模板

會員登入流程

- 商業邏輯(business logic)：負責根據使用者輸入的帳號密碼，搭配資料庫，判斷是否為會員
- 展示邏輯(presentation logic)：登入後，根據是否為會員，顯示不同網頁內容
- 使用 view 函式製作完整的HTML檔案太複雜，而且不好維護。將展示邏輯移到模板可以協助改善app的可維護性
- Flask使用一種強大的模板引擎 Jinja2 來執行轉譯(render)模板的工作

使用模板

- 模板預設目錄「templates」
- 在使用模板之前，需要先使用render_template函式
  from flask import render_template
- app.py
- @app.route('/')
  def index():
  return render_template('index.html')
- @app.route('/user/<username>')
  def show_user_profile(username):
  return render_template('user.html', username = username)
- index.html
  <h1>Hello World</h1>
- user.html
  <h1>Hello {{ username }}</h1>

模板中的變數

- Jinia2 可辦識任何型態的變數，包括串列、字典與物件這些複雜的型態，如：
  - A value from a dictionary: {{ mydict['key'] }}.
  - A value from a list: {{ mylist[3] }}.
  - A value from a list, with a variable index: {{ mylist[myintvar] }}.
  - A value from an object's method: {{ myobj.somemethod() }}.
- 也可以使用過濾器(filter)來修改變數，如：

  Hello, {{ username|capitalize }}

  # 把 username 的第一個字改成大寫，其他的為小寫

Jinja2 變數過濾器

過濾器名稱	說明
Safe	在不轉義 (escape)的情況下轉譯值
Capitalize	值的第一個字元大寫，其他的字元小寫
Lower	將值改成小寫字元
Upper	將值改成大寫字元
Title	值的每一個單字的第一個字母大寫
Trim	移除值的前面與後面的空白字元
striptags	在轉譯前移除值的所有 HTML標籤

Jinja2預設過濾器： https://jinja.palletsprojects.com/en/3.0.x/templates/#builtin-filters

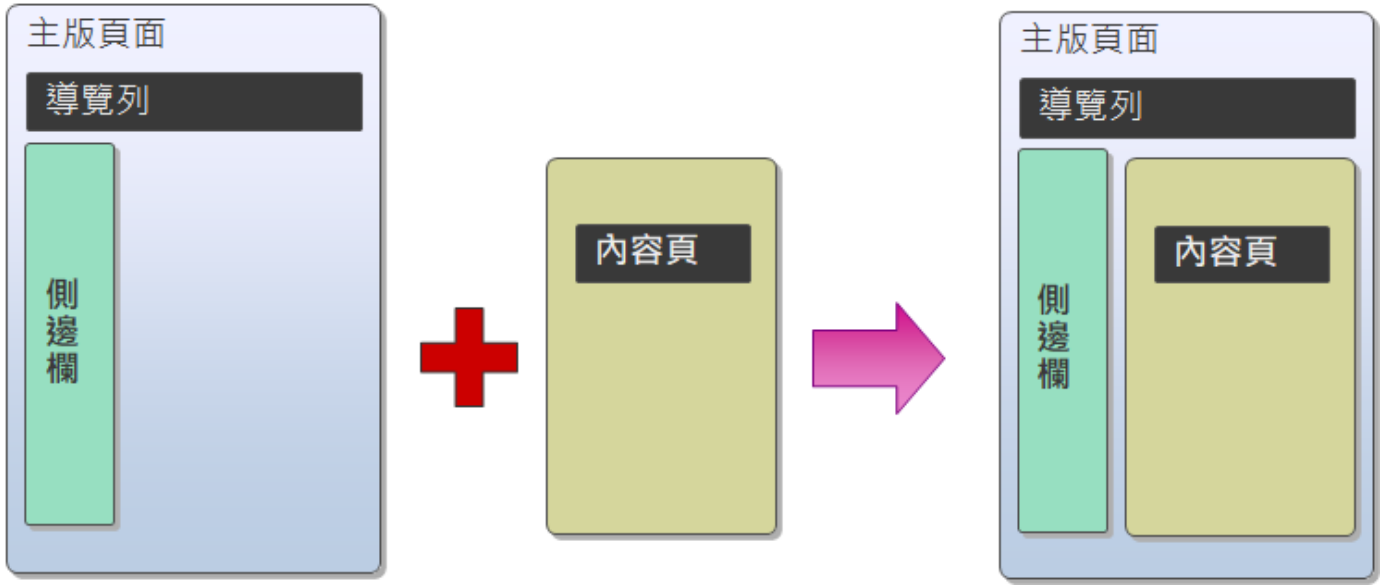
控制結構

- if判斷式
  - {% if name %}
    <h1>Hello {{ name }}! </h1>
    {% else %}
    <h1>Hello, World! </h1>
    {% endif %}
- 迴圈
  -
    {% for item in seq %}
    {{ item }}
    {% endfor %}

