Chapter 5：樹莓派 × 影像串流

本章重點

介紹樹莓派的串流技術。

準備材料

可上網的電腦

樹莓派（含相機模組與外殼）

學習目標

1. 架設 RTSP 串流伺服器。
2. 架設網頁串流伺服器。
3. 執行 OpenCV 專案。

5-1. 架設 RTSP 串流伺服器

串流 (Streaming)：從伺服器端傳送壓縮媒體資料至客戶端來及時觀看影像，在 Web 客戶端的使用者不需要花費時間等待下載整個檔案，就能即時播放視訊內容。

架設 RTSP 串流伺服器

執行樹莓派內建的 RTSP Server

樹莓派已內建 RTSP，輸入：

```
~$ raspivid -o - -t 0 -hf -w 320 -h 240 -fps 15 | cvlc -vvv
stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554}' :demux=h264
```

※ 8554 代表設定哪一個 port 給 RTSP。

若需要旋轉鏡頭，請額外加入旋轉的參數：

```
~$ raspivid -o - -t 0 -hf --rotation 180 -w 320 -h 240 -fps 15 | cvlc -
vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554}' :demux=h264
```

常見問題 若無法執行，檢查該欲設定的 port 有無被占用。

找出佔用的 Port Number 的 PID (Process ID)：

```
~$ sudo netstat -lpn |grep 8554
```

刪除佔用的 PID：

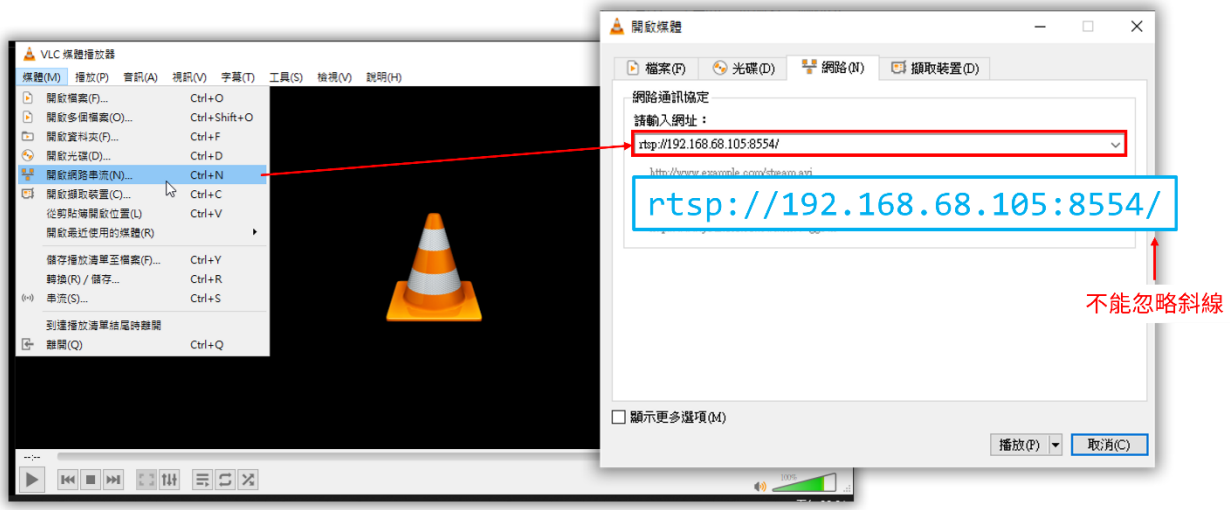
```
~$ sudo kill -9 你的 PID
```

RTSP Client for Windows

Step 1 下載 Windows 的 VLC 播放器。

官網 https://get.videolan.org

Step 2 輸入以下資料：`rtsp://你的樹莓派 IP:8554/`，然後按下『 播放 』。



如何降低延遲？

安裝輕量級的 RTSP Server 可降低延遲。

Step 1 安裝輕量級的 RTSP Server。

回到家目錄：

```
~$ cd ~
```

下載 h264_v4l2_rtspserver：

```
~$ git clone https://github.com/mpromonet/h264_v4l2_rtspserver.git
```

cmake：

```
~$ sudo apt-get install liblivemedia-dev libv4l-dev cmake
```

進入 h264_v4l2_rtspserver 資料夾：

```
~/h264_v4l2_rtspserver$ cd h264_v4l2_rtspserver
```

