FLASK & SQLITE

張家瑋 博士

副教授

國立臺中科技大學資訊工程系

FLASK & SQLITE

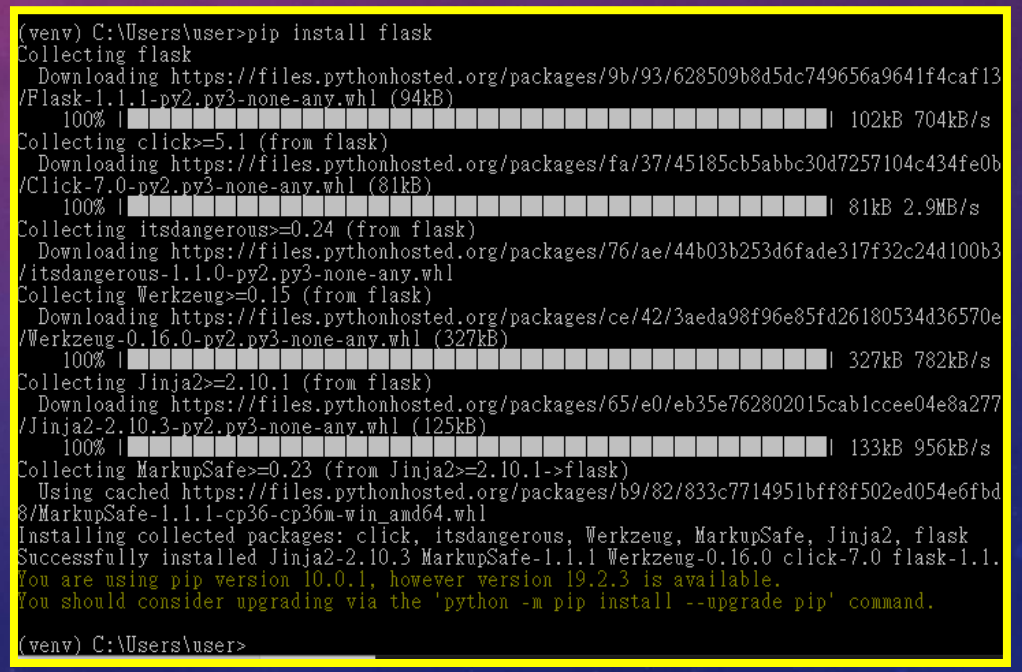
SQLITE

- Python 有許多能用於創建 Web 應用程式和 Web API 的框架，而輕量的 Flask 框架可以勝任Web API 的需求：管理 HTTP 請求和顯示資料內容。

- flask-restful 是針對 restful api 開發的一個flask的套件，建構在 flask 的輕薄短小的基礎下，flask-restful 可以在短短幾行內完成 restful api的開發。

虛擬環境設定

- 前提在 Python 3.6 以上版本，使用內建的 python3-venv 套件。
  - python -m venv venv (虛擬環境名稱)
- Windows
  - .\venv\Scripts\activate.bat
- Linux/macOS
  - source ./venv/bin/activate



安裝所需模組

1. pip install flask
2. pip freeze　#觀看虛擬環境下已安裝的模組

```
(venv) C:\Users\user>pip freeze
aniso8601==8.0.0
Click==7.0
Flask==1.1.1
Flask-RESTful==0.3.7
itsdangerous==1.1.0
Jinja2==2.10.3
MarkupSafe==1.1.1
pytz==2019.3
six==1.12.0
Werkzeug==0.16.0
```

GET DATA FROM CSV

gapminder.csv

	A	B	C	D	E	F
1	country	continent	year	lifeExp	pop	gdpPercap
2	Afghanista	Asia	1952	28.801	8425333	779.4453
3	Afghanista	Asia	1957	30.332	9240934	820.853
4	Afghanista	Asia	1962	31.997	10267083	853.1007
5	Afghanista	Asia	1967	34.02	11537966	836.1971
6	Afghanista	Asia	1972	36.088	13079460	739.9811
7	Afghanista	Asia	1977	38.438	14880372	786.1134
8	Afghanista	Asia	1982	39.854	12881816	978.0114
9	Afghanista	Asia	1987	40.822	13867957	852.3959
10	Afghanista	Asia	1992	41.674	16317921	649.3414
11	Afghanista	Asia	1997	41.763	22227415	635.3414
12	Afghanista	Asia	2002	42.129	25268405	726.7341
13	Afghanista	Asia	2007	43.828	31889923	974.5803
14	Albania	Europe	1952	55.23	1282697	1601.056
15	Albania	Europe	1957	59.28	1476505	1942.284
16	Albania	Europe	1962	64.82	1728137	2312.889
17	Albania	Europe	1967	66.22	1984060	2760.197
18	Albania	Europe	1972	67.69	2263554	3313.422
19	Albania	Europe	1977	68.93	2509048	3533.004
20	Albania	Europe	1982	70.42	2780097	3630.881

GET DATA FROM CSV

```python
import flask
from flask import jsonify, request
import numpy as np
import pandas as pd

app = flask.Flask(__name__)
app.config["DEBUG"] = True # True 表示開啟除錯模式，正式對外運行時需註解掉
app.config["JSON_AS_ASCII"] = False # False 表示不編譯為 ASCII

gapminder = pd.read_csv("gapminder.csv")
gapminder_list = []
nrows = gapminder.shape[0]
for i in range(nrows):
    ser = gapminder.loc[i, :]
    row_dict = {}
    for idx, val in zip(ser.index, ser.values):
        if type(val) is str:
            row_dict[idx] = val
        elif type(val) is np.int64:
            row_dict[idx] = int(val)
        elif type(val) is np.float64:
            row_dict[idx] = float(val)
    gapminder_list.append(row_dict)
```

```python
@app.route('/gapminder/all', methods=['GET'])
def gapminder_all():
    return jsonify(gapminder_list)


@app.route('/gapminder', methods=['GET'])
def country():
    if 'country' in request.args:
        country = request.args['country']
    else:
        return "Error: No country provided. Please specify a country."
    results = []

    for elem in gapminder_list:
        if elem['country'] == country:
            results.append(elem)

    return jsonify(results)


app.run()
```

