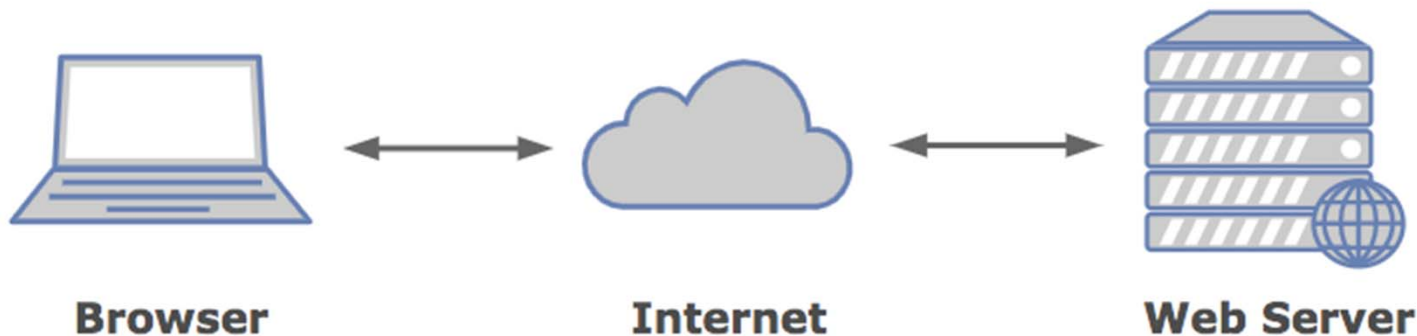


# Flask的基本app結構

# HTTP簡介

# WWW 全球資訊網

- WWW (World Wide Web) 全球資訊網，也常簡稱為 Web。



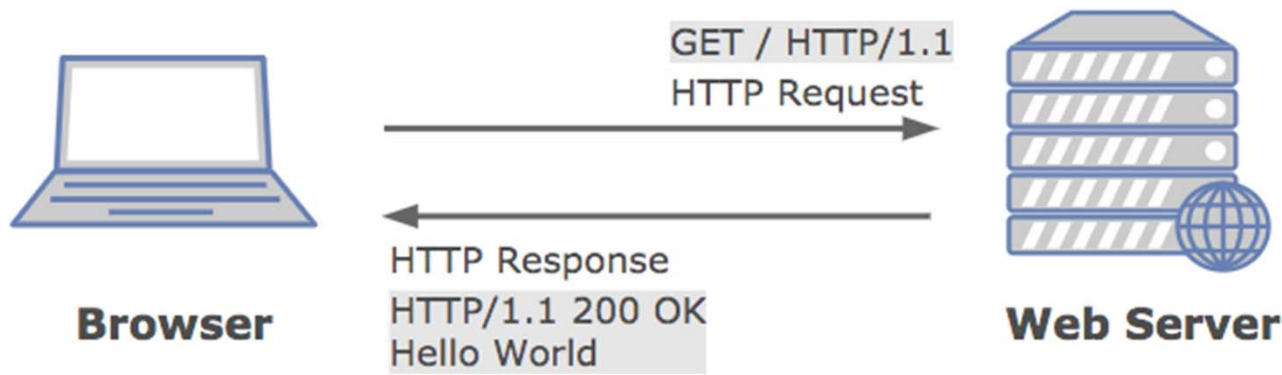
# HTML 基礎認識

- HTML ( Hypertext Markup Language ) 超文字標示語言。
- HTML 是一種標記語言，主要描述一個網站的語意結構，結合 CSS 與 JavaScript 技術可以實現多種網站的應用。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My Web</title>
  </head>
  <body>
    
