

Python jinja2 + flask

Flask 概述

什麼是 Web Framework?

Web Application Framework (Web 應用程序框架) 或簡單的 Web Framework (Web 框架) 表示一個庫和模塊的集合, 使 Web 應用程序開發人員能夠編寫應用程序, 而不必擔心協議, 線程管理等低級細節。

什麼是 Flask?

Flask 是一個用 Python 編寫的 Web 應用程序框架。 它由 Armin Ronacher 開發, 他領導一個名為 Pocco 的國際 Python 愛好者團隊。 Flask 基於 Werkzeug WSGI 工具包和 Jinja2 模板引擎。兩者都是 Pocco 項目。

WSGI

Web Server Gateway Interface (Web 服務器網關接口, WSGI) 已被用作 Python Web 應用程序開發的標準。 WSGI 是 Web 服務器和 Web 應用程序之間通用接口的規範。

Werkzeug

它是一個 WSGI 工具包, 它實現了請求, 響應對象和實用函數。 這使得能夠在其上構建 web 框架。 Flask 框架使用 Werkzeug 作為其基礎之一。

jinja2

jinja2 是 Python 的一個流行的模板引擎。 Web 模板系統將模板與特定數據源組合以呈現動態網頁。

Flask 通常被稱為微框架。 它旨在保持應用程序的核心簡單且可擴展。 Flask 沒有用於數據庫處理的內置抽象層, 也沒有形成驗證支持。相反, Flask 支持擴展以向應用程序添加此類功能。一些受歡迎的 Flask 擴展將在本教程後續章節進行討論。

```
pyView
@app.route('/<name>'):
def index(name):
    return render_template('user.html', name=name)
```

```
user.html
<p>
    i am {{ name }}
</p>
```

不僅字串可以傳遞, 如 dict、list、甚至是 method 都可以傳遞。

```
<p>
    dict:{{ dict['keys']}}
</p>
<p>
    lsit:{{ list[0] }}
</p>
<p>
    method:{{ obj.method() }}
</p>
```

並且可以配合過濾器來做加工(透過過濾器來將字母轉大寫)

```
<p>
    name = {{ name|upper }}
</p>
```

過濾器

變數搭配過濾器可以做顯示上的一些調控，或是傳值上的一些調控。

比如說有些報表的參數只吃大寫字母之類的。

而且不限搭一個

```
<p>
    name = {{ name|upper|striptags }}
</p>
```

上面的範例除了做大寫之外，還將 html 的標籤拿掉。

過濾器清單

過濾器	說明	備註
abs(number)	絕對值	
attr(obj,name)	取得物件屬性	foo
batch(value,linecount,fill_with=None)		
capitalize (s)	第一個字母轉大寫，其餘為小寫	
center(value,width=80)	置中	
default(value,default_value=u' ' ,boolean=False)	如果為 undefined，就回給予預設置，否則維持原變數的值	
dictsort(value,case_sensitive=False,by= 'key')	排序 dict，因為 dict 本身無序	
escape(s)		
filesizeformat(value,binary=False)		
first(seq)	回傳序列的第一個項目	
float(value,default=0.0)	格式轉為浮點數，若轉換失敗則回傳 default	
forceescape(value)		
format(value,*args,**kwargs)	如 python 的字串格式化應用	
groupby(value,attribute)		
indent(s,width=4,indentfirst=False)		
int(value,default=0)	格式轉為整數，若轉換失敗則回傳 default	
join(value,d=u' ' ,attribute=None)	如 python 的 join 般串接字串	參數 d 是串接符號設置
last(seq)	回傳序列的最後一個項目	
length(object)	回傳序列內的數量	
list(value)	將值轉 list	
lower(s)	將字母轉小寫	
map()		
pprint(value,verbose=False)	漂亮的列印出資料，用於測試	
random(seq)	從序列中隨機返回一個項目	
reject()		
rejectattr()		
replace(s, old, new, count=None)	字串替換	
reverse(value)		
round(value, precision=0, method= 'common')	四捨五入	method 有三種可參閱官方說明
safe(value)	信任來源資料	
select()		
selectattr()		
slice(value, slices, fill_with=None)		
sort(value, reverse=False, case_sensitive=False, attribute=None)	排序迭代器	可調控由大至小或由小至大

過濾器	說明	備註
string(object)		
striptags(value)		
sum(iterable, attribute=None, start=0)	返回一系列加上 start 值的數列	
title(s)		
trim(value)	去頭尾空白	
truncate(s, length=255, killwords=False, end=' ...')	截斷字串，由 length 控制長度	
upper(s)	字串轉大寫	
urlencode(value)	轉 url 編碼	
urlize(value, trim_url_limit=None, nofollow=False)	將純文字的 url 轉可按連結	
wordcount(s)	計算字串總字數	
wordwrap(s, width=79, break_long_words=True, wrapstring=None)		
xmlattr(d, autospace=True)	依 dict 項目生成 sgml/xml 屬性字串符號	

測試

在 jinja2 中，除了過濾器之外，也還可以透過測試的方式來確認相關變數是否正常傳遞。

```
{% if loop.index is divisibleby 3 %}
```

```
name is defined
```

測試清單

測試模式	說明	備註
callable(object)	回傳物件是否可以調用	
defined(value)	如果變數已宣告，回傳 true	
divisibleby(value, num)	檢查是否可以變整除	
escaped(value)	檢查是否被轉譯	
even(value)	若為偶數，回傳 true	
iterable(value)	檢查是否可被迭代	
lower(value)	若為小寫，回傳 true	
mapping(value)	若為 dict，回傳 true	
none(value)	若為 none，回傳 true	
number(value)	若為數值，回傳 true	
odd(value)	若為奇數，回傳 true	
sameas(value, other)	檢查兩物件是否指向相同記憶體位址	
sequence(value)	若為序列，回傳 true	
string(value)	若為字串，回傳 true	
undefined(value)	若為 undefined，回傳 true	
upper(value)	若為大寫，回傳 true	