使用Jinja2導入頁面

- 把重複使用的模板程式碼放在一個單獨的檔案內，並在有需要使用的模板引用它，用來避免重複的程式撰寫
- 可以引入頁首或頁尾：

```
{% include 'header.html' %}
    Body
{% include 'footer.html' %}
```



產生網頁連結

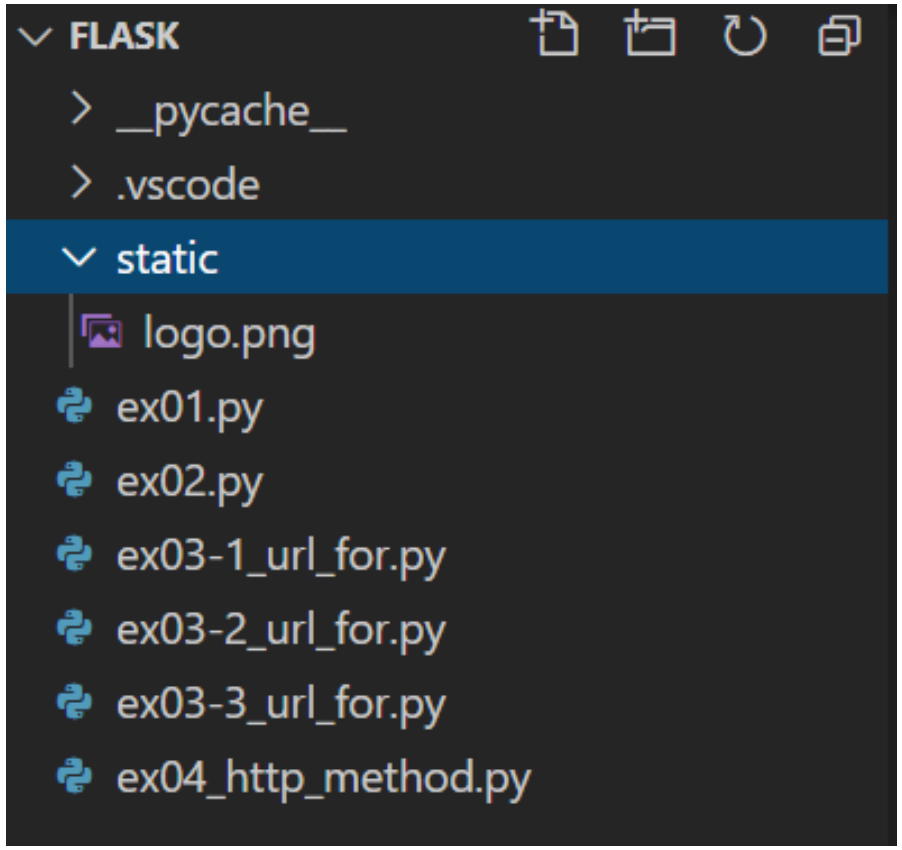
- Flask提供 url_for() 函式，可用 app 的 URL map 裡面的資訊來產生 URL
- url_for('index') 會收到 / ，即 app 的根 URL
- url_for('index', _external=True) 會收到完整的 URL ，如：
  http://127.0.0.1:5000/
- 也可以使用 url_for 產生動態 URL，如：
  url_for('user', name='Benjamin', _external_True)，會收到
  http://127.0.0.1:5000/user/Benjamin

靜態檔案

- Flask 預設會在 app 根目錄的 static 子目錄中尋找靜態檔案
- 可以把 CSS, Javascript 檔案放到 static 子目錄中提供網頁使用
- 把bootstrap的CSS與Javascript分別放到static/css與static/js資料夾內
- 為了使用 bootstrap 的 CSS 檔，可以使用 url_for() 函式產生連結路徑

```
<link href="{{ url_for('static', filename='css/bootstrap.min.css') }}"
rel="stylesheet">
```