創建SQLITE資料庫與資料表

```python
import sqlite3

conn = sqlite3.connect('user.db')

cursor = conn.cursor()
cursor.execute('DROP TABLE IF EXISTS users')
cursor.execute('CREATE TABLE IF NOT EXISTS users('
               'id INTEGER PRIMARY KEY, '
               'name TEXT, '
               'email TEXT, '
               'password TEXT)')

conn.commit()
conn.close()
```

SQLITE 新增資料

```python
import sqlite3

conn = sqlite3.connect('user.db')
cursor = conn.cursor()
insert_query = 'INSERT INTO users VALUES(?, ?, ?, ?)'

users = []

users.append((None, 'Gary', 'gary@gmail.com', '123456'))
users.append((None, 'Jason', 'jason@gmail.com', '123456'))
users.append((None, 'Anita', 'anita@gmail.com', '123456'))

cursor.executemany(insert_query, users)

conn.commit()
conn.close()
```

SQLITE 更新資料

```python
import sqlite3

conn = sqlite3.connect('user.db')
cursor = conn.cursor()
update_query = 'UPDATE users SET name=?, email=?, password=? WHERE id=?'
cursor.execute(update_query, (name, email, password, uid))
conn.commit()
conn.close()
```

SQLITE 查詢資料

```python
import sqlite3

conn = sqlite3.connect('user.db')
cursor = conn.cursor()

for row in cursor.execute('SELECT * FROM users'):
    print(row)

conn.commit()
conn.close()
```

SQLITE 刪除資料

```python
import sqlite3

conn = sqlite3.connect('user.db')
cursor = conn.cursor()
delete_query = 'DELETE FROM users WHERE id=?'
cursor.execute(delete_query, (id,))
conn.commit()
conn.close()
```

安裝所需模組

1. pip install flask
2. pip install flask-cors

```
from flask_cors import *
CORS(app, resources=r'/*')
```

RESTFUL SQLITE

```
1  import flask
2  from flask_cors import *
3  from flask import request
4  from flask import jsonify
5  import sqlite3
6
7
8  def add_user(name, email, password):⋯
16
17 def update_user(uid, name, email, password):⋯
25
26
27 def delete_user(id):⋯
37
38
39 def get_user(name):⋯
50
51
52 def get_all_user():⋯
62
63
64 app = flask.Flask(__name__)
65 CORS(app, resources=r'/*')
66 app.config["DEBUG"] = True
```

RESTFUL SQLITE

```
69  @app.route('/', methods=['GET', 'POST'])
70  def home():
71      return "<h1>Hello Flask!</h1>"
72
73
74  @app.route('/users/all', methods=['GET', 'POST'])
75  def getAllUsers():
76      return get_all_user()
77
78  @app.route('/user', methods=['GET', 'POST'])
79  def getUser():...
85
86  @app.route('/remove', methods=['GET', 'POST'])
87  def removeUser():...
93
94  @app.route('/add', methods=['GET', 'POST'])
95  def addUser():...
103
104 @app.route('/update', methods=['GET', 'POST'])
105 def updateUser():...
114
115
116 app.run()
```

RESTFUL SQLITE – ADD USER

```python
def add_user(name, email, password):
    conn = sqlite3.connect('user.db')
    cursor = conn.cursor()
    insert_query = 'INSERT INTO users VALUES(?, ?, ?, ?)'
    cursor.execute(insert_query, (None, name, email, password))
    conn.commit()
    conn.close()
    return "Add the user successfully!"
```

```python
@app.route('/add', methods=['GET', 'POST'])
def addUser():
    if request.method == 'POST' or request.method == 'GET':
        name = request.values['name']
        email = request.values['email']
        password = request.values['password']
        return add_user(name, email, password)
    else:
        return "Error: No Data provided. Please specify a User Data."
```

RESTFUL SQLITE – UPDATE USER

```python
def update_user(uid, name, email, password):
    conn = sqlite3.connect('user.db')
    cursor = conn.cursor()
    update_query = 'UPDATE users SET name=?, email=?, password=? WHERE id=?'
    cursor.execute(update_query, (name, email, password, uid))
    conn.commit()
    conn.close()
    return "Update the user successfully!"
```

```python
@app.route('/update', methods=['GET', 'POST'])
def updateUser():
    if request.method == 'POST' or request.method == 'GET':
        uid = request.values['uid']
        name = request.values['name']
        email = request.values['email']
        password = request.values['password']
        return update_user(int(uid), name, email, password)
    else:
        return "Error: No Data provided. Please specify a User Data."
```

RESTFUL SQLITE – GET ALL USERS

```
def get_all_user():
    users = {}
    conn = sqlite3.connect('user.db')
    cursor = conn.cursor()
    query_one_query = 'SELECT * FROM users'
    for item in cursor.execute(query_one_query):
        user = {'name': item[1], 'email': item[2], 'pwd': item[3]};
        users.update({item[0]:user})
    conn.close()
    return users
```

```
@app.route('/users/all', methods=['GET', 'POST'])
def getAllUsers():
    return get_all_user()
```

THANK YOU