cmake：

```
~/h264_v4l2_rtspserver$ cmake .
```

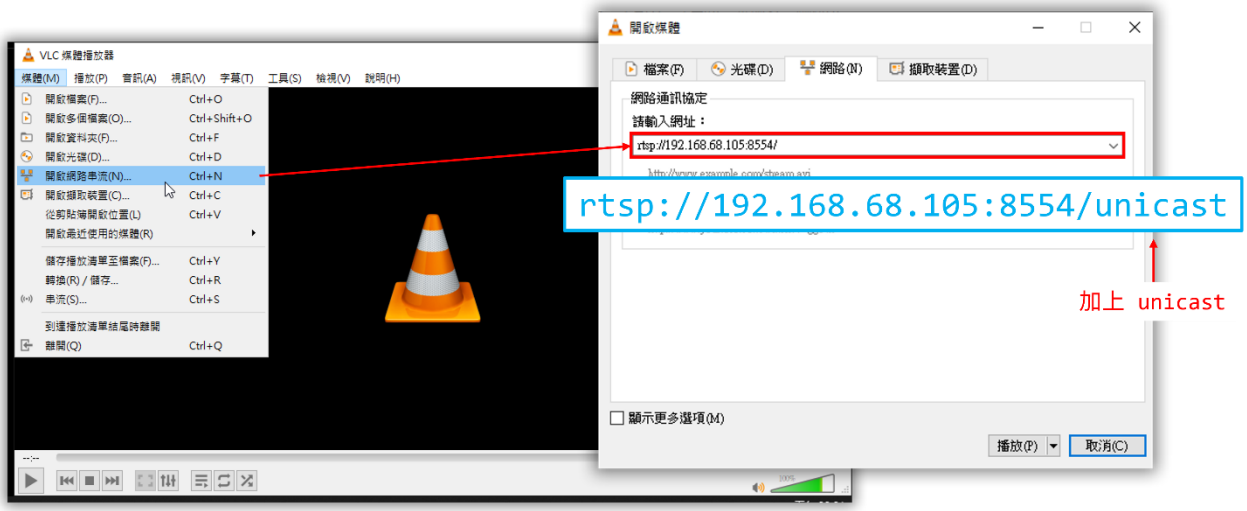
編譯，並使用樹莓派的 4 核 CPU 來執行：

```
~/h264_v4l2_rtspserver$ make -j4
```

執行 v4l2rtspserver：

```
~/h264_v4l2_rtspserver$ sudo ./v4l2rtspserver -F 15 -W 800 -H 600 -P 8554
/dev/video0
```

Step 2 輸入以下資料：`rtsp://你的樹莓派 IP：8554/unicast`，然後按下『 播放 』。



RTSP Client for Android/iOS

請自行操作。

5-2. 架設網頁串流伺服器

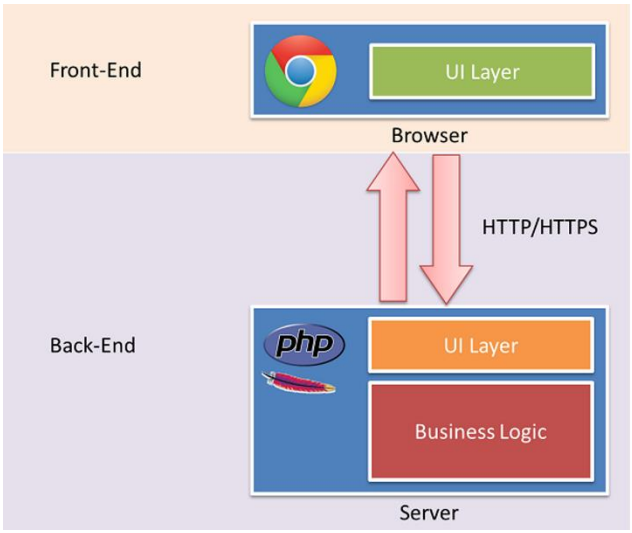
串流 (Streaming)：從伺服器端傳送壓縮媒體資料至客戶端來及時觀看影像，在 Web 客戶端的使用者不需要花費時間等待下載整個檔案，就能即時播放視訊內容。

