專題名稱：[簡易 HTTP 伺服器-從零打造 HTTP/1.1]

作者: [柯昱如] - [11303122A]、 [蘇聖恩] - [11303121A]、[侯芊妤] - [11303098A] 、[包耀文] - [11303804A]

專題簡介:

本專題從 TCP Socket 層開始，實作一個能夠正確處理 HTTP/1.1 Request/Response 的簡易 Web 伺服器。透過 Python Socket 程式設計，實作一個簡易的 HTTP Client 與 HTTP Server，以實際程式操作的方式理解 Client－Server 架構與 HTTP 通訊流程。專題中不使用現成的網頁框架，而是從 TCP Socket 底層開始建立連線，讓我們能更清楚了解資料在網路中傳輸的過程。

Server 端負責監聽指定的埠號，接收 Client 傳送的 HTTP Request，並依照請求的方法與路徑（如 GET、POST）回傳對應的 HTTP Response；Client 端則主動建立連線，手動組合 HTTP 請求封包，向 Server 發送請求並接收回應內容。透過此方式，可以清楚觀察 HTTP Header 與 Body 的結構，以及請求與回應的完整流程。

本專題示範了 HTTP 協定的基本運作原理，包含請求格式、回應格式與資料傳輸方式，幫助我們將課堂中所學的網路概念實際應用於程式實作，並加深對 Client－Server 通訊與網路協定的理解。

系統支援

- 支援基本 HTTP/1.1 GET、POST
- 支援靜態 HTML 與圖片回傳
- 使用 Content-Length 回傳資料
- 使用多執行緒處理連線

此專題能讓開發者深入理解 HTTP/1.1 底層運作方式，並能透過瀏覽器或 curl 進行測試與展示。

功能特色:

- ✅ Server：接受連線並回傳 HTML 檔案
- ✅ Client：向 Server 送出 GET 請求
- ✅ 支援簡單 HTTP/1.1 格式
- ✅ 可查看瀏覽器、客戶端回應

Client (瀏覽器/curl)
↓
(TCP 連線) Server (Socket 監聽)
↓
處理執行緒 (Thread)
↓
解析 HTTP Request (GET / HEAD)

↓

檔案查找與 MIME 判斷

↓

構造 HTTP Response (含 Content-Length / Chunked)

↓

回傳 Response (200 OK / 404 Not Found)

採用多執行緒（Thread）設計，是為了讓 Server 能同時處理多個 Client 連線，避免單一請求造成整個 Server 阻塞。HTTP Request 的解析採用字串處理方式，目的是直接理解 HTTP 封包的實際文字格式，而非仰賴函式庫。

[說明您的協定設計]

範例

◈Request 格式（你支援的）

GET /index.html HTTP/1.1
Host: 127.0.0.1
Connection: keep-alive

◈Response 格式（你伺服器產生的）

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 140
Connection: close

◈如果 404 你怎麼回

HTTP/1.1 404 Not Found
Content-Length: 0

安裝與執行

需求

- Python 3.7+
- 新增 www/index.html

安裝

1. 先啟動 server：
2. 再執行 client：
3. 或者在瀏覽器輸入：

http://127.0.0.1:8080

測試結果:

[server]

```
import socket
import threading
import base64

HOST = "0.0.0.0" #要與別台電腦連線必須使用 0.0.0.0，否則則用 127.0.0.1
PORT = 8000

# 一張很小的紅點 PNG（base64，內建圖片）
IMAGE_BASE64 = (
    "iVBORw0KGgoAAAANSUhEUgAAAAoAAAAKCAYAAACNMs+9AAAAF0lEQVQoU2NkYGD4"
    "//8/AxjgYGBgAAAABQABDQottAAAAABJRU5ErkJggg=="
)

IMAGE_DATA = base64.b64decode(IMAGE_BASE64)

def handle_client(conn, addr):
    print("Client connected:", addr)

    request = conn.recv(4096)
    if not request:
        conn.close()
        return

    request_text = request.decode(errors="ignore")
    lines = request_text.split("\r\n")
    request_line = lines[0]
    print("Request Line:", request_line)

    method, path, _ = request_line.split()

    # ===== GET / 顯示 HTML =====
    if method == "GET" and path == "/":

        html = f"""
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>HTTP Server Demo</title>
</head>
<body>
```

```
    <h1>HTTP Server Course Project</h1>

    <p>This server supports:</p>
    <ul>
        <li>GET / POST</li>
        <li>Multi-threading</li>
        <li>Image (base64 embedded)</li>
    </ul>

    <h2>POST Test</h2>
    <form method="POST" action="/">
        <input type="text" name="message" placeholder="Type something">
        <button type="submit">Send</button>
    </form>

    <h2>Image Test</h2>
    <img src="/image" alt="demo image">

    <hr>
    <p>Client: {addr}</p>
</body>
</html>
"""
        body = html.encode("utf-8")

        header = (
            "HTTP/1.1 200 OK\r\n"
            "Content-Type: text/html; charset=UTF-8\r\n"
            f"Content-Length: {len(body)}\r\n"
            "\r\n"
        )

        conn.sendall(header.encode() + body)

    # ===== GET /image 顯示圖片 =====
    elif method == "GET" and path == "/image":

        header = (
            "HTTP/1.1 200 OK\r\n"
            "Content-Type: image/png\r\n"
            f"Content-Length: {len(IMAGE_DATA)}\r\n"
            "\r\n"
```

```
        )

        conn.sendall(header.encode() + IMAGE_DATA)

    # ===== POST / =====
    elif method == "POST":

        body_data = request_text.split("\r\n\r\n", 1)[1]

        response = f"""
<html>
<body>
    <h1>POST Data Received</h1>
    <p>{body_data}</p>
    <a href="/">Back</a>
</body>
</html>
"""
        body = response.encode("utf-8")

        header = (
            "HTTP/1.1 200 OK\r\n"
            "Content-Type: text/html; charset=UTF-8\r\n"
            f"Content-Length: {len(body)}\r\n"
            "\r\n"
        )

        conn.sendall(header.encode() + body)

    # ===== 不支援的方法 =====
    else:
        msg = "404 Not Found".encode()
        header = (
            "HTTP/1.1 404 Not Found\r\n"
            f"Content-Length: {len(msg)}\r\n"
            "\r\n"
        )
        conn.sendall(header.encode() + msg)

    conn.close()

# ===== 主程式（多執行緒） =====
```

```python
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((HOST, PORT))
server.listen(10)

print(f"Server running at http://{HOST}:{PORT}")

while True:
    conn, addr = server.accept()
    thread = threading.Thread(target=handle_client, args=(conn, addr))
    thread.start()
```

[sever 結果]

[client]

```python
import socket

HOST = "192.168.250.239" #與別台電腦連線需要改電腦的 ip 位址，否則使用 127.0.0.1
PORT = 8000

def send_request(request):
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((HOST, PORT))
    client.sendall(request.encode())

    response = b""
    while True:
        data = client.recv(4096)
```

```
        if not data:
            break
        response += data

    client.close()
    return response

# ===== 1. GET / 測試 =====
print("===== GET / =====")

get_request = (
    "GET / HTTP/1.1\r\n"
    "Host: localhost\r\n"
    "\r\n"
)

response = send_request(get_request)
print(response.decode(errors="ignore"))

# ===== 2. POST / 測試 =====
print("\n===== POST / =====")

post_data = "message=HelloServer"

post_request = (
    "POST / HTTP/1.1\r\n"
    "Host: localhost\r\n"
    "Content-Type: application/x-www-form-urlencoded\r\n"
    f"Content-Length: {len(post_data)}\r\n"
    "\r\n"
    + post_data
)

response = send_request(post_request)
print(response.decode(errors="ignore"))

# ===== 3. GET /image 測試 =====
print("\n===== GET /image =====")

image_request = (
"GET /image HTTP/1.1\r\n"
"Host: localhost\r\n"
```

```
"\r\n"
)
response = send_request(image_request)
# 只印出 header（避免圖片亂碼）
header, _, body = response.partition(b"\r\n\r\n")
print(header.decode(errors="ignore"))
print(f"[Image data received: {len(body)} bytes]")
```

[client 結果]

```
===== POST / =====
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 119


<html>
<body>
    <h1>POST Data Received</h1>
    <p>message=HelloServer</p>
    <a href="/">Back</a>
</body>
</html>


===== GET /image =====
HTTP/1.1 200 OK
Content-Type: image/png
Content-Length: 79
[Image data received: 79 bytes]
PS D:\py>
```

未來改進方向

- 支援更多 HTTP Keep-Alive 機制
- 支援更多 HTTP Method 與狀態碼
- 改善 Request 解析和錯誤處理機制
- 效能與架構優化
- 安全性與存取控制強化

參考資料

- Python socket 官方文件
- MDN HTTP/1.1 說明