註解的寫法

```
{# 我的註解 #}

{# note: disabled template because we no longer use this
   {% for user in users %}

       ...

   {% endfor %}
#}
```

空白控制

預設空白的部份不會有任何的修正，但如果在表示式中加入『-』就會移除

```
{% for item in seq -%}
```

```
    {{ item }}
```

```
{%- endfor %}
```

如果 seq 是 1 到 9，那就會產出 123456789，中間不帶任何空白。

tag 跟-之間是不允許空格，否則空白的控制是會失效的。

有效的:

```
{%- if foo -%}...{% endif %}
```

無效的:

```
{% - if foo - %}...{% endif %}
```

行語句

使用了行語句，就可以透過一個 tag 來代表表達式。

```
<ul>
```

```
# for item in seq
  <li>{{ item }}</li>
# endfor
</ul>
```

```
<ul>
{% for item in seq %}
  <li>{{ item }}</li>
{% endfor %}
</ul>
```

樣板繼承

jinja2 中最好用的部份就是樣板的繼承。

好處就是不用寫太多重覆性的東西。

透過 block 的調整即可。

直接看官方的範例!

樣板繼承範例

```
base.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  {% block head %}
  <link rel="stylesheet" href="style.css" />
  <title>{% block title %}{% endblock %} - My Webpage</title>
  {% endblock %}
</head>
<body>
  <div id="content">{% block content %}{% endblock %}</div>
  <div id="footer">
    {% block footer %}
    &copy; Copyright 2008 by <a href="http://domain.invalid/">you</a>.
    {% endblock %}
  </div>
</body>
```

上面的部份我們定義為基板，其中會發現有幾個表達式

```
{% block head%}...{% endblock head%}
```

這個 block 就是讓我們可以操作的區塊。

```
i_base_1
{% extends "base.html" %}
{% block title %}Index{% endblock %}
{% block head %}
  {{ super() }}
  <style type="text/css">
    .important { color: #336699; }
  </style>
{% endblock %}
{% block content %}
  <h1>Index</h1>
  <p class="important">
    Welcome on my awesome homepage.
  </p>
{% endblock %}
```

可以看到在一開始的時候我們宣告了一段

```
{% extends "base.html" %}
```

這段即宣告樣板繼承自 base.html

接著，針對了 block_title 的部份設置為 index，也保留了 block_head。

這樣子的作法另一個好處大概就是分離關注點吧？

每個 block 的 name 是唯一的。

保留樣板 block 內容

剛才的案例，關於 block_head 的部份提到可以保留。

靠的就是在裡面插入一個表達式來繼承母板表達式內的內容

```
{{ super() }}
```

```
{% block head %}
    {{ super() }}
    <style type="text/css">
        .important { color: #336699; }
    </style>
{% endblock %}
```

轉譯

預設情況下，jinja2 會做自動轉譯

轉譯的意思是什麼？

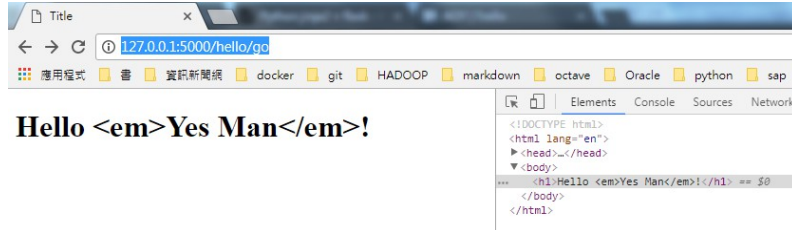
舉例來說

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

```
@app.route('/hello/<name>')
def hello(name=None):
    if name is not None:
        name = '<em>Yes Man</em>'
    return render_template('hello.html', name=name)
<h1>Hello {{ name }}!</h1>
```

在時候，在網址上填入『http://127.0.0.1:5000/hello/go』

這時候，網頁上所顯示的是 Hello Yes Man



Hello Yes Man!

這就是 jinja2 將所拋過來的資料做了轉譯，預防 html 被注入！

如果不想被自動轉譯的話，就可以加上表達式。

```
{% autoescape false %}
    <h1>Hello {{ name }}!</h1>
{% endautoescape %}
```

表達式

for
for 的表達式跟 pytho 用法相似，但在 jinja2 的部份有支援 else。
在整個 for 走完的時候就會進入 else 內。

```
dict 範例
<dl>
{% for key, value in my_dict.iteritems() %}
    <dt>{{ key|e }}</dt>
    <dd>{{ value|e }}</dd>
{% endfor %}
</dl>
```

```
list 範例
<h1>Members</h1>
<ul>
{% for user in users %}
    <li>{{ user.username|e }}</li>
{% endfor %}
</ul>
```

在 jinja2 的迭代中無法 break，所以只能以透過過濾來處理。

```
{% for user in users if not user.hidden %}
    <li>{{ user.username|e }}</li>
{% endfor %}
```

特殊變數

在 for 的期間有些特殊的變數是可以取得的

變數	描述
loop.index	目前迭代的次數（從 1 開始）
loop.index0	目前迭代的次數（從 0 開始）
loop.revindex	到迭代結束還需要的次數（從 1 開始
loop.revindex0	到迭代結束還需要的次數（從 0 開始
loop.first	如果是第一次迭代，則為 True。
loop.last	如果是最後一次迭代，則為 True。
loop.length	序列中的項目數。
loop.cycle	在一串序列間取值的輔助函數。

```

if
用法上跟 python 是一致的
{% if kenny.sick %}
    Kenny is sick.
{% elif kenny.dead %}
    You killed Kenny! You bastard!!!
{% else %}
    Kenny looks okay --- so far
{% endif %}