  </body>
</html>
```

# HTTP 通訊協定

- HTTP : HyperText Transfer Protocol - 超文字傳輸協定
- HTTP 是一種網路傳輸協定，屬於 TCP/IP 協定中的應用層
- HTTP 初期設計為收發 HTML (HyperText Markup Language)
- HTTP 是無狀態的，客戶端與伺服器端的請求間不會儲存任何狀態



# HTTP 狀態碼

- HTTP狀態碼是用以表示 HTTP回應狀態的3位數字代碼
- 由 RFC 2616 規範定義的，並得到 RFC 2518、2817、2295、2774 與 4918 等規範擴充
- HTTP 狀態碼表明一個 HTTP 要求是否已經被完成。回應分為五種：
  - 資訊回應 (Informational responses, 100–199),
  - 成功回應 (Successful responses, 200–299),
  - 重定向 (Redirects, 300–399),
  - 用戶端錯誤 (Client errors, 400–499),
  - 伺服器端錯誤 (Server errors, 500–599).

# URL

- 統一資源定位器 (URL) 是一個在網際網路上查找指定資源(例如網頁，圖片或影片)位置的文字字串，如：
- `https://zh.wikipedia.org/wiki/Wiki`
- `https://www.youtube.com/watch?v=Qr4QMBUPxWo&ab\_channel=freeCodeCamp.org`

# HTTP Request Methods

- HTTP 定義了一組能令給定資源，執行特定操作的請求方法(request methods)
- GET      獲取某一資源
- POST     提交某一資源
- PUT      更新已存在的資源
- DELETE   刪除某一資源
  
- 參考：<https://docs.discourse.org/>



# Hello Flask

- 從一個基本的Flask app 網站來了解如何使用Flask框架

# Hello Flask

■ 範例檔案 app.py 內容 (嘗試使用Visual Studio Code的終端機執行它) :

```
from flask import Flask  
app = Flask(__name__)
```

```
@app.route('/')  
def index():  
    return 'Hello, Flask!'
```

```
if __name__ == "__main__":  
    app.run(host='127.0.0.1', port=5000, debug=True)
```

# Flask app 初始化

- 所有 Flask web app 都必須建立一個 app 實例 (instance) 。 App 實例是 Flask 類別 (class) 的物件 (object) ， 它的建立方式通常是：

```
      模組(module)    類別(class)  
from flask import Flask  
app = Flask(__name__)
```

- Flask類別建構式唯一需要的引數是app的主模組名稱或套件名稱，對多數app而言，這個引數的值是 Python 的 `__name__` 變數

# 特殊 Python 系統變數

■ `__name__` 代表模組名稱，內容有以下兩種可能

- `__main__`：最上層腳本環境

- 一般是代表直接執行該腳本(script)

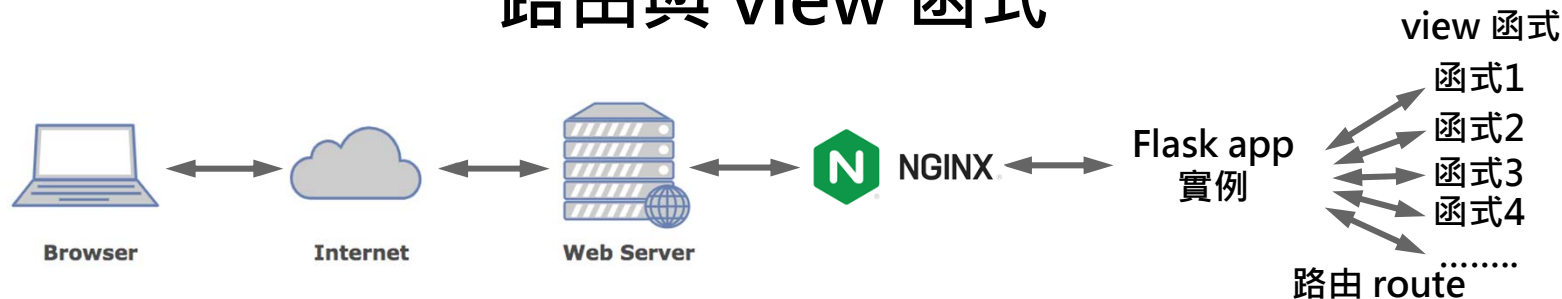
- 模組名稱

- 代表該腳本是被匯入的(import)

■ 當直接執行 Python 程式時，`__name__` 會被自動設定為 `'__main__'`，如此程式可以依此判斷是直接執行還是被其他程式匯入