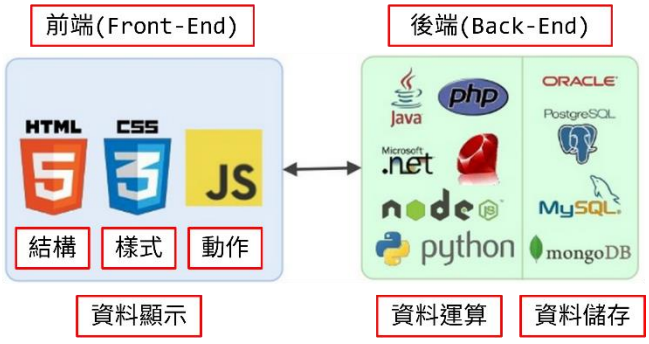
用 HTTP 做 Streaming

使用 HTTP + MJPG

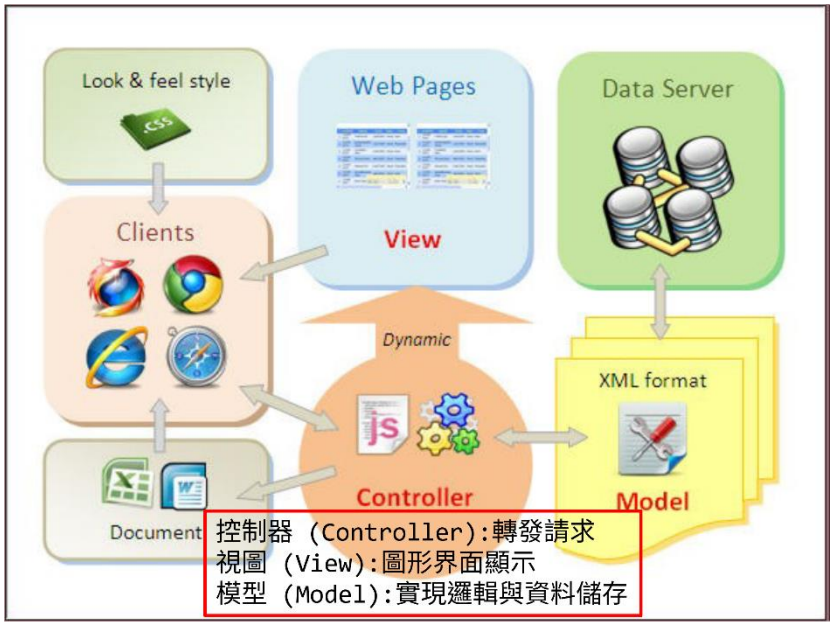
原理：向 Web Server 請求一個很大的檔案，該檔案是一個即時的資料。

網頁的運作與實作原理：



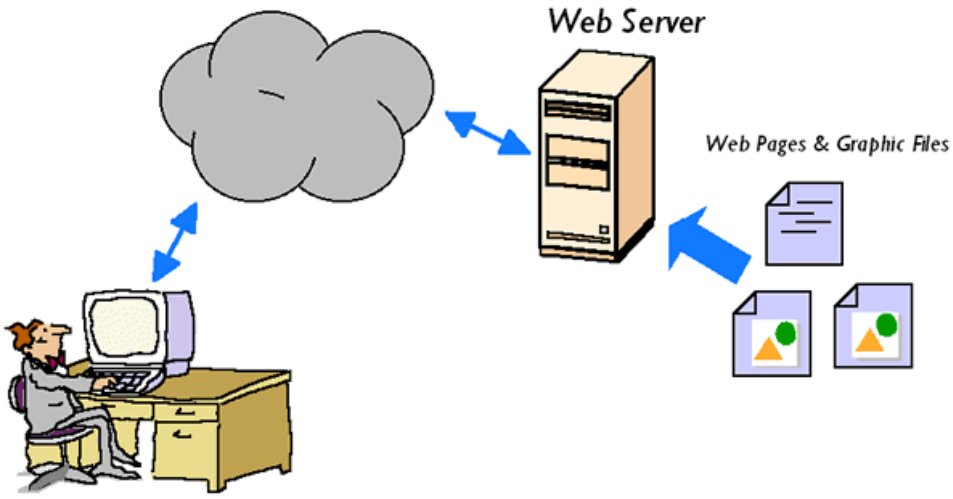


Model-View-Controller 設計模式：



網頁伺服器 (Web Server)：是一個軟體。

- 回應從 80/8080 port 進來的 HTTP 要求。
- 可透過 CGI 或 module 方式擴充。
- 如 Apache, Nginx, Boa。



Python Microframework (網站微框架)：Flask

目前網路程式架構很多，例如：Django、Flask、Pyramid、Bottle、Tornado。其中 Flask 是發佈於 2010 年的年輕網路架構，吸收其他框架的優點而建立。

網站微框架 (microframework)，卻兼具核心程式碼簡潔以及強大的擴充能力。除此之外，Flask 也不會替開發者做任何技術決定，不會替您決定該使用哪種資料庫，但提供各種擴充套件協助整合各種不同資料庫。

Flask 的設計理念：Micro

喜歡 Flask 的 Micro 輕量化的設計理念，在架設 Flask 時就像是堆積木，可以自己決定要使用什麼積木 (擴充套件)，不會有多餘的積木，達到簡單、輕量、高擴充性的架構。

Flask，Hello World!