```

macro_宏

一個類似 function 的作法!

macro_導入

實務上會將 macro 寫在一個 html 內，需要的時候再 import 就可以了。

```

{% import 'form.html' as form %}
<p>{{ form.input('input1', value='user') }}</p>
<p>{{ form.input('input2', 'password') }}</p>

```

也可以跟 python 一樣的作法

```

{% from 'form.html' import input %}
<p>{{ input('input1', value='user') }}</p>
<p>{{ input('input2', 'password') }}</p>

```

macro_範例

```

{% macro input(name, value='', type='text', size=20) -%}
    <input type="{{ type }}" name="{{ name }}" value="{{
        value|e }}" size="{{ size }}">
{%- endmacro %}

```

以 macro 開始，以 endmacro 來結束。

input 是它的名稱。

使用方式如下

```

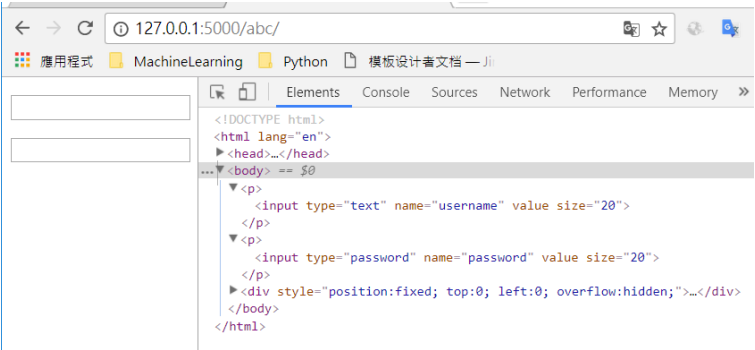
<p>{{ input('username') }}</p>
<p>{{ input('password', type='password') }}</p>

```

當你去透過 macro input 來做渲染的時候，就會生成一個 input 的元件在你的 html 上，並且 value 預設為空，type 為 text。

看下面的 password，type 的部份是可以再調整的，此例即調整為 password。

這麼做的好處在於，你要建置一個 input 的時候不需要<input ...xxx>的輸入一堆，而僅僅是透過 macro 來宣告你要的 input 的 name、value、type 即可。也算是一種優雅!

<pre> <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>Title</title> </head> <body> {% macro input(name, value='', type='text', size=20) -%} <input type="{{ type }}" name="{{ name }}" value="{{ value e }}" size="{{ size }}"> {%- endmacro %} <p>{{ input('username') }}</p> <p>{{ input('password', type='password') }}</p> </body> </html> </pre>	
---	--

macro caller

caller 的部份，可以想成是丟一個參數給 macro 去生成。

怎麼說，看下面的例子！
一樣的，我們產生一個 macro，並且在後面的部份加了一個{{ caller() }}的表達式。
{% macro list_users(users) -%}

```
<table>
  <tr><th>Name</th><th>Memo</th></tr>
  {% for user in users %}
    <tr><td>{{ user.name | e }}</td>{{ caller() }}</tr>
  {% endfor %}
</table>
{% endmacro %}
```

接著，我們透過 set 來設置一些資料
{% set users=[{'name':'Tomcat'},
 {'name':'Johndog'},
 {'name':'Marybird'}]
%}

使用 macro
{% call list_users(users) %}
 <td><input name="Call" type="button" value="Caller"> </td>
{% endcall %}

Name	Memo
Tomcat	<input type="button" value="Caller"/>
Johndog	<input type="button" value="Caller"/>
Marybird	<input type="button" value="Caller"/>

在渲染網頁的時候，後面的 caller 就會被 call 內的元件給取代掉

所以在迭代的同時，也將每個欄位都加入一個 button。

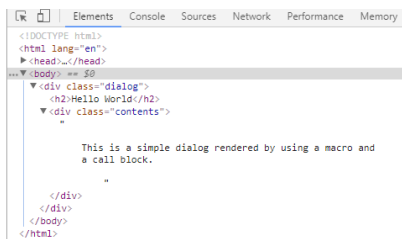
macro_call_帶參數
{% macro render_dialog(title, class='dialog') -%}
 <div class="{{ class }}">
 <h2>{{ title }}</h2>
 <div class="contents">
 {{ caller() }}
 </div>
 </div>
{% endmacro %}

{% call render_dialog('Hello World') %}
 This is a simple dialog rendered by using a macro and
 a call block.
{% endcall %}

可以看到，title 的部份是以 Hello World 去帶入。

Hello World

This is a simple dialog rendered by using a macro and a call block.



過濾_filter

```
{% filter upper %}
```

This text becomes uppercase

```
{% endfilter %}
```

賦值_set

```
{% set navigation = [('index.html', 'Index'), ('about.html', 'About')] %}
```

```
{% set key, value = call_something() %}
```

include

不同於 extend 的繼承方式，include 的話就是把你 include 過來的網頁直接整個渲染在設置的位置上。

```
<body>
```

```
...
```

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

```
</body>
```

一個問題即是，如果沒有這個來源檔案會報錯。

所以在設置上可以透過 ignore missing 來調適。

```
{% include 'footer.html' ignore missing %}
```

上面案例的意思即是如果沒有就放棄引入渲染。

```
{% include 'footer.html','footer2.html','footer3.html' ignore missing %}
```

include 也可以多板，以上例來看，沒有 footer 就 footer2...