- ```
if __name__ == '__main__':  
    # execute only if run as a script  
    main()
```

# 路由與 view 函式



■ 網頁瀏覽器的用戶會將 request (請求) 傳送給網頁伺服器，然後再傳送給 Flask app 實例。Flask app 實例根據 URL 將請求分配給對應的函式處理。URL 與處理它的函式之間的關係稱為路由(route)

■ 在 Flask 裡面，使用 app 實例的 `app.route` 裝飾器來宣告路由

■ 像 `index()` 這種處理 app URL 的函式稱為「view 函式」

```
@app.route('/')
```

```
def index():
```

```
    return 'Hello, Flask!'
```

■ 將函式 `index()` 註冊成 app 的根 URL 處理函式

## 另一種設定路由的傳統方法

- Flask 提供另一種將函式 `index()` 註冊成 app 的 URL 處理函式的方法
- 可以使用 `app.add_url_rule()` 來註冊 `index()` 函式，它接收三個引數：URL、端點名稱、view 函式

```
def index():  
    return 'Hello, Flask!'  
app.add_url_rule('/', 'index', index)
```

# 開發 web 伺服器

- Flask 有開發 web 伺服器，可以用 run 命令啟動它

```
if __name__ == "__main__":  
    app.run(host='127.0.0.1', port=5000, debug=True)
```

- 伺服器會在啟動後執行迴圈來接收 requests 並服務它們，這個迴圈會持續執行，直到按下 Ctrl + C 來停止 app 為止
- 在伺服器開始運行之後，連線到 <http://127.0.0.1:5000/>

# 語法解釋

```
from flask import Flask  
app = Flask(__name__)
```

```
@app.route('/')  
def index():  
    return 'Hello, Flask!'
```

```
if __name__ == "__main__":  
    app.run(host='127.0.0.1',  
            port=5000,  
            debug=True)
```

■ 匯入模組：from flask import Flask

■ 裝飾器：@app.route('/')

■ 定義function：def hello\_world():

■ function回傳內容：return 'Hello, World!'

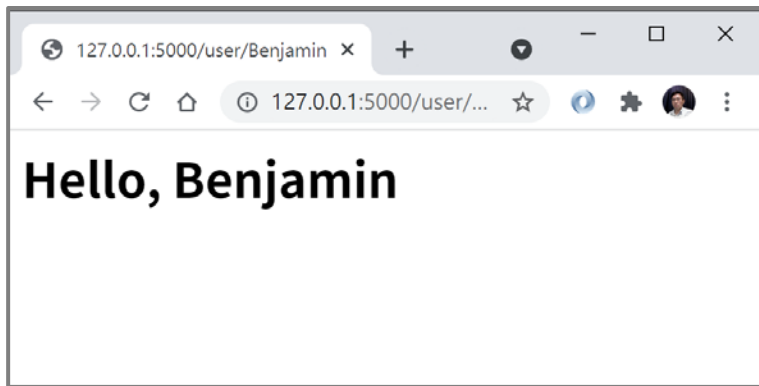
■ \_\_name\_\_：目前的執行體是獨自執行或被其他程式引用

■ \_\_main\_\_：表目前是獨自執行，非其他程式引用



# 動態路由

- 使用動態路由，app 會用 username 動態引數來回應個人化的歡迎訊息



- 開啟範例程式：app\_w\_variable.py
- 嘗試修改範例程式，觀察網頁有何變化

# Lab

- 製作一個能顯示目前時間於網頁的 view 函式



# 動態路由範例程式 (app\_w\_variable.py)

```
■ from markupsafe import escape
   @app.route('/user/<username>')
   def show_user_profile(username): # show the user profile for that user
       return 'User %s' % escape(username)

   @app.route('/post/<int:post_id>')
   def show_post(post_id): # show the post with the given id, the id is an integer
       return 'Post %d' % post_id