Step 1 安裝 Flask。

```
~$ pip install Flask
```

Step 2 創建 Flask 架構。

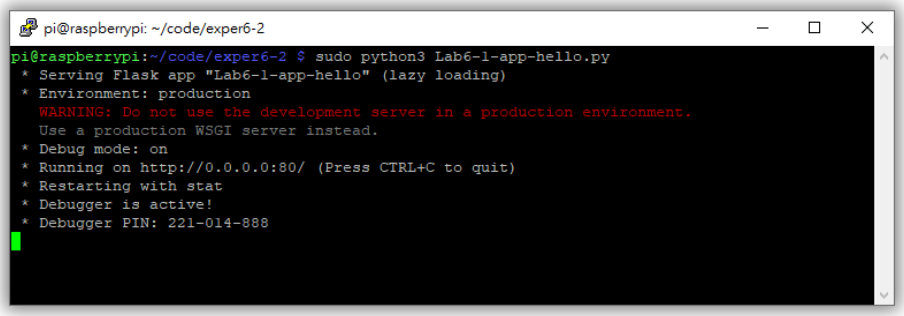
app.py 測試預設的檔名 app.py。

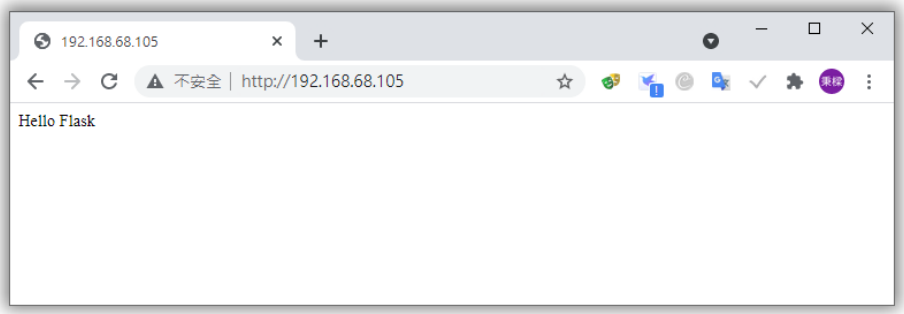
```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello Flask'
```

Step 3 運行 Flask。

```
~$ sudo flask run --reload --debugger --host 0.0.0.0 --port 80
```

輸出結果：





- 需要 sudo 權限。
- 離開只能使用 Crtl + C 正常中斷程序。
- --reload：修改 py 檔後，Flask server 會自動 reload。
- --debugger：如果有錯誤，會在頁面上顯示是哪一行錯誤。
- --host：可以指定允許訪問的主機 IP，0.0.0.0 為所有主機的意思。
- --port：自訂網路埠號的參數。

加入常用的 Python 的 main 方法

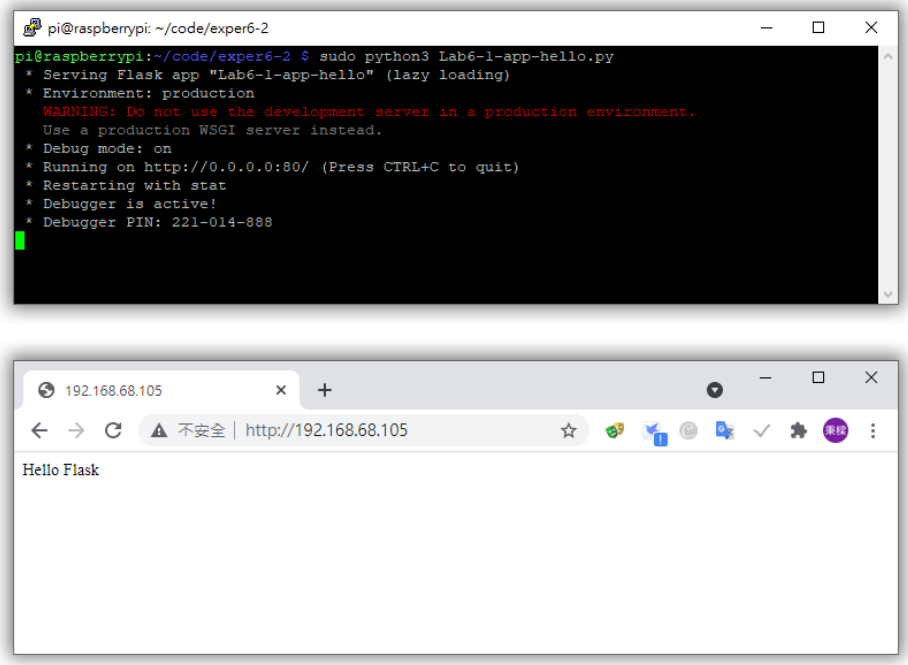
在日常的開發中，可以再加上 python 的 main 方法，執行 app.run() 函式，執行網頁伺服器的啟動動作。

5-1 測試直接運行的訊息。(5-1-app-hello.py)

```
from flask import Flask

app = Flask(__name__)
@app.route('/')
def index():
   return 'Hello Flask'
if __name__ == '__main__':
    app.run(host = '0.0.0.0', port = 80, debug = True)
```

