

3-静态Web服务器

3-1 HTTP协议

HTTP协议（HyperText Transfer Protocol），超文本传输协议。

作用：规定了浏览器和Web服务器通信数据的格式。

URL（Uniform Resource Locator），统一资源定位符。

组成：协议部分--域名部分--资源路径部分。

浏览器开发者工具

标签选项说明：

- 元素（Elements）：用于查看或修改HTML标签
- 控制台（Console）：执行js代码
- 源代码（Sources）：查看静态资源文件，断点调试JS代码
- 网络（Network）：查看http协议的通信过程

使用说明：

- 1、点击Network标签选项
- 2、在浏览器的地址输入网址，即可看到请求Http的通信过程
- 3、每项记录都是请求+相应的一次过程

3-2 HTTP请求报文

一个HTTP请求报文可以由请求行、请求头、空行和请求体4个部分组成

1、GET请求报文（用于获取web服务器数据）

尝试请求大连工业大学主页报文：

```
1  ---请求行---
2  GET / HTTP/1.1
3  ---请求头---
4  //服务器的主机地址和端口号，默认为80端口
5  Host: www.dlpu.edu.cn
6  //和服务器长期保持连接
7  Connection: keep-alive
8  //升级不安全请求，使用Https协议
9  Upgrade-Insecure-Requests: 1
10 //用户代理
11 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.21 Safari/537.36
12 //可接受数据类型
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
14 //可接受的压缩格式
15 Accept-Encoding: gzip, deflate
16 //可接受的语言
17 Accept-Language: zh-CN,zh;q=0.9
18 //（可选项）登陆用户身份标识
19 Cookie:*****
```

2、POST请求报文（用于向服务器提交数据）

请求教务系统报文：

```
1  ---请求行---
2  POST /jsxsd/xk/LoginToXk HTTP/1.1
```

```
3 ---请求头---
4 //服务器主机地址和端口号, 默认80
5 Host: 210.30.62.37:8080
6 //服务器保持长期链接
7 Connection: keep-alive
8 Content-Length: 43
9 Cache-Control: max-age=0
10 Upgrade-Insecure-Requests: 1
11 Origin: http://210.30.62.37:8080
12 //告知服务器请求的数据类型
13 Content-Type: application/x-www-form-urlencoded
14 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4431.24 Safari/537.36
15 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
16 Referer: http://210.30.62.37:8080/jxsxd/
17 Accept-Encoding: gzip, deflate
18 Accept-Language: zh-CN,zh;q=0.9
19 Cookie: JSESSIONID=CC1724A7F34CC03925E1A3FD675BB343
```

3-3 HTTP响应报文

HTTP响应报文由响应行、响应头、空行和响应体4个部分组成

```
1 ---响应行---
2 //响应行由三部分组成:
3 //HTTP协议版本 状态码 (最常见为200) 状态描述
4 HTTP/1.1 200 OK
5 ---响应头---
6 Bdpagetype: 1
7 Bdqid: 0x8db3db9600017d37
8 Cache-Control: private
9 //和客户端保持长连接
10 Connection: keep-alive
11 Content-Encoding: gzip
12 //内容类型
13 Content-Type: text/html; charset=utf-8
14 //服务器响应时间
15 Date: Fri, 19 Nov 2021 06:35:10 GMT
16 Expires: Fri, 19 Nov 2021 06:34:27 GMT
17 //服务器名称
18 Server: BWS/1.1
19 Set-Cookie: BDSVRTM=0; path=/
20 Set-Cookie: BD_HOME=1; path=/
21 Set-Cookie: H_PS_PSSID=34446_35105_31660_35240_35049_35097_34584_34505_35233_34579_34606_263!
22 Strict-Transport-Security: max-age=172800
23 Traceid: 1637303710273723469810210746217470721335
24 X-Frame-Options: sameorigin
25 X-UA-Compatible: IE=Edge,chrome=1
26 //发送给客户端内容不确定内容长度
27 Transfer-Encoding: chunked
```

HTTP状态码

用于表示web服务器响应状态的3位数字代码

200: 请求成功
307: 重定向
400: 错误的请求, 请求地址或者参数错误
404: 请求资源在服务器不存在
500: 服务器内部源代码出现错误

3-4 搭建Python自带静态Web服务器

```
1 liangyulong@yulongMacbook ~ % python3 -m http.server 9000
2 Serving HTTP on :: port 9000 (http://[::]:9000/) ...
```

1、返回固定页面数据

```
1 import socket
2 if __name__ == '__main__':
3     tcp_server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4     tcp_server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, True)
5     tcp_server_socket.bind(("", 9000))
6     tcp_server_socket.listen(128)
7     while True:
8         new_socket, ip_port = tcp_server_socket.accept()
9         recv_client_data = new_socket.recv(4096)
10        print(recv_client_data.decode("utf-8"))
11        with open("index.html", "rb") as file:
12            file_data = file.read()
13            response_line = "HTTP/1.1 200 OK\r\n"
14            response_header = "Server:PWS1.0\r\n"
15            response_body = file_data
16            response_data = (response_line + response_header + "\r\n").encode("utf-8") + response_body
17            new_socket.send(response_data)
18            new_socket.close()
```

2、返回指定页面数据

实现步骤:

- 1、获取用户请求资源路径
- 2、根据请求资源路径, 读取指定文件的数据
- 3、组装指定文件数据的响应报文, 发送给浏览器
- 4、若文件在服务器不存在, 组装404状态的响应报文, 发送给浏览器。

```
1 import socket
2 if __name__ == '__main__':
3     tcp_service_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4     tcp_service_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, True)
5     tcp_service_socket.bind(("", 9000))
6     tcp_service_socket.listen(128)
7     while True:
8         new_socket, ip_port = tcp_service_socket.accept()
9         recv_client_data = new_socket.recv(4096)
10        if len(recv_client_data) == 0:
11            print("客户端连接失败")
12            new_socket.close()
13            break
14        recv_client_context = recv_client_data.decode("utf-8")
```

```

15     request_list = recv_client_context.split(" ",maxsplit=2)
16     request_path = request_list[1]
17     print(request_path)
18     if request_path == "/":
19         request_path = "index.html"
20     else:
21         request_path_list = request_path.split("/",maxsplit=1)
22         request_path = request_path_list[1]
23     try:
24         with open(request_path,"rb") as file:
25             file_data = file.read()
26     except Exception as e:
27         response_line = "HTTP/1.1 404 Not Found\r\n"
28         response_header = "Server:PWS1.0\r\n"
29         with open("error.html","rb") as file:
30             file_data = file.read()
31         response_body = file_data
32         response_data = (response_line + response_header + "\r\n").encode("utf-8") + response_body
33         new_socket.send(response_data)
34     else:
35         response_line = "HTTP/1.1 200 OK\r\n"
36         response_header = "Server:PWS1.0\r\n"
37         response_body = file_data
38         response_data = (response_line+response_header+"\r\n").encode("utf-8")+response_body
39         new_socket.send(response_data)
40     finally:
41         new_socket.close()

```

3、多线程实现返回指定界面

```

1 import socket
2 import threading
3
4 def thread_for_server(new_socket):
5     recv_client_data = new_socket.recv(4096)
6     if len(recv_client_data) == 0:
7         print("客户端连接失败")
8         new_socket.close()
9         return
10    recv_client_context = recv_client_data.decode("utf-8")
11    request_list = recv_client_context.split(" ", maxsplit=2)
12    request_path = request_list[1]
13    if request_path == "/":
14        request_path = "index.html"
15    else:
16        request_path_list = request_path.split("/", maxsplit=1)
17        request_path = request_path_list[1]
18    try:
19        with open(request_path, "rb") as file:
20            file_data = file.read()
21    except Exception as e:

```

```

22     response_line = "HTTP/1.1 404 Not Found\r\n"
23     response_header = "Server:PWS1.0\r\n"
24     with open("error.html", "rb") as file:
25         file_data = file.read()
26     response_body = file_data
27     response_data = (response_line + response_header + "\r\n").encode("utf-8") + response_body
28     new_socket.send(response_data)
29 else:
30     response_line = "HTTP/1.1 200 OK\r\n"
31     response_header = "Server:PWS1.0\r\n"
32     response_body = file_data
33     response_data = (response_line + response_header + "\r\n").encode("utf-8") + response_body
34     new_socket.send(response_data)
35 finally:
36     new_socket.close()
37
38
39
40 def main():
41     tcp_service_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
42     tcp_service_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, True)
43     tcp_service_socket.bind(("", 9000))
44     tcp_service_socket.listen(128)
45     while True:
46         new_socket, ip_port = tcp_service_socket.accept()
47         print(ip_port)
48         sub_thread = threading.Thread(target=thread_for_server, args=(new_socket,))
49         sub_thread.setDaemon(True)
50         sub_thread.start()
51
52
53 if __name__ == '__main__':
54     main()

```

4、面向对象开发

实现步骤：

- 1、把提供服务的Web服务器抽象成一个类（HTTPWebServer）
- 2、提供Web服务器初始化方法，在初始化方法里面创建socket对象
- 3、提供一个开启Web服务器的方法，让Web服务器处理客户端请求操作

示例代码：

```

1 import socket
2 import threading
3
4
5 class HttpWebServer():
6     def __init__(self):
7         tcp_server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8         tcp_server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, True)
9         tcp_server_socket.bind(("", 9000))
10        tcp_server_socket.listen(128)
11        self.tcp_server_socket = tcp_server_socket

```

```

12
13     #处理客户端的请求
14     @staticmethod
15     def handle_client_request(new_socket):
16         recv_client_data = new_socket.recv(4096)
17         if len(recv_client_data) == 0:
18             print("客户端连接失败")
19             new_socket.close()
20             return
21         recv_client_context = recv_client_data.decode("utf-8")
22         request_list = recv_client_context.split(" ", maxsplit=2)
23         request_path = request_list[1]
24         if request_path == "/":
25             request_path = "index.html"
26         else:
27             request_path_list = request_path.split("/", maxsplit=1)
28             request_path = request_path_list[1]
29         try:
30             with open(request_path, "rb") as file:
31                 file_data = file.read()
32         except Exception as e:
33             response_line = "HTTP/1.1 404 Not Found\r\n"
34             response_header = "Server:PWS1.0\r\n"
35             with open("error.html", "rb") as file:
36                 file_data = file.read()
37             response_body = file_data
38             response_data = (response_line + response_header + "\r\n").encode("utf-8") + response_body
39             new_socket.send(response_data)
40         else:
41             response_line = "HTTP/1.1 200 OK\r\n"
42             response_header = "Server:PWS1.0\r\n"
43             response_body = file_data
44             response_data = (response_line + response_header + "\r\n").encode("utf-8") + response_body
45             new_socket.send(response_data)
46         finally:
47             new_socket.close()
48
49     def start(self):
50         while True:
51             new_socket, ip_port = self.tcp_server_socket.accept()
52             print(ip_port)
53             sub_thread = threading.Thread(target=self.handle_client_request, args=(new_socket,))
54             sub_thread.setDaemon(True)
55             sub_thread.start()
56
57 if __name__ == '__main__':
58     web_server = HttpWebServer()
59     web_server.start()

```