上下文變量與函數使用

request

官方文件_request method

利用表達式的方式，也可以取得請求端的資訊！

```
{{ request.url }}
```

更多的應用可以參閱官方文件說明

session

session 是一個 dict 性質的物件。

```
{{ session.Count }}
```

舉例來說

```
from flask import Flask, render_template, session
```

```
app = Flask(__name__)
```

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

```
def index():
```

```
    session['Count'] = 'admin'
```

```
    return render_template('hello.html')
```

```
# Key 的部份在正式環境上會置於 instanceos\config 內
```

```
app.secret_key = '77'
```

<p>{{ request.url }}</p>

```
<p>{{ session['Count'] }}</p>
```

或是

```
<p>{{ request.url }}</p>
```

```
<p>{{ session.Count }}</p>
```

g

g 的部份本身用來記錄全域變數。

在後端所定義的 g，前端也是可以呼叫使用。

```
from flask import Flask, render_template, g
```

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

```
def indexddd():
```

```
    g.db = 'mysql'
```

```
    return render_template('hello.html')
```

```
<p>db: {{ g.db }}</p>
```

config

對於 flask 的 config 的部份，本身亦為全域變數。

故在前端也可以調用。

```
{{ config.DEBUG }}
```

自定義

自定義變數

待理解

自定義函數

透過裝飾器 context_processor 可以自定義函數提供前端調用。

```
from datetime import datetime
```

```
@app.context_processor
```

```
def get_current_time():
```

```
    def get_time():
```

```
        return datetime.now()
```

```
<p>Now Time is: {{ current_time() }}</p>
```

相關資料參考

<https://flask.palletsprojects.com/en/1.1.x/>

<https://jinja.palletsprojects.com/en/2.11.x/>

<https://hackmd.io/@shaoeChen/SJ0X-PnkG?type=view>

https://www.w3cschool.cn/flask/flask_variable_rules.html

<https://www.tutorialspoint.com/flask/index.htm>

Index.html	<pre><!DOCTYPE html> <html> <head> <title>Page Title</title> </head> <body> <h1>這是標題 1</h1> <p>這是段落</p> Coffee Tea Milk <table> <tr> <th>First Name</th> <th>Last Name</th> <th>Points</th> </tr> <tr> <td>Peter</td> <td>Griffin</td> <td>\$100</td> </tr> <tr> <td>Lois</td> <td>Griffin</td> <td>\$150</td> </tr> <tr> <td>Joe</td> <td>Swanson</td> <td>\$300</td> </tr> </table> </body> </html></pre>

CSS	
	<pre><style> body { background-color: lightblue; } h1 { color: white; text-align: center; } p { font-family: verdana; font-size: 20px; } #myA{ } .myB{ } #myA p{ } </style></pre>

基本 css 和 html	<pre><!DOCTYPE html> <html> <head> <style> body { background-color: lightblue; } h1 { color: white; text-align: center; } p { font-family: verdana; font-size: 20px; } </style> </head> <body> <h1>這是標題</h1> <p>這是內容</p> </body> </html></pre>
myjs01.html	<pre><!DOCTYPE html> <html> <body> <h2>My First JavaScript</h2> <button type="button" onclick="document.getElementById('demo').innerHTML = Date()"> Click me to display Date and Time.</button> <p id="demo"> </p> </body> </html></pre>

myjs02.html	
	<pre> <!DOCTYPE html> <html> <body> <button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">Turn on the light</button> <button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Turn off the light</button> </body> </html> </pre>
myjs03.html	
	<pre> <!DOCTYPE html> <html> <body> <p id="demo"> </p> <script> function myFunction(p1, p2) { return p1 * p2; } document.getElementById("demo").innerHTML = myFunction(4, 3); </script> </body> </html> </pre>

RWD01

響應式網頁設計(Responsive Web Design) 使用 CSS

page01.html	
	<pre><!DOCTYPE html> <html> <head> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <style> * { box-sizing: border-box; } .row::after { content: ""; clear: both; display: table; } [class*="col-"] { float: left; padding: 15px; } html { font-family: "Lucida Sans", sans-serif; } .header { background-color: #9933cc; color: #ffffff; padding: 15px; } .menu ul { list-style-type: none; margin: 0; padding: 0; } .menu li { padding: 8px; margin-bottom: 7px; background-color: #33b5e5; color: #ffffff;</pre>


```
    box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.menu li:hover {
    background-color: #0099cc;
}

.aside {
    background-color: #33b5e5;
    padding: 15px;
    color: #ffffff;
    text-align: center;
    font-size: 14px;
    box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.footer {
    background-color: #0099cc;
    color: #ffffff;
    text-align: center;
    font-size: 12px;
    padding: 15px;
}

/* For mobile phones: */
[class*="col-"] {
    width: 100%;
}

@media only screen and (min-width: 600px) {
    /* For tablets: */
    .col-s-1 {width: 8.33%;}
    .col-s-2 {width: 16.66%;}
    .col-s-3 {width: 25%;}
    .col-s-4 {width: 33.33%;}
    .col-s-5 {width: 41.66%;}
    .col-s-6 {width: 50%;}
    .col-s-7 {width: 58.33%;}
    .col-s-8 {width: 66.66%;}
    .col-s-9 {width: 75%;}
    .col-s-10 {width: 83.33%;}
    .col-s-11 {width: 91.66%;}
    .col-s-12 {width: 100%;}
}
```

```

@media only screen and (min-width: 768px) {
  /* For desktop: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}
</style>
</head>
<body>

<div class="header">
  <h1>Chania</h1>
</div>

<div class="row">
  <div class="col-3 col-s-3 menu">
    <ul>
      <li>The Flight</li>
      <li>The City</li>
      <li>The Island</li>
      <li>The Food</li>
    </ul>
  </div>

  <div class="col-6 col-s-9">
    <h1>The City</h1>
    <p>Chania is the capital of the Chania region on the island of Crete. The city can be
divided in two parts, the old town and the modern city.</p>
  </div>

  <div class="col-3 col-s-12">
    <div class="aside">
      <h2>What?</h2>
      <p>Chania is a city on the island of Crete.</p>
      <h2>Where?</h2>
    </div>
  </div>