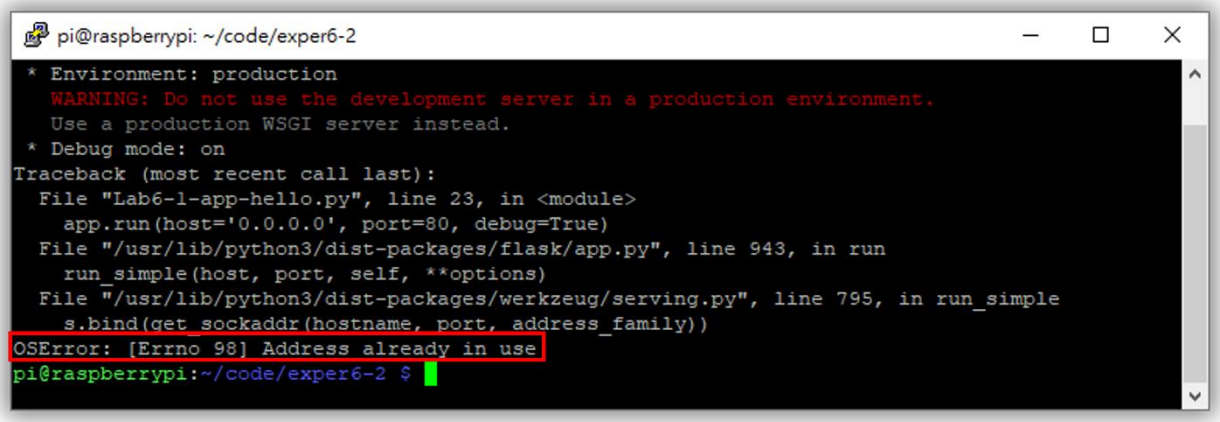
```
~$ sudo python 5-1-app-hello.py
```

輸出結果：



- 上面需要 `sudo python 檔案名稱.py` 執行這個檔案。

常見問題 若發生 Address 被占用，無法執行時。例如：Crtl + Z 不正常中斷程序。



查找哪一個 Python 的程序正在執行中：

```
~$ ps -fA | grep python
```

刪除 root 占用的程序 (需自行找出哪一個 PID)：

```
~$ sudo kill -9 PID
```

執行中的 python 是 root 的程序，所以加 sudo。

樣板引擎 (Template Engine)