   @app.route('/path/<path:subpath>')
   def show_subpath(subpath): # show the subpath after /path/
       return 'Subpath %s' % escape(subpath)
```

# 動態路由變數的資料類型

---

|               |                     |
|---------------|---------------------|
| <b>string</b> | (預設) 接受所有不含斜槓(/)的字串 |
| <b>int</b>    | 接受正整數               |
| <b>float</b>  | 接受正浮點數              |
| <b>path</b>   | 與字串相似，但可允許斜槓(/)     |
| <b>uuid</b>   | 接受UUID字串            |

---

# markupsafe 模組

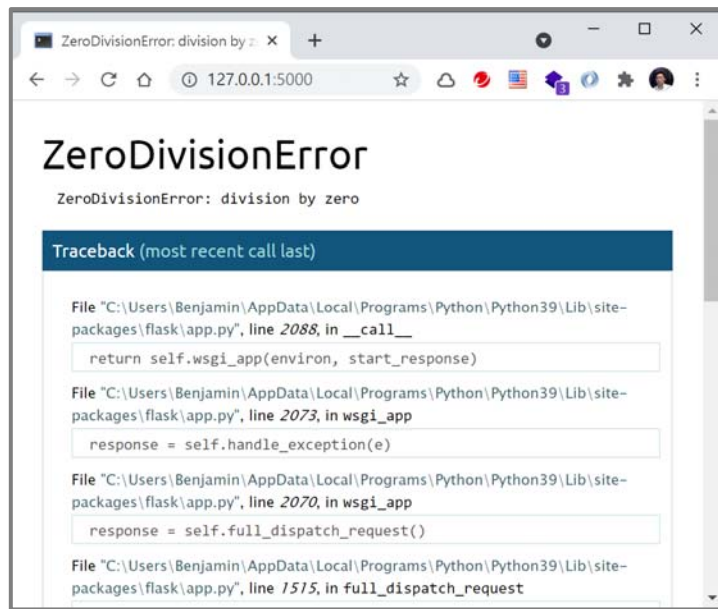
- markupsafe 的功用為阻止注入攻擊
- 防止在動態路由輸入SQL注入或是Javascript程式碼
- 如果使用者注入 `<script>alert("bad")</script>`，`escape()` 函數會將轉譯為文本，從而避免瀏覽器執行注入的Javascript