```

```
<p>Crete is a Greek island in the Mediterranean Sea.</p>
<h2>How?</h2>
<p>You can reach Chania airport from all over Europe.</p>
</div>
</div>
</div>

<div class="footer">
  <p>Resize the browser window to see how the content respond to the resizing.</p>
</div>

</body>
</html>
```

RWD02

響應式網頁設計(Responsive Web Design) 使用 bootstrap

page01.html	
	<pre><!DOCTYPE html> <html lang="en"> <head> <title>Bootstrap Example</title> <meta charset="utf-8"> <meta name="viewport" content="width=device-width, initial-scale=1"> <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script> <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script> <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></script> </head> <body> <div class="jumbotron text-center"> <h1>My First Bootstrap Page</h1> <p>Resize this responsive page to see the effect!</p> </div> <div class="container"> <div class="row"> <div class="col-sm-4"> <h3>Column 1</h3> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit...</p> <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...</p> </div> <div class="col-sm-4"> <h3>Column 2</h3> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit...</p> <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...</p> </div> <div class="col-sm-4"> <h3>Column 3</h3> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit...</p> <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...</p> </div> </div> </div> </body> </html></pre>

Class01

app.py	
	<pre>from flask import Flask import os app=Flask(__name__) @app.route('/') def index(): return '<h1>hello world!!</h1>' if __name__ == "__main__": port = int(os.environ.get('PORT', 5000)) app.run(host='127.0.0.1', port=port)</pre>

Class02

app.py	
	<pre>from flask import Flask import os app=Flask(__name__) @app.route('/') def index(): return '<h1>hello world!!</h1>' if __name__ == "__main__": port = int(os.environ.get('PORT', 5000)) app.run(host='0.0.0.0', port=port)</pre>
Procfile	web: python app.py
requirements.txt	Flask==1.0.2

Class03

app.py	
	<pre>from flask import Flask import os app=Flask(__name__) @app.route('/') def index(): return '<h1>hello world!!</h1>' @app.route('/user/<name>') def user(name): return '<h1>hello,{!</h1>'.format(name) if __name__ == "__main__": port = int(os.environ.get('PORT', 5000)) app.run(host='127.0.0.1', port=port)</pre>

Class04

app.py	<pre>from flask import Flask, render_template import os app=Flask(__name__) @app.route('/') def index(): return render_template('index.html') @app.route('/user/<name>') def user(name): return render_template('user.html',name=name) @app.route('/user01/<name>') def user01(name): return render_template('user01.html',name=name) @app.route('/user02') def user02(): users02 = [{"username": "超人一號", "url": "https://www.pcschool.com.tw"}, {"username": "超人二號", "url": "https://www.pcschool.com.tw"}] return render_template('user02.html',users=users02) if __name__ == "__main__": port = int(os.environ.get('PORT', 5000)) app.run(host='127.0.0.1', port=port)</pre>
templates /index.html	<pre><!DOCTYPE html> <html> <head> <title>Page Title</title> </head> <body> <h1>hello world</h1> </body> </html></pre>

templates /user.html	
	<!DOCTYPE html> <html> <head> <title>Page Title</title> </head> <body> <h1>hello, {{name}}</h1> </body> </html>
templates /user01.html	
	<!DOCTYPE html> <html> <head> <title>Page Title</title> </head> <body> {% if name=='123' %} <h1>HELLO,{{name}}!</h1> {% else %} <h1>HELLO,Stranger!</h1> {% endif %} </body> </html>
templates /user02.html	
	<!DOCTYPE html> <html> <head> <title>Page Title</title> </head> <body> {% for user in users %} {{ user.username }} {% endfor %} </body> </html>

Class05

app.py	
	<pre>from flask import Flask, render_template import os app=Flask(__name__) @app.route('/') def index(): return render_template('index.html') if __name__ == "__main__": port = int(os.environ.get('PORT', 5000)) app.run(host='127.0.0.1', port=port)</pre>
templates /index.html	
	<pre>{% extends "layout.html" %} {% block title %}Index{% endblock %} {% block head %} <style type="text/css"> .important { color: #3f0430; text-decoration: underline; } </style> {% endblock %} {% block content %} <h1>Index</h1> {{ super() }} <h2>這個是子模板中的標題二</h2> <p class="important"> Welcome on my awesome homepage. </p> {% endblock %}</pre>
templates /layout.html	
	<pre><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"> <html lang="en"> <html xmlns="http://www.w3.org/1999/xhtml"> <head> {% block head %} <title>{% block title %}{% endblock %} - My Webpage</title></pre>

```
{% endblock %}
</head>
<body>
  <div id="content">
    {% block content %}
      <h1>這個是父模板中的標題一</h1>
    {% endblock %}

  </div>
  <div id="footer">
    {% block footer %}
      &copy; Copyright 2008 by <a href="http://domain.invalid/">you</a>.
    {% endblock %}
  </div>
</body>
```