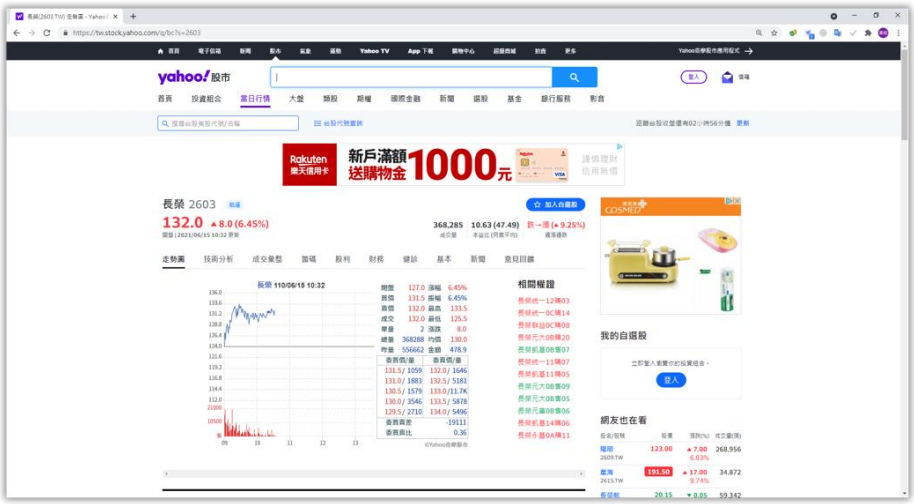
首先介紹靜態與動態網頁。

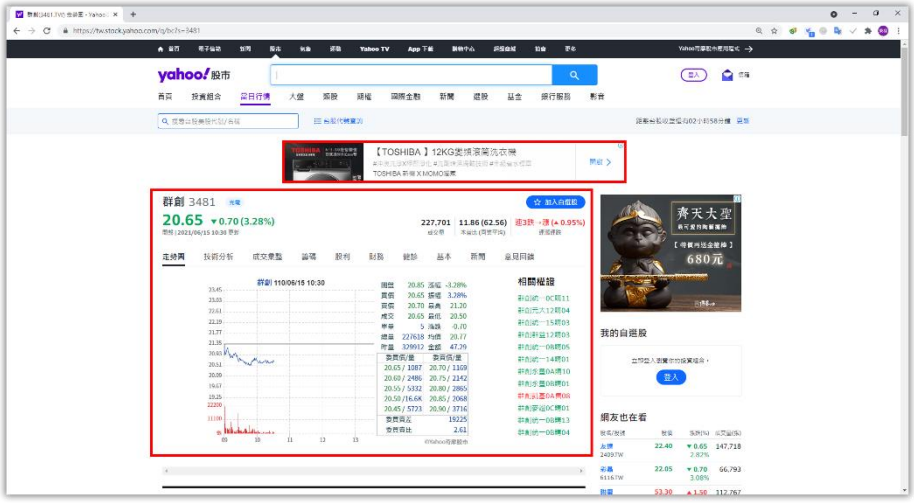
- 靜態網頁：

  超文件標示語言 (HyperText Markup Language，簡稱：HTML)，HTML 是網站建置的基礎技術，常與 CSS 與 Javascript 配合成一個適合觀看的網頁，讓瀏覽器去讀取，一般判斷方式為網頁副檔名為 html 或 htm 皆為靜態網頁，靜態網頁的優勢為容易為搜尋引擎所接受，所以很多動態網頁會將動態網頁轉變成靜態方式，就是所謂的『 偽靜態網頁 』來提高搜尋引擎的友善度達到排名優化的成效。

- 動態網頁：

  動態網頁主要是搭配伺服器與資料庫共同運作，主要是使用大量編譯的地方，如會員功能、購物車、討論區等等...，意思是指可以與網頁做互動編譯的網頁，動態網頁的內容隨著用戶的輸入和互動而有所不同有 Perl、PHP、ASP、JSP、ColdFusion 等編譯方式，從而對動態網頁的內容進行改變。



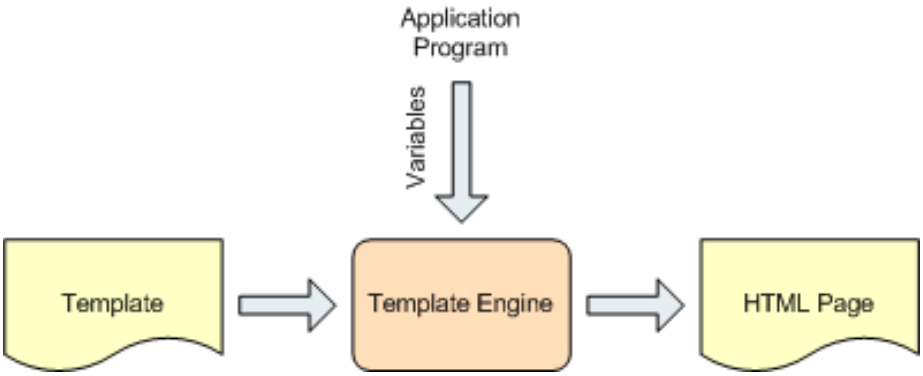


- 靜態網頁與動態的區別：

  靜態網頁與動態網頁是可以同時存在一個網站上的，二種語言各有其優勢，靜態網站主要是用於較於簡單，更新的不頻繁的網頁，反之動態網站較適合用於，資料內容較大，更新快速的網頁，讓維護人員可以更方便管理網站，也大幅降低維護成本。

動態網頁是由靜態 HTML 加上動態文字產生：

- 樣板引擎將產生動態的程式碼與使用者介面分離。
- Flask 預設使用 Jinja 做為樣板引擎。



Step 1 建立 templates。

建立一個資料夾 templates：

```
~$ mkdir templates && cd templates
```

新增一個 link.html：

```
~/templates$ nano link.html
```

編輯 link.html 的網頁碼：

```
<h1>Hello Template</h1>
<a href = "{{ url_for('foo') }}">foo</a>
```

再新增一個 bar.html：

```
~/templates$ nano bar.html
```

編輯 bar.html 的網頁碼：

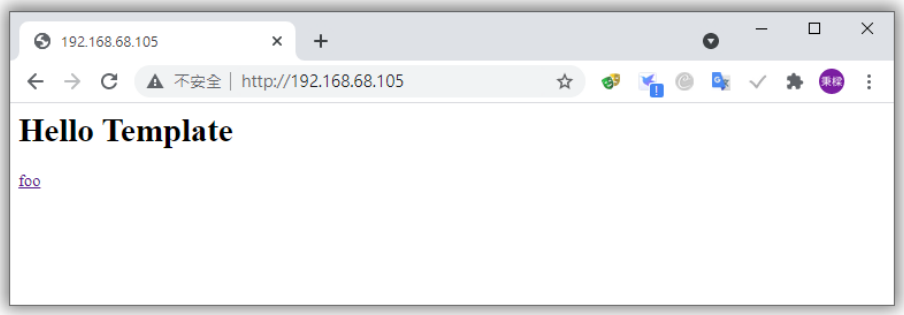
```
<ul>
{% for ext in extns %}
<li>{{ ext }}</li>
{% endfor %}
</ul>
```