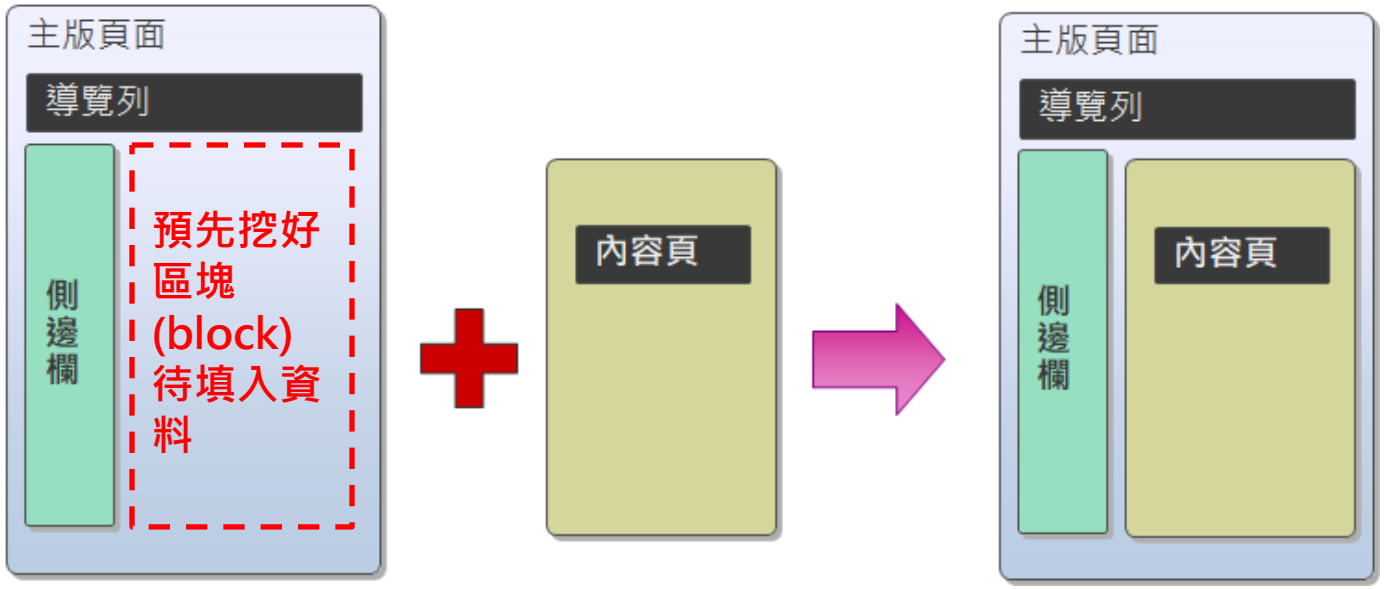
靜態檔案示範



- 啟動網站後，將可以連線到 http://127.0.0.1:5000/static/logo.png 讀取 logo.png 檔案。

模板繼承

- 預先製作網頁版型，預留區塊
- 多個頁面可以使用同一個網頁版型，填入不同區塊即可呈現網頁



模板繼承語法

- base.html (模板頁)

```
<head>
    <title>{% block title%}{% endblock title%}</title>
</head>
<body>
    {% block message %}{% endblock %}
</body>
```

- index.html (呈現網頁)

```
{% extends 'base.html' %}
{% block title %}Homepage{% endblock %}
{% block message %}
<h1>Hello World</h1>
{% endblock message %}
```

Jinja2 的符號使用說明

- {{ 兩個大括號是參數綁定 }}
- {% 這樣是執行 Jinja2 內置函數 %}
- {# 這樣是註解說明 #}

自訂錯誤網頁

- Flask可以讓app用模板來定義自己的錯誤網頁，最常見的兩種錯誤碼是404(在用戶端請求未知的網頁或路由時觸發)以及500(在app有未處理的異常狀況時觸發)
- 使用 app.errorhandler 裝飾器來自訂錯誤網頁
-
```
@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html'), 404
@app.errorhandler(500)
def page_error(e):
    return render_template('500.html'), 500
```

重點回顧

- 使用模板
- 模板繼承
- 靜態檔案
- 自訂錯誤網頁
- url_for()