Class06

app.py	
	<pre>from flask import Flask, render_template import os app=Flask(__name__) @app.route('/user/<name>') def user(name): return render_template('user.html',name=name) #return render_template('user.html',name=None) if __name__ == "__main__": port = int(os.environ.get('PORT', 5000)) app.run(host='127.0.0.1', port=port)</pre>
templates /layout.html	
	<pre><!DOCTYPE html> <html> <head> <title>Page Title</title> <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}"> </head> <body> <div id="page"> {% block body %} {% endblock %} </div> </body> </html></pre>
templates /user.html	
	<pre><!DOCTYPE html> <html> <head> <title>Page Title</title> </head> <body> {% extends "layout.html" %} {% block body %} {% if name %} <h1>Hello {{ name }}!</h1></pre>

	<pre>{% else %} <h1>Hello World!</h1> {% endif %} {% endblock %} </body> </html></pre>
static /style.css	
	<pre>#page{ background-color: blue; width: 800px; height: 200px; }</pre>

Class07

app.py	
	<pre>from flask import Flask, render_template import os app=Flask(__name__) @app.route('/') def index(): return render_template('index.html') @app.route('/index01') def index01(): return render_template('index01.html') if __name__ == "__main__": port = int(os.environ.get('PORT', 5000)) app.run(host='127.0.0.1', port=port)</pre>
templates /index.html	
	<pre><!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>Title</title> </head> <body> {% macro input(type="text",value="",size=50) %} <input type="{{type}}" value="{{value}}" size="{{size}}"> {% endmacro %} <p>預設:</p>{{ input() }} <p>密碼框:</p>{{ input("password") }} <p>size=100:</p>{{ input(size=100) }} </body> </html></pre>
templates /index01.html	

	<pre><!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>Title</title> </head> <body> <h1>import minput01.html</h1> {% import "minput01.html" as minput %} <p>預設:</p>{{ minput.input() }} <p>密碼框:</p>{{ minput.input("password") }} <p>size=100:</p>{{ minput.input(size=100) }} </body> </html></pre>
templates /minput01.html	
	<pre>{% macro input(type="text",value="",size=50) %} <input type="{{type}}" value="{{value}}" size="{{size}}"> {% endmacro %}</pre>

Class08

app.py	
	<pre>from flask import Flask, render_template, request from flask_bootstrap import Bootstrap import os app=Flask(__name__) Bootstrap(app) @app.route('/form',methods = ['GET','POST']) def hello_form(): if request.method == 'POST': name=request.form.get('name') print(name) return render_template('form.html',name=name) else: return render_template('form.html') if __name__ == "__main__": port = int(os.environ.get('PORT', 5000)) app.run(host='127.0.0.1', port=port)</pre>
templates /base.html	
	<pre>{% block doc -%} <!DOCTYPE html> <html{% block html_attribs %}{% endblock html_attribs %}> {%- block html %} <head> {%- block head %} <title>{% block title %}{{title default}}{% endblock title %}</title> {%- block metas %} <meta name="viewport" content="width=device-width, initial-scale=1.0"> {%- endblock metas %} {%- block styles %} <!-- Bootstrap --> <link href="{{bootstrap_find_resource('css/bootstrap.css', cdn='bootstrap')}}" rel="stylesheet" "> {%- endblock styles %} {%- endblock head %} </head></pre>

	<pre> <body{% block body_attrbs %}{% endblock body_attrbs %}> {% block body -%} {% block navbar %} {%- endblock navbar %} {% block content -%} {%- endblock content %} {% block scripts %} <script src="{{bootstrap_find_resource('jquery.js', cdn='jquery')}}"> </script> <script src="{{bootstrap_find_resource('js/bootstrap.js', cdn='bootstrap')}}"> </script> {%- endblock scripts %} {%- endblock body %} </body> {%- endblock html %} </html> {% endblock doc -%} </pre>
templates /form.html	
	<pre> {% extends "bootstrap/base.html" %} {%- block head %} <title>表單測試頁</title> <meta charset="utf-8"> <meta http-equiv="X-UA-Compatible" content="IE=edge"> <meta name="viewport" content="width=device-width, initial-scale=1"> <!-- 上述 3 個 meta 標籤*必須*放在最前面，任何其他內容都*必須*跟隨其後！ --> {%- block styles %} <!-- Bootstrap --> <link href="{{bootstrap_find_resource('css/bootstrap.css', cdn='bootstrap')}}" rel="stylesheet"> <!-- Bootstrap core CSS --> <link href="static/css/bootstrap.min.css" rel="stylesheet"> {%- endblock styles %} {%- endblock head %} {% block body -%} <form role="form" method="POST"> <div class="form-group"> <label for="name">名稱</label> <input type="text" id="name" name="name" placeholder="請輸入名稱"> </div> <button type="submit" class="btn btn-default">提交</button> </form> {{name}} {%- endblock body %} </pre>

Class09

app.py	
	<pre>from flask import * import os from werkzeug.utils import secure_filename app = Flask(__name__) app.config['UPLOAD_FOLDER'] = './static/' @app.route("/") def index(): return render_template('index.html') @app.route('/upload', methods=['GET', 'POST']) def upload(): if request.method == 'POST': file = request.files['photo'] filename = secure_filename(file.filename) filename = file.filename file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename)) return render_template('upload.html') else: return render_template('upload.html') if __name__ == "__main__": port = int(os.environ.get('PORT', 5000)) app.run(host='127.0.0.1', port=port)</pre>

templates /upload.html	
	<pre><!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>upload</title> </head> <body> <form method=POST enctype=multipart/form-data action="{{ url_for('upload') }}"> <input type="file" name="photo"> <input type="submit"> </form> </body> </html></pre>