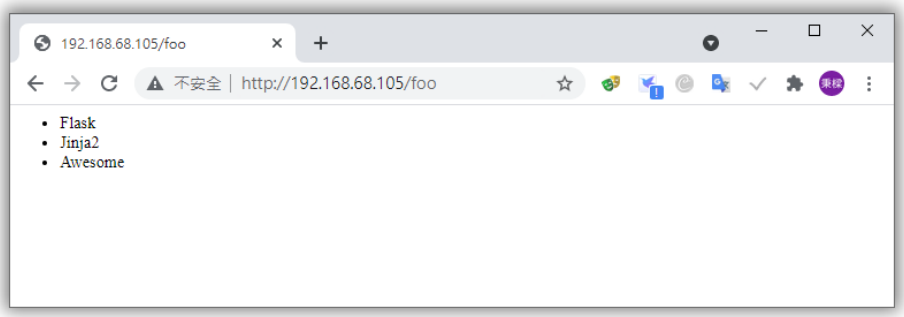
Step 2 建立一個 route。

5-2 route 程式。(5-2-app-route.py)

```
from flask import Flask, render_template, Response

app = Flask(__name__)
@app.route('/')
def index():
    return render_template('link.html')
@app.route('/foo')
def foo():
    extns = ['Flask', 'Jinja2', 'Awesome']
    return render_template('bar.html', extns = extns)
if __name__ == '__main__':
    app.run(host = '0.0.0.0', port = 80, debug = True)
```

輸出結果：





Streaming 圖片

建立 Camera 類別 (模組化)

自訂函式 (function)、類別 (class)、模組 (module)，完成模組化的設計優點。

Step 1 建立一個模組。

自訂類別 自訂一個模組，裡頭只有一個 Camera 類別。(stream_pi.py)

```
from time import time

class Camera(object):
    def __init__(self):
        self.frames = [open('static/' + f + '.jpg', 'rb').read()
            for f in ['1', '2', '3']]
    def get_frame(self):
        return self.frames[int(time()) % 3]
```

Step 2 建立串流。

5-3 串流一連串圖片。(5-3-app-stream.py)

```
from flask import Flask, render_template, Response
from stream_pi import Camera

app = Flask(__name__)
@app.route('/')
def index():
    return render_template('stream.html')
def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
@app.route('/video_feed')
def video_feed():
    return Response(gen(Camera()),
                    mimetype = 'multipart/x-mixed-replace; boundary=frame')
if __name__ == '__main__':
```

```
    app.run(host = '0.0.0.0', port = 80, debug = True)
```

Step 3 建立一個 template。

新增一個 stream.html：

```
~/templates$ nano stream.html
```

編輯 stream.html 的網頁碼：

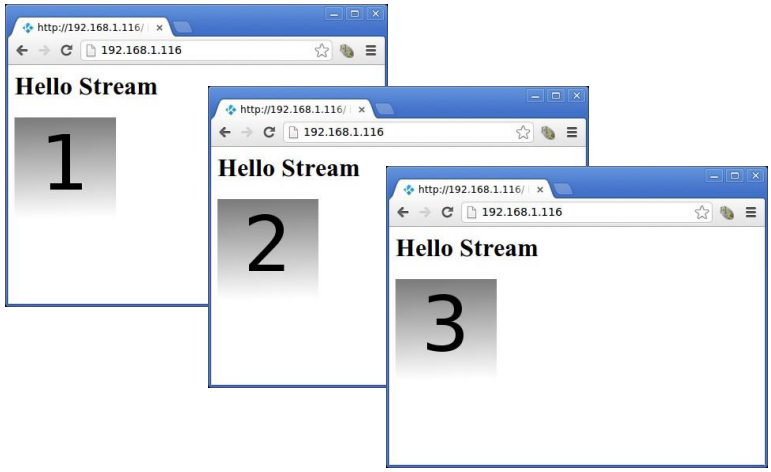
```
<h1>Hello Stream</h1>
<img id = "bg" src = "{{ url_for('video_feed') }}">
```

※ 圖檔已建立好，共有 1.jpg、2.jpg、3.jpg 三個檔。

Step 4 執行 5-3-app-stream.py。

```
~$ sudo python 5-3-app-stream.py
```

輸出結果：



Streaming 即時影像

MJPEG = Motion + JPEG

- 一種視訊壓縮格式。
- 每一個 frame 都使用 JPEG 編碼。
- 對運算能力與記憶體的需求較低。

建立 Camera 類別：從 Camera 讀取影像

自訂函式 (function)、類別 (class)、模組 (module)，完成模組化的設計優點。

Step 1 建立一個模組。

自訂類別 自訂一個模組，Camera 類別用來讀取影像。(camera_pi.py)

```
import cv2

class Camera(object):
    def __init__(self):
        self.video = cv2.VideoCapture(0)

    def __del__(self):
        self.video.release()

    def get_frame(self):
        success, image = self.video.read()
        ret, jpeg = cv2.imencode('.jpg', image)
        return jpeg.tostring()
```

