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：

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

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

```
---请求头---
//服务器主机地址和端口号，默认80
Host: 210.30.62.37:8080
//服务器保持长期链接
Connection: keep-alive
Content-Length: 43
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://210.30.62.37:8080
//告知服务器请求的数据类型
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
Referer: http://210.30.62.37:8080/jsxsd/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
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

```
---响应行---
//响应行由三部分组成:
//HTTP协议版本 状态码（最常见为200） 状态描述
HTTP/1.1 200 OK
---响应头---
Bdpagetype: 1
Bdqid: 0x8db3db9600017d37
Cache-Control: private
//和客户端保持长连接
Connection: keep-alive
Content-Encoding: gzip
//内容类型
Content-Type: text/html;charset=utf-8
//服务器响应时间
Date: Fri, 19 Nov 2021 06:35:10 GMT
Expires: Fri, 19 Nov 2021 06:34:27 GMT
//服务器名称
Server: BWS/1.1
Set-Cookie: BDSVRTM=0; path=/
Set-Cookie: BD_HOME=1; path=/
Set-Cookie: H_PS_PSSID=34446_35105_31660_35240_35049_35097_34584_34505_35233_34579_34606_263
Strict-Transport-Security: max-age=172800
Traceid: 163730371027372346981021074621747072133 5
X-Frame-Options: sameorigin
X-Ua-Compatible: IE=Edge,chrome=1
//发送给客户端内容不确定内容长度
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

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

1、返回固定页面数据

```
import socket
if __name__ == '__main__':
    tcp_server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    tcp_server_socket.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,True)
    tcp_server_socket.bind(("",9000))
    tcp_server_socket.listen(128)
    while True:
        new_socket,ip_port = tcp_server_socket.accept()
        recv_client_data = new_socket.recv(4096)
         print(recv_client_data.decode("utf-8"))
         with open("index.html","rb") as file:
             file_data = file.read()
         response_line = "HTTP/1.1 200 OK\r\n"
         response_header = "Server:PWS1.0\r\n"
         response_body = file_data
         response_data = (response_line + response_header + "\r\n").encode("utf-8")+response_
         new_socket.send(response_data)
         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。

```
import socket
if __name__ == '__main__':
    tcp_service_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    tcp_service_socket.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,True)
    tcp_service_socket.bind(("",9000))
    tcp_service_socket.listen(128)
    while True:
        new_socket,ip_port = tcp_service_socket.accept()
        recv_client_data = new_socket.recv(4096)
         if len(recv_client_data) == 0:
             print("客户端连接失败")
             new_socket.close()
             break
         recv_client_context = recv_client_data.decode("utf-8")
```

```
        request_list = recv_client_context.split(" ",maxsplit=2)
        request_path = request_list[1]
        print(request_path)
        if request_path == "/":
            request_path = "index.html"
        else:
            request_path_list = request_path.split("/",maxsplit=1)
            request_path = request_path_list[1]
        try:
            with open(request_path,"rb") as file:
                file_data = file.read()
        except Exception as e:
            responce_line = "HTTP/1.1 404 Not Found\r\n"
            responce_header = "Server:PWS1.0\r\n"
            with open("error.html","rb") as file:
                file_data = file.read()
            responce_body = file_data
            responce_data = (responce_line + responce_header +"\r\n").encode("utf-8") + resp
            new_socket.send(responce_data)
        else:
            responce_line = "HTTP/1.1 200 OK\r\n"
            responce_header = "Server:PWS1.0\r\n"
            responce_body = file_data
            responce_data = (responce_line+responce_header+"\r\n").encode("utf-8")+responce_
            new_socket.send(responce_data)
        finally:
            new_socket.close()
```

3、多线程实现返回指定界面

```
import socket
import threading

def thread_for_server(new_socket):
    recv_client_data = new_socket.recv(4096)
    if len(recv_client_data) == 0:
        print("客户端连接失败")
        new_socket.close()
        return
     recv_client_context = recv_client_data.decode("utf-8")
     request_list = recv_client_context.split(" ", maxsplit=2)
     request_path = request_list[1]
     if request_path == "/":
         request_path = "index.html"
     else:
         request_path_list = request_path.split("/", maxsplit=1)
         request_path = request_path_list[1]
     try:
         with open(request_path, "rb") as file:
             file_data = file.read()
     except Exception as e:
```

```
        responce_line = "HTTP/1.1 404 Not Found\r\n"
        responce_header = "Server:PWS1.0\r\n"
        with open("error.html", "rb") as file:
            file_data = file.read()
        responce_body = file_data
        responce_data = (responce_line + responce_header + "\r\n").encode("utf-8") + responc
        new_socket.send(responce_data)
    else:
        responce_line = "HTTP/1.1 200 OK\r\n"
        responce_header = "Server:PWS1.0\r\n"
        responce_body = file_data
        responce_data = (responce_line + responce_header + "\r\n").encode("utf-8") + responc
        new_socket.send(responce_data)
    finally:
        new_socket.close()


def main():
    tcp_service_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    tcp_service_socket.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,True)
    tcp_service_socket.bind(("",9000))
    tcp_service_socket.listen(128)
    while True:
        new_socket,ip_port = tcp_service_socket.accept()
        print(ip_port)
        sub_thread = threading.Thread(target=thread_for_server, args=(new_socket,))
        sub_thread.setDaemon(True)
        sub_thread.start()


if __name__ == '__main__':
    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：

```
import socket
import threading


class HttpWebServer():
    def __init__(self):
        tcp_server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        tcp_server_socket.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,True)
        tcp_server_socket.bind(("",9000))
        tcp_server_socket.listen(128)
        self.tcp_server_socket = tcp_server_socket
```

```python

    #处理客户端的请求
    @staticmethod
    def handle_client_request(new_socket):
        recv_client_data = new_socket.recv(4096)
        if len(recv_client_data) == 0:
            print("客户端连接失败")
            new_socket.close()
            return
        recv_client_context = recv_client_data.decode("utf-8")
        request_list = recv_client_context.split(" ", maxsplit=2)
        request_path = request_list[1]
        if request_path == "/":
            request_path = "index.html"
        else:
            request_path_list = request_path.split("/", maxsplit=1)
            request_path = request_path_list[1]
        try:
            with open(request_path, "rb") as file:
                file_data = file.read()
        except Exception as e:
            responce_line = "HTTP/1.1 404 Not Found\r\n"
            responce_header = "Server:PWS1.0\r\n"
            with open("error.html", "rb") as file:
                file_data = file.read()
            responce_body = file_data
            responce_data = (responce_line + responce_header + "\r\n").encode("utf-8") + resp
            new_socket.send(responce_data)
        else:
            responce_line = "HTTP/1.1 200 OK\r\n"
            responce_header = "Server:PWS1.0\r\n"
            responce_body = file_data
            responce_data = (responce_line + responce_header + "\r\n").encode("utf-8") + resp
            new_socket.send(responce_data)
        finally:
            new_socket.close()

    def start(self):
        while True:
            new_socket, ip_port = self.tcp_server_socket.accept()
            print(ip_port)
            sub_thread = threading.Thread(target=self.handle_client_request, args=(new_socke
            sub_thread.setDaemon(True)
            sub_thread.start()

if __name__ == '__main__':
    web_server = HttpWebServer()
    web_server.start()
```