```
from markupsafe import escape
```

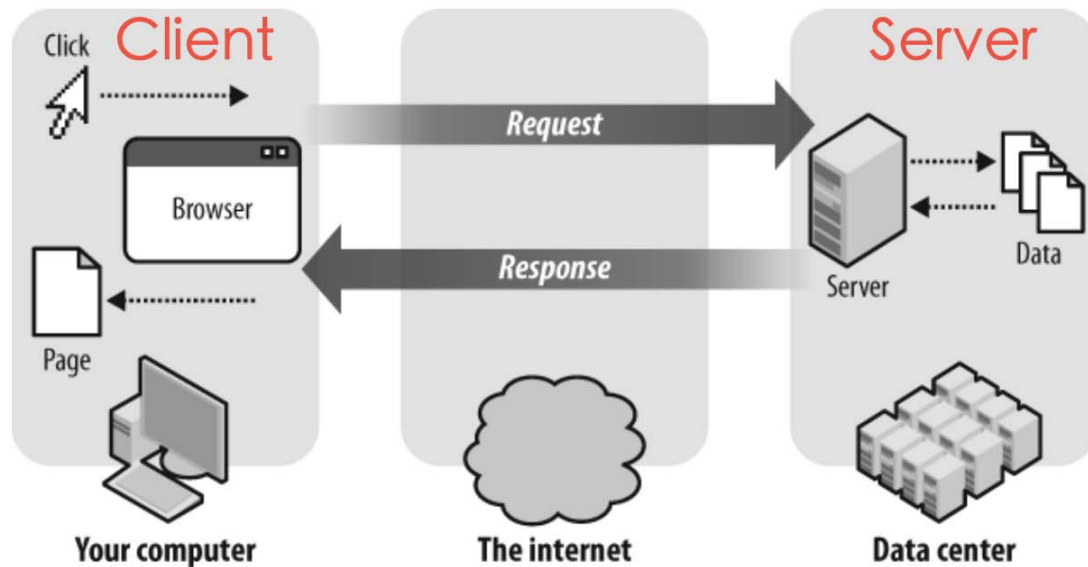
```
@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return 'User %s' % escape(username)
```

# 除錯模式

- Flask app可以選擇使用除錯模式，這個模錯模式預設開啟兩個開發伺服器模組，分別為：reloader 與 debugger
- reloader：Flask會監看專案的原始碼檔案，如果檔案被修改就自動重新啟動伺服器
- debugger：當 app 出現未處理的例外異常時，debugger會在瀏覽器上出現，此時瀏覽器視窗會變成互動式堆疊追蹤 (stack trace)，從而可以檢視原始碼，並在堆疊追蹤的任何地方執行運算式



# Request and response

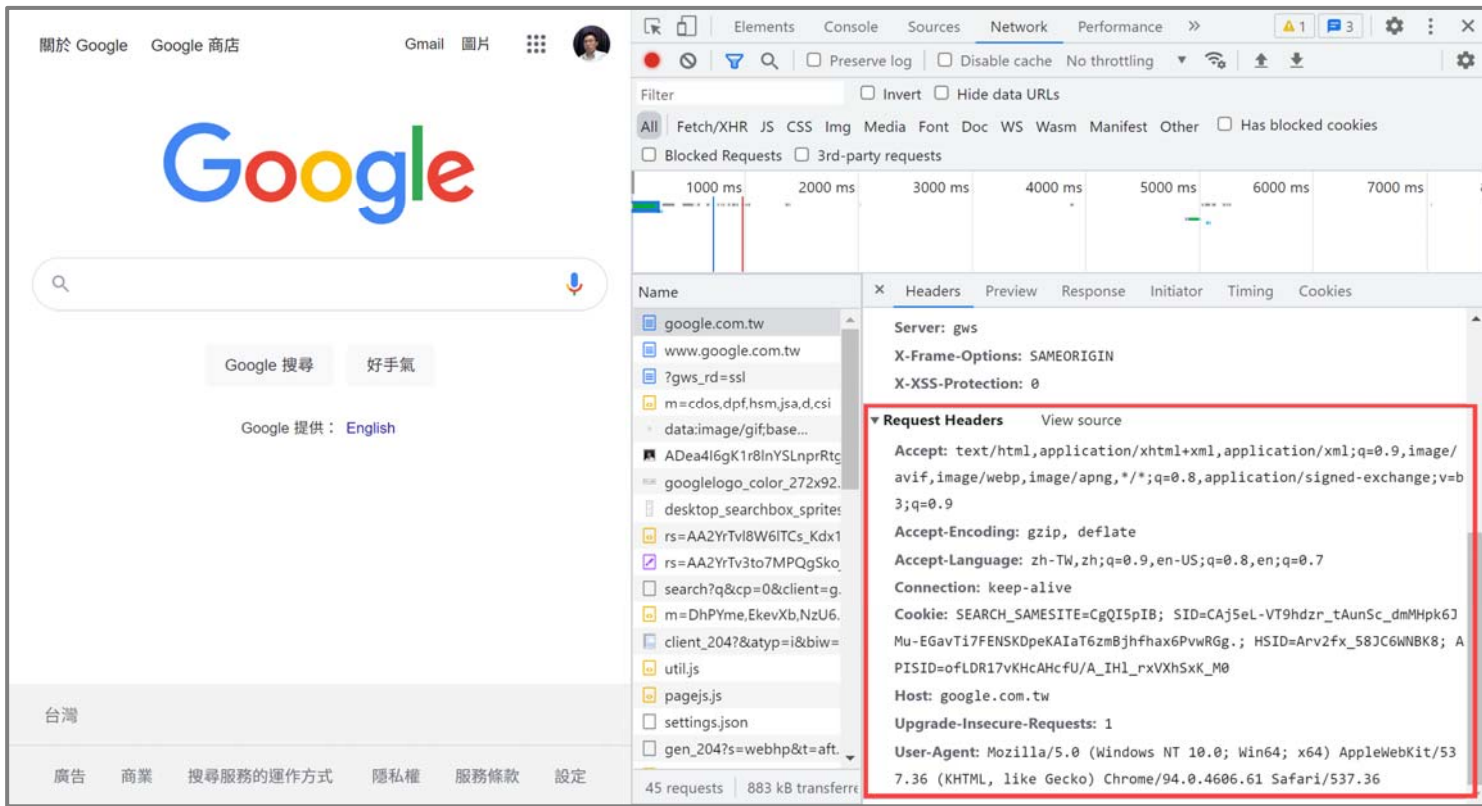


<https://medium.com/@lokeschinni123/django-request-and-response-lifecycle-fae8f6467e3d>





# 使用者瀏覽網頁時，發送的request請求



The screenshot displays the Google homepage in a web browser. The browser's developer tools are open, specifically the Network tab. A list of network requests is shown on the left, with 'google.com.tw' selected. The right pane shows the details of this request, including the Request Headers. The Request Headers are highlighted with a red box and include:

- Server: gws
- X-Frame-Options: SAMEORIGIN
- X-XSS-Protection: 0
- Request Headers** (expanded):
  - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9
  - Accept-Encoding: gzip, deflate
  - Accept-Language: zh-TW,zh;q=0.9,en-US;q=0.8,en;q=0.7
  - Connection: keep-alive
  - Cookie: SEARCH\_SAMESITE=CgQISpIB; SID=CAj5eL-VT9hdzr\_tAunSc\_dmMHpk6JMu-EGavT17FENSKDpeKAIaT6zmBjhfhax6PvwRGg.; HSID=Arv2fx\_58JC6WNBK8; APISID=ofLDR17vKHcAHcfU/A\_IH1\_rxVXhSxK\_M0
  - Host: google.com.tw
  - Upgrade-Insecure-Requests: 1
  - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.61 Safari/537.36

# request 物件 - 1

- Flask request物件內有用戶端發出的HTTP request裡面所有的資訊
- Flask request物件常用的屬性與方法如下：

| 屬性或方法      | 說明                                     |
|------------|----------------------------------------|
| form       | 字典，裡面有以 request 送出的所有表單欄位              |
| args       | 字典，裡面有傳入「URL的查詢字串」的所有引數                |
| values     | 字典，結合 form 與 args 的值                   |
| cookies    | 字典，存有request 之中的所有 cookie              |