Step 2 建立串流即時影像。

5-4 串流即時影像。(5-4-app-camera.py)

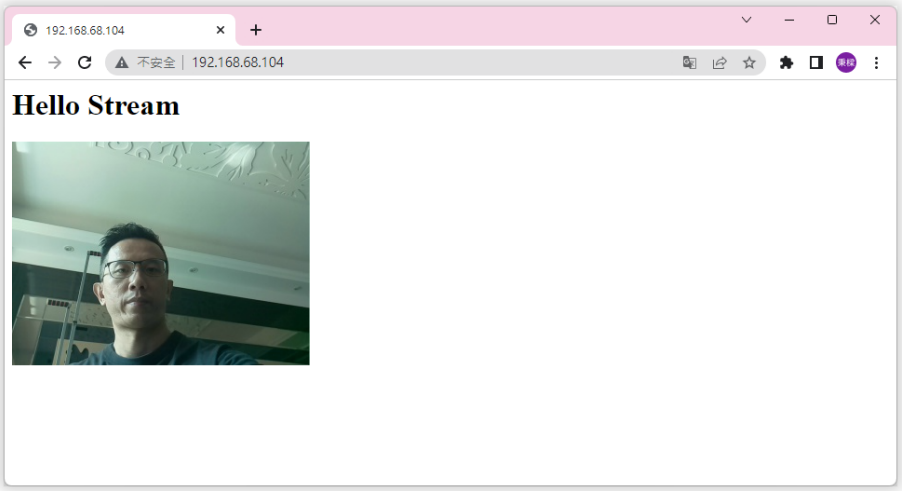
```
from flask import Flask, render_template, Response
from camera_pi import Camera

app = Flask(__name__)
@app.route('/')
def index():
    return render_template('stream.html')
def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
@app.route('/video_feed')
def video_feed():
    return Response(gen(Camera()),
```

```
                mimetype = 'multipart/x-mixed-replace; boundary = frame')
if __name__ == '__main__':
    app.run(host = '0.0.0.0', port = 80, debug = True)
```

Step 3 執行 5-4-app-camera.py。

```
~$ sudo python 5-4-app-camera.py
```

輸出結果：



讓 sudo 正確執行 OpenCV

在 root 的權限下，安裝以下 Python 模組才能正確執行 OpenCV：

```
~$ sudo pip install numpy==1.21.1
~$ sudo pip install opencv-python==4.5.1.48
~$ sudo pip install opencv-contrib-python==4.5.1.48
```

如何降低延遲？

改變像素大小與調整 JPG 的壓縮品質。

自訂類別 改進的 camera_pi.py。

```
import cv2
class Camera(object):
    def __init__(self):
        self.video = cv2.VideoCapture(0)
        self.video.set(PROP_FRAME_WIDTH, 320)
        self.video.set(PROP_FRAME_HEIGHT, 240)
```

```
    def __del__(self):
        self.video.release()

    def get_frame(self):
        success, image = self.video.read()
        ret, jpeg = cv2.imencode('.jpg', image, [1, 50])
        return jpeg.tostring()
```

[1, 50] 表示兩個部分：

1. 1：這是參數的鍵，表示 JPEG 圖像的 『 壓縮質量 』參數 (對應 OpenCV 常量 cv2.IMWRITE_JPEG_QUALITY)。
2. 50：這是參數的值，表示壓縮質量的數值範圍，取值從 0 到 100 (值越高，質量越好，文件越大)。

[1, 50] 的作用是指定 JPEG 圖像的質量為 50%。質量較低時，生成的文件大小更小，但圖像會有更多的壓縮失真。相反，質量更高時，圖像會更清晰，但文件大小也會變大。

結論

本章重點：

1. 串流服務可使用 RTSP + H.264，或是 HTTP + MJPG 等多種組合。
2. 使用自己寫的 HTTP 串流，可以在 JPG 丟出去前再行處理。

5-3. 執行 OpenCV 專案

加入 OpenCV 人臉辨識

進入 project_opencv 的資料夾：

```
~$ cd ~/ch05/project_opencv
```

執行 app-opencv.py：

```
~/project_opencv$ sudo python app-opencv.py
```

物聯網應用：加入 OpenCV 人臉辨識 + LINE 推播

進入 project_opencv 的資料夾：

```
~$ cd ~/ch05/project_opencv+line
```

執行 app-opencv+line.py：

```
~/project_opencv+line$ sudo python app-opencv+line.py
```