以参考**HTTP/1.1**协议的详细说明：https://www.rfc-editor.org/rfc/inline-errata/rfc2616.html

在这里我们仅做简要说明。HTTP状态码被分成了五大类。由以1、2、3、4、5开头的三个数字组成。

1. **1xx**状态码

   1字头状态码是信息性状态码，HTTP/1.1向协议中引入了信息性状态码。这些状态码相对较新，关于其复杂性和其价值存在一些争论。

   表 B.1. 1字头状态码

状态码	原因短语	描述
100	Continue	说明收到了请求的初始部分，请客户端继续。发送了这个状态码之后，服务器在收到请求之后必须进行响应。
101	SwitchingProtocols	说明服务器正在根据客户端的指定，将协议切换成Update

2. **2xx**状态码

   2字头状态码是成功状态码，客户端发起请求时，这些请求通常都是成功的。

   表 B.2. 2字头状态码

状态码	原因短语	描述
200	OK	请求没问题，实体的主体部分包含了所请求的资源。
201	Created	用于创建服务器对象的请求(比如， PUT)。响应的实体主体部分中应该包含各引用了已创建的资源的URL。
202	Accepted	请求已被接受，但服务器还未对其执行任何动作。不能保证服务器会完成这个请求。
203	Non- Authoritative Information	实体首部包含的信息不是来自于源端服务器，而是可能调用了其他的资源。
204	No Content	响应报文中包含若干首部和一个状态行，但没有实体的主体部分。
205	Reset Content	另一个主要用于浏览器的代码。负责告知浏览器清除当前页面中的所有HTML表单元素。
206	Partial Content	成功执行了一个部分或Range (范围)请求。

3. **3xx**状态码

   3字头状态码是重定向状态码。重定向状态码要么告知客户端使用替代位置来访问他们所感兴趣的资源，要么就提供一个替代的响应而不是资源的内容。如果资源已被移动，可发送一个重定向状态码和一个可选的Location首部来告知客户端资源已被移走，以及现在可以在哪里找到它。这样，浏览器就可以在不打扰使用者的情况下，透明地转入新的位置了。

表 B.3. 3字头状态码

状态码	原因短语	描述
300	Multiple Choices	客户端请求一个实际指向多个资源的URL时会返回这个状态码，比如服务器上有某个HTML文档的中文和英文版本。返回这个代码时会带有一个选项列表，这样用户就可以选择他希望使用的那一项了。
301	Moved Permanently	在请求的 URL已被移除时使用。
302	Found	与301状态码类似;但是，客户端应该使用Location首部给出的URL来临时定位资源。
303	See Other	告知客户端应该用另一个URL来获取资源。新的URL位于响应报文的Location首部。其主要目的是允许POST请求的响应将客户端定向到某个资源上去
304	Not Modified	客户端可以通过所包含的请求首部，使其请求变成有条件的。如果客户端发起了一个条件GET请求，而最近资源未被修改的话，就可以用这个状态码来说明资源未被修改。带有这个状态码的响应不应该包含实体的主体部分
305	Use Proxy	用来说明必须通过一个代理来访问资源;代理的位置由Location首部给出。
306	null	这个状态码目前还未使用。
307	Temporary Redirect	与301状态码类似;但客户端应该使用Location首部给出的URL来临时定位资源。

4. **4xx**状态码

4字头状态码是客户端错误状态码，有时客户端会发送一些服务器无法处理的东西，比如格式错误的请求报文，或者最常见的是，请求一个不存在的URL。

表 B.4. 4字头状态码

状态码	原因短语	描述
400	Bad Request	用于告知客户端它发送了一个错误的请求
401	Unauthorized	与适当的首部一同返回，在这些首部中请求客户端在获取对资源的访问权之前，对自己进行认证。
402	Payment Required	现在这个状态码还未使用，但已经被保留，以作未来之用。
403	Forbidden	用于说明请求被服务器拒绝了。如果服务器想说明为什么拒绝请求，可以包含实体的主体部分来对原因进行描述。但这个状态码通常是在服务器不想说明拒绝原因的时候使用。
404	Not Found	用于说明服务器无法找到所请求的URL。
405	Method Not Allowed	发起的请求中带有所请求的URL不支持的方法时，使用此状态码。
406	Not Acceptable	客户端可以指定参数来说明它们愿意接收什么类型的实体。服务器没有与客户端可接受的URL相匹配的资源时，使用此代码。
407	Proxy Authentication	与401状态码类似，但用于要求对资源进行认证的代理服务器。

状态码	原因短语	描述
408	Request Timeout	如果客户端完成请求所花的时间太长，服务器可以回送此状态码，并关闭连接。超时时长随服务器的不同有所不同。
409	Conflict	用于说明请求可能在资源上引发的一些冲突。服务器担心请求会引发冲突时，可以发送此状态码。
410	Gone	与404类似，只是服务器曾经拥有过此资源。主要用于Web站点的维护，这样服务器的管理者就可以在资源被移除的情况下通知客户端。
411	Length Required	服务器要求在请求报文中包含Content-Length首部时使用。
412	Precondition Failed	客户端发起了条件请求，且其中一个条件失败了的时候使用。
413	RequestEntity TooLarge	客户端发送的实体主体部分比服务器能够或者希望处理的要大时，使用此状态码。
414	Request URI Too Long	客户端所发请求中的请求URL比服务器能够或者希望处理的要长时，使用此状态码。
415	Unsupported Media Type	服务器无法理解或无法支持客户端所发实体的内容类型时，使用此状态码。
416	Requested Range Not	请求报文所请求的是指定资源的某个范围，而此范围无效或无法满Satisfiable足时，使用此状态码。
417	Expectation Failed	请求的Expect请求首部包含了一个期望，但服务器无法满足此期望时，使用此状态码。

5. **5xx**状态码

5字头状态码是服务器错误状态码，有时客户端发送了一条有效请求，服务器自身却出错了。

表 B.5. 5字头状态码

状态码	原因短语	描述
500	Internal Server Error	服务器遇到一个妨碍它为请求提供服务的错误时，使用此状态码。
501	Not Implemented	客户端发起的请求超出服务器的能力范围(比如，使用了服务器不支持的请求方法)时，使用此状态码。
502	Bad Gateway	作为代理或网关使用的服务器从请求响应链的下一条链路上收到了一条伪响应(比如，它无法连接到其父网关)时，使用此状态码。
503	Service Unavailable	用来说明服务器现在无法为请求提供服务。
504	Gateway Timeout	与状态码408类似，只是这里的响应来自一个网关或代理，它们在等待另一服务器对其请求进行响应时超时了。
505	HTTP Version Not	服务器收到的请求使用了它无法或不愿支持的协议版本时，使用此状态码。