| headers    | 字典，裡面有request 內的所有 HTTP 標頭             |
| files      | 字典，裡面有放在 request裡面的所有上傳檔案              |
| get_data() | 回傳 request 內文的緩存資料                     |
| get_json() | 回傳一個 Python 字典，裡面有request 內文中已解析的 JSON |

## request 物件 - 2

| 屬性或方法        | 說明                                                  |
|--------------|-----------------------------------------------------|
| blueprint    | 處理 request 的 Flask 藍圖(blueprint) 名稱                 |
| Endpoint     | 處理request 的 Flask 端點名稱。Flask 會將 view 兩式的名稱當成路由的端點名稱 |
| method       | HTTP request 方法，例如GET 或 POST                        |
| scheme       | URL協定 (http 或https)                                 |
| is_secure()  | 如果 request 是用安全連結 (HTTPS) 傳來的，則回傳 True              |
| host         | 在request 內定義的主機，如果它是用戶端提供的話，也包含連接埠號碼                |
| path         | URL 的路徑部分                                           |
| query_string | URL 的查詢字串部分，用原始二進位值表示                               |
| full_path    | URL的路徑與查詢字串部分                                       |
| url          | 用戶端請求的完整URL                                         |
| base_url     | 與 url 一樣，但沒有查詢字串部分                                  |
| remote_addr  | 用戶端的IP位址                                            |
| environ      | request 的原始 WSGI 環境字典                               |

# request 勾點 (request hook) - 1

- 有時我們可以在處理每一個request之前或之後執行一些有用的程式碼。例如，在處理每一個request 之前，我們可能需要連接資料庫，或驗證發出 request 的使用者。為了避免在每一個 view 函式裡面放入重複的程式碼，Flask 讓你可以註冊在指派request 之前或之後執行的通用函式
- request 勾點 (request hook)使一種裝飾器，Flask提供四種勾點：
  - before\_request
  - before\_first\_request
  - after\_request
  - teardown\_request

# request 勾點 (request hook) - 2

- `before_request`

註冊在每一個request之前的函式

- `before_first_request`

註冊只需要在處理第一個request之前執行的函式，很適合用來加入伺服器初始化工作

- `after_request`

註冊需要在每一個request之後執行的函式，但是只會在沒有被處理的異常狀況時執行

- `teardown_request`

註冊需要在每一個request之後執行的函式，即使在有未處理的異常的形況下

# request 勾點 (request hook) 程式範例