app.py	
	<pre> from flask import * import os from werkzeug.utils import secure_filename import firebase_admin from firebase_admin import credentials, firestore, storage app = Flask(__name__) #cred=credentials.Certificate('放憑證檔案') cred=credentials.Certificate('pcschoolline.json') #firebase_admin.initialize_app(cred, {'storageBucket': 'storage 位置'}) firebase_admin.initialize_app(cred, {'storageBucket': 'pcschoolline.appspot.com'}) #app01 = firebase_admin.initialize_app(cred, {'storageBucket': 'storage 位置'}, name='storage') app01 = firebase_admin.initialize_app(cred, {'storageBucket': 'pcschoolline.appspot.com'}, name='storage') app.config['UPLOAD_FOLDER'] = './static/' @app.route("/") def index(): return render_template('index.html') @app.route('/upload', methods=['GET', 'POST']) def upload(): if request.method == 'POST': file = request.files['photo'] filename = file.filename file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename)) bucket = storage.bucket(app=app01) blob = bucket.blob(filename) blob.upload_from_filename(os.path.join(app.config['UPLOAD_FOLDER'], filename)) return render_template('upload.html') else: return render_template('upload.html') if __name__ == "__main__": port = int(os.environ.get('PORT', 5000)) app.run(host='127.0.0.1', port=port) </pre>

templates /upload.html	<pre><!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>upload</title> </head> <body> <form method=POST enctype=multipart/form-data action="{{ url_for('upload') }}"> <input type="file" name="photo"> <input type="submit"> </form> </body> </html></pre>
---------------------------	--

app.py	<pre> from flask import * import os from werkzeug.utils import secure_filename import firebase_admin from firebase_admin import credentials, firestore, storage app = Flask(__name__) app.config['UPLOAD_FOLDER'] = './static/' cred=credentials.Certificate('pcschoolline.json') firebase_admin.initialize_app(cred, {'storageBucket': 'pcschoolline.appspot.com'}) app01 = firebase_admin.initialize_app(cred, {'storageBucket': 'pcschoolline.appspot.com'}, name='storage') @app.route("/") def index(): return render_template('index.html') @app.route('/upload', methods=['GET', 'POST']) def upload(): if request.method == 'POST': file = request.files['photo'] #filename = secure_filename(file.filename) filename = file.filename file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename)) bucket = storage.bucket(app=app01) blob = bucket.blob(filename) blob.upload_from_filename(os.path.join(app.config['UPLOAD_FOLDER'], filename)) #blob.make_public() print(blob.public_url) return redirect(url_for('upload')) else: return render_template('upload.html') if __name__ == "__main__": port = int(os.environ.get('PORT', 5000)) app.run(host='127.0.0.1', port=port) </pre>
--------	---

templates /upload.html	
	<pre><!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>upload</title> </head> <body> <form method=POST enctype=multipart/form-data action="{{ url_for('upload') }}"> <input type="file" name="photo"> <input type="submit"> </form> </body> </html></pre>

app.py	<pre> from flask import * import os from werkzeug.utils import secure_filename import firebase_admin from firebase_admin import credentials, firestore, storage import MySQLdb app = Flask(__name__) app.config['UPLOAD_FOLDER'] = './static/' cred=credentials.Certificate('pcschoolline.json') firebase_admin.initialize_app(cred, {'storageBucket': 'pcschoolline.appspot.com'}) app01 = firebase_admin.initialize_app(cred, {'storageBucket': 'pcschoolline.appspot.com'}, name='storage') pstorage="" @app.route("/") def index(): return render_template('index.html') @app.route('/upload', methods=['GET', 'POST']) def upload(): if request.method == 'POST': db = MySQLdb.connect("us-cdbr-east-06.cleardb.net", "b9d5e0986cfc58", "9e0c1166", "heroku_8002f92dacd4943", charset='utf8') cursor = db.cursor() psort = request.form.get('psort') pname = request.form.get('pname') pmoney = request.form.get('pmoney') file = request.files['pimg'] filename = file.filename filename = secure_filename(file.filename) file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename)) bucket = storage.bucket(app=app01) blob = bucket.blob(filename) blob.upload_from_filename(os.path.join(app.config['UPLOAD_FOLDER'], filename)) blob.make_public() fbimg=blob.public_url #print(fbimg) #pstorage='https://pcschooll04.herokuapp.com/static/' + filename pstorage=fbimg sql = 'INSERT INTO product(psort,pname,pmoney, pimg,pstorage) VALUES ("%s", "%s", "%s", "%s", "%s")' % (psort,pname,pmoney,filename,pstorage) </pre>
--------	---

```
try:
    cursor.execute(sql)
    db.commit()
except:
    db.rollback()
db.close()
return render_template('uploadF01.html',okk='thanks')
else:
    return render_template('uploadF01.html')
```

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

```
def getinfo():
```

```
    db = MySQLdb.connect("us-cdbr-east-06.cleardb.net", "b9d5e0986cfc58", "9e0c1166", "heroku_8002f92dacd4943", charset='utf8' )
```

```
    cursor = db.cursor()
```

```
    sql = 'SELECT * FROM product'
```

```
    try:
```

```
        cursor.execute(sql)
```

```
        U=cursor.fetchall()
```

```
        db.commit()
```

```
    except:
```

```
        db.rollback()
```

```
    db.close()
```

```
    return render_template('getinfo.html',u=U)
```

```
@app.route('/deletec',methods=['GET', 'POST'])
```

```
def delete_entry():
```

```
    if request.method=='POST':
```

```
        print(type(request.form['entry_id']))
```

```
        #MySQLdb.connect("主機", "帳號", "密碼", "default schema", charset='utf8' )
```

```
        db = MySQLdb.connect("us-cdbr-east-06.cleardb.net", "b9d5e0986cfc58", "9e0c1166", "heroku_8002f92dacd4943", charset='utf8' )
```

```
        cursor = db.cursor()
```

```
        #myid=request.POST['pid']
```

```
        myid=request.form['entry_id']
```

```
        intmid=int(myid)
```

```
        myimg=request.form['entry_img']
```

```
        print(myimg)
```

```
        cursor.execute('DELETE FROM product WHERE pid =%s',(intmid,))
```

```
        db.commit()
```

```
        bucket = storage.bucket(app=app01)
```

```
        blob_name=myimg
```

```
        blob=bucket.blob(blob_name)
```

```
        blob.delete()
```


