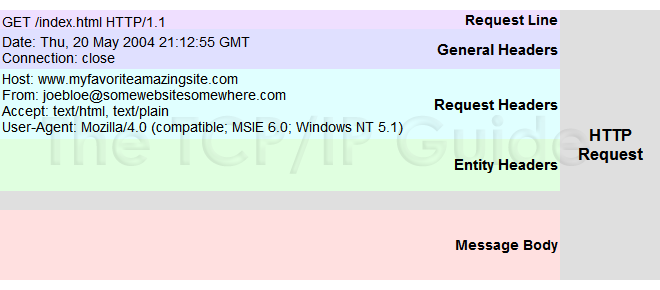
一次完整的HTTP请求过程从TCP三次握手建立连接成功后开始，客户端按照指定的格式开始向服务端发送HTTP请求，服务端接收请求后，解析HTTP请求，处理完业务逻辑，最后返回一个HTTP的响应给客户端，HTTP的响应内容同样有标准的格式。无论是什么客户端或者是什么服务端，大家只要按照HTTP的协议标准来实现的话，那么它一定是通用的。

**HTTP 请求格式**

HTTP请求格式主要有四部分组成，分别是：请求行、请求头、空行、消息体，每部分内容占一行

|  |  |
| --- | --- |
| 1 2 3 4 5 6 | <request-line> <general-headers> <request-headers> <entity-headers> <empty-line> [<message-body>] |



**请求行**：请求行是请求消息的第一行，由三部分组成：分别是请求方法（GET/POST/DELETE/PUT/HEAD）、请求资源的URI路径、HTTP的版本号

|  |  |
| --- | --- |
| 1 | GET /index.html HTTP/1.1 |

**请求头**：请求头中的信息有和缓存相关的头（Cache-Control，If-Modified-Since）、客户端身份信息（User-Agent）等等。例如：

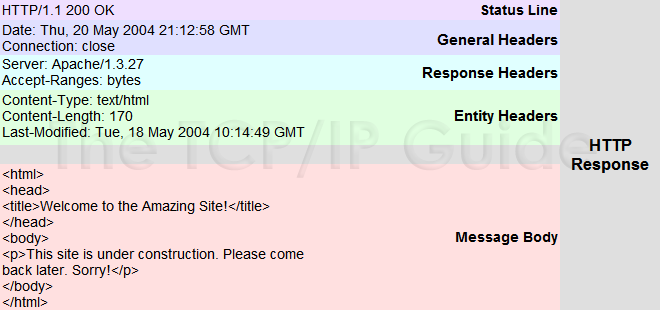
|  |  |
| --- | --- |
| 1 2 3 4 | Cache-Control:max-age=0 Cookie:gsScrollPos=; \_ga=GA1.2.329038035.1465891024; \_gat=1 If-Modified-Since:Sun, 01 May 2016 11:19:03 GMT User-Agent:Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.84 Safari/537.36 |

**消息体**：请求体是客户端发给服务端的请求数据，这部分数据并不是每个请求必须的。

**HTTP 响应格式**

服务器接收处理完请求后返回一个HTTP相应消息给客户端。HTTP响应消息的格式包括：状态行、响应头、空行、消息体。每部分内容占一行。

|  |  |
| --- | --- |
| 1 2 3 4 5 6 | <status-line> <general-headers> <response-headers> <entity-headers> <empty-line> [<message-body>] |



**状态行**：状态行位于相应消息的第一行，有HTTP协议版本号，状态码和状态说明三部分构成。如：

|  |  |
| --- | --- |
| 1 | HTTP/1.1 200 OK |

**响应头**：响应头是服务器传递给客户端用于说明服务器的一些信息，以及将来继续访问该资源时的策略。

|  |  |
| --- | --- |
| 1 2 3 4 5 6 | Connection:keep-alive Content-Encoding:gzip Content-Type:text/html; charset=utf-8 Date:Fri, 24 Jun 2016 06:23:31 GMT Server:nginx/1.9.12 Transfer-Encoding:chunked |

**响应体**：响应体是服务端返回给客户端的HTML文本内容，或者其他格式的数据，比如：视频流、图片或者音频数据。

**Socket**

WEB Server都是基于Socket编程，又称之为网络编程，网络协议通过一个叫做socket的对象抽象出来，socket可以建立网络连接，读数据，写数据。socket模块定义了一些常量参数，用来指定socket的的地址族、socket的类型、以及支持的TCP/IP协议。

socket.socket([family[, type[, proto]]])：根据指定的地址族和套接字类型、协议编号（默认为0）来创建套接字对象。AF\_INET对应的IPV4, AF\_INET6对应的IPV6。



**Socket 对象方法**

* socket.bind(address)：绑定IP地址以及端口
* socket.listen(backlog) ：在指定的端口开始监听，backlog表示connection队列的最大长度
* socket.setblocking(flag) ： 设置为非阻塞还是阻塞的socket，如果是非阻塞的，那么调用recv的时候如果没有数据可读，那么久直接返回一个错误，相反如果设置为阻塞模式，如果没有数据可读，那么就一直处于阻塞等待数据的状态。
* socket.accept()：当有连接请求过来时，接收该连接，返回一个socket对象，该对象可以在基于该连接发送和接收数据。
* socket.sendall(string[, flags])：发送数据
* socket.recv(bufsize[, flags])：接收数据
* socket.close()：关闭socket连接。

搞清楚了HTTP规范和Socket之后，我们就可以使用Socket实现一个对简单的HTTP服务器了。代码：

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 | # -\*- coding:utf-8 -\*-  import socket  if \_\_name\_\_ == '\_\_main\_\_': PORT = 8000 sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) sock.bind(('127.0.0.1', PORT)) sock.listen(1) print 'Serving HTTP on port %s ...' % PORT  while 1: conn, addr = sock.accept() print conn, addr request = conn.recv(1024) # HTTP响应消息 response = "HTTP/1.1 200 OK\nContent-Type:text/html\nServer:myserver\n\nHello, World!" conn.sendall(response) conn.close() |

浏览器访问地址：[http://localhost:8000](http://localhost:8000/)