```
from flask import Flask
app = Flask(__name__)

# request 勾點 (request hook)
@app.before_request
def before():
    print('before request!')

# 第一次調用時才會被執行
@app.before_first_request
def before():
    print('before first!!!')

@app.route('/index/')
def index():
    # 1/0 # 如果取消這裡的注釋
    # after_request就不會執行了
    return 'index'
```

```
# 程式在不出異常才會被執行
@app.after_request
def after(response):
    print('after request')
    return response

@app.teardown_request
def teardown(exception):
    print('teardown request')

if __name__ == '__main__':
    app.run()
```

# request 勾點 (request hook) 執行結果

- 第一次執行 (使用者發送request)

- before first!!!

- before request!

- after request

- teardown request

- 第二次執行 (使用者發送request)

- before request!

- after request

- teardown request

# 使用者瀏覽網頁時，接收的response回應

The image shows a screenshot of the Google homepage on the left and the Chrome DevTools Network tab on the right. The Network tab is displaying a list of requests, and the selected request is 'google.com.tw'. The 'Response Headers' section is expanded, showing the following details:

- Request URL:** http://google.com.tw/
- Request Method:** GET
- Status Code:** 301 Moved Permanently
- Remote Address:** 142.251.43.3:80
- Referrer Policy:** strict-origin-when-cross-origin
- Response Headers:**
  - BFCache-Opt-In: unload
  - Cache-Control: public, max-age=2592000
  - Content-Length: 222
  - Content-Type: text/html; charset=UTF-8
  - Date: Sun, 03 Oct 2021 13:34:32 GMT
  - Expires: Tue, 02 Nov 2021 13:34:32 GMT
  - Location: http://www.google.com.tw/
  - Server: gws
  - X-Frame-Options: SAMEORIGIN
  - X-XSS-Protection: 0



# HTTP 狀態碼

- HTTP狀態碼是用以表示 HTTP回應狀態的3位數字代碼
- 由 RFC 2616 規範定義的，並得到 RFC 2518、2817、2295、2774 與 4918 等規範擴充
- HTTP 狀態碼表明一個 HTTP 要求是否已經被完成。回應分為五種：
  - 資訊回應 (Informational responses, 100–199),
  - 成功回應 (Successful responses, 200–299),
  - 重定向 (Redirects, 300–399),
  - 用戶端錯誤 (Client errors, 400–499),
  - 伺服器端錯誤 (Server errors, 500–599).

# response 物件

## ■ Flask response物件如下：

| 屬性或方法           | 說明                      |
|-----------------|-------------------------|
| status_code     | HTTP 數字狀態碼              |
| headers         | 類似字典的物件，裡面有準備以回應傳送的所有標頭 |
| set_cookie()    | 將一個 cookie 加入回應         |
| delete_cookie() | 移除一個 cookie             |
| content_length  | 回應內文的長度                 |
| content_type    | 回應內文的媒體類型               |
| set_data()      | 將回應內文射程字串或 byte 值       |
| get_data()      | 取得回應內文                  |

# Flask response

- Flask的view函式除了回傳文字之外，也可以加上第二個引述作為狀態碼
- `@app.route('/')  
def index():  
 a=1/0  
 return 'Hello, Flask!', 400`
- 400狀態碼代表不良的請求錯誤

# make\_response() 函式

- Flask view函式也可以回傳一個回應物件
- make\_response()函式可以接收多個引數，除了讓view函式可以回傳值之外，也可以回傳一個等效的回應物件。有時在view函式內製作回應物件再使用這個物件提供的方法來進一步設置回應很方便
- 建立一個回應物件，接著在它裡面設置一個cookie：
- ```
from flask import make_response
@app.route('/')
def index():
    response=make_response('<h1>寫入cookie</h1>')
    response.set_cookie('name', 'Benjamin')
    return response
```

# 轉址

- Flask提供一個 `redirect()` 協助函式建立轉址的回應

- `from flask import redirect`

- `@app.route('/')`

- `def index():`

- `return redirect('https://google.com/')`

# Lab

- 執行範例 app.py 檔案
- 嘗試修改程式，修改網站內容



# 重點回顧

- 了解 Flask app 實例
- route 路由的用法
- 動態路由
- request
- response
- response 勾點 (hook)
- make\_response()
- redirect()