	<pre>print('blob{}deleted.'.format(blob_name)) return redirect(url_for('getinfo')) else: return redirect(url_for('getinfo')) if __name__ == "__main__": port = int(os.environ.get('PORT', 5000)) app.run(host='0.0.0.0', port=port)</pre>
templates /upload.html	
	<pre><!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>upload</title> </head> <body> <form method=POST enctype=multipart/form-data action="{{ url_for('upload') }}"> <input type="file" name="photo"> <input type="submit"> </form> </body> </html></pre>
templates /uploadF01.html	<pre><!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>upload</title> </head> <body> <form method="post" enctype="multipart/form-data" action="{{ url_for('upload') }}"> <table class="table table-bordered"> <tbody></pre>

	<pre><tr> <th >-上傳商品-</th> </tr> <tr> <th>產品類別</th> <td><input type="text" name="psort" ></td> </tr> <tr> <tr> <th>產品名稱</th> <td><input type="text" name="pname" ></td> </tr> <tr> <th>產品金額</th> <td><input type="text" name="pmoney"></td> </tr> <tr> <th>產品圖片</th> <td><input type="file" size="30" name="pimg"/></td> </tr> <tr> <th>送出</th> <td><input type="submit" value="送出"></td> </tr> </tbody> </table> {{okk}} </form> </body> </html></pre>
templates /getinfo.html	
	<pre><!DOCTYPE html> <html> <head></pre>

```
<meta charset="utf-8">
<title>上傳</title>
</head>

<body>
  

  <table width="1200" border="1">

    </table>

    <table class="table table-striped">
      <thead>
        <tr>
          <th colspan="6" style="text-align:center" class="h1">-商品列表-</th>
        </tr>
        <tr>

          <th scope="col">類別</th>
          <th scope="col">名稱</th>
          <th scope="col">金額</th>
          <th scope="col">圖片</th>
          <th scope="col">刪除</th>

        </tr>
      </thead>
      <tbody>
        {% for i in u %}
          <tr>

            <td>{{ i[1] }}</td>
            <td>{{ i[2] }}</td>
            <td>{{ i[3] }}</td>
            <td><img src={{ i[5] }} align="middle" width="200px" /></td>
            <td>

              <form action="{{ url_for('delete_entry') }}" method=POST class=delete-entry>
                <input type="hidden" name="entry_id" value='{{ i[0] }}'>
                <input type="hidden" name="entry_img" value='{{ i[4] }}'>
                <input type="submit" value="Delete" />

            </td>
          </tr>
        {% endfor %}
      </tbody>
    </table>
  </body>
</html>
```

	<pre></form> </td> </tr> {% endfor %} </tbody> </table> </body> </html></pre>
templates /index.html	
	<pre><!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>上傳</title> </head> <body> upload or getinfo </body> </html></pre>
static	放圖片位置
Procfile	web: python app.py
requirements.txt	<pre>Flask==1.0.2 requests==2.21.0 line-bot-sdk dialogflow==0.4.0 apiai==1.2.3 jsongatabase==0.1.7 numpy==1.13.3 urllib3==1.22 python-dotenv==0.8.2 pusher==2.0.1 gspread oauth2client</pre>

	mysqlclient pymysql gunicorn==19.6.0 firebase_admin Pillow Flask-Login flask-session psycopg2 Flask-Migrate Flask-Script flask_uploads flask-wtf Werkzeug
--	---

MySQL Workbench

pcschoollupload - Warning - n... x

File Edit View Query Database Server Tools Scripting Help

Navigator: Filter objects

SCHEMAS

- heroku_8002f92dacd4943
 - Tables
 - product
 - Columns
 - pid
 - psort
 - pname
 - pmoney
 - pimg
 - pstorage
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions

product heroku_8002f92dacd4943 produ... x

Info Columns Indexes Triggers Foreign keys Partitions Grants DDL

Column	Type	D	Nullable	Chara...	Collation	Privileges	Extra	Comments
pid	int(11)		NO			select,insert,update,references	auto_increment	
psort	varchar(10)		YES	utf8	utf8_general_ci	select,insert,update,references		
pname	varchar(50)		YES	utf8	utf8_general_ci	select,insert,update,references		
pmoney	int(11)		YES			select,insert,update,references		
pimg	varchar(100)		YES	utf8	utf8_general_ci	select,insert,update,references		
pstorage	varchar(200)		YES	utf8	utf8_general_ci	select,insert,update,references		

MySQL Workbench

pcschoollupload - Warning - n... x

File Edit View Query Database Server Tools Scripting Help

Navigator: Filter objects

SCHEMAS

- heroku_8002f92dacd4943
 - Tables
 - product
 - Columns
 - pid
 - psort
 - pname
 - pmoney
 - pimg
 - pstorage
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions

product heroku_8002f92dacd4943 produ... product x

Limit to 1000 rows

1 • `SELECT * FROM heroku_8002f92dacd4943.product;`

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	pid	psort	pname	pmoney	pimg	pstorage
▶	81	ee	apple	500	christmas.jpg	https://storage.googleapis.com/pcschoolline.appspot.com/christmas.jpg
	101	洋装	ee	1000	bba0420.jpg	https://storage.googleapis.com/pcschoolline.appspot.com/bba0420.jpg