

Lonely Caesar



# lineBot 氣象小助理

作者 吳彥瑾

|   |
|---|
| 作品名稱：lineBot 氣象小助理  |
| 一、說明  |
| <p>本專題使用 anaconda3.9 以上版本、網路爬蟲(爬取)、linebot 開發流程及預設功能提供地址製作出一個 lineBot 氣象小助理，串接雷達回波與地震資訊、目前氣象資訊、天氣預報和空氣品質快速問答說明功能。</p> <p>套件安裝準備：</p> <ol style="list-style-type: none"> <li>1. 請先註冊並登入 LINE Developer 並建立 Line Channel</li> <li>2. 請先下載 ngrok.exe 放置於桌面上，點開後輸入 ngrok http 5000 就會產生網址將貼在建立好的 api 的 Webhook URL 上然後測試。</li> <li>3. 輸入 flask --app 自己的檔名 --debug run。</li> <li>4. 輸入 pip install flask-ngrok 指令安裝套件。</li> <li>5. 輸入 pip install line-bot-sdk 指令安裝套件。</li> <li>6. 氣象資料開放平台 <a href="https://opendata.cwa.gov.tw/index">https://opendata.cwa.gov.tw/index</a>。</li> </ol> |
| 二、相關文章  |
| <p>透過 Messaging API，機器人伺服器可以向 LINE 平台發送資料並從 LINE 平台接收資料。請求以 JSON 格式透過 HTTPS 傳送。Bot 伺服器與 LINE 平台之間的通訊流程如下：</p> <ol style="list-style-type: none"> <li>1. 用戶向 LINE 官方帳戶發送訊息。</li> <li>2. LINE 平台將 webhook 事件傳送到機器人伺服器的 webhook URL。</li> <li>3. 機器人伺服器檢查 webhook 事件並透過 LINE 平台回應用戶。</li> </ol> <p>參考網址：</p> <p><a href="https://developers.line.biz/en/docs/messaging-api/overview/">https://developers.line.biz/en/docs/messaging-api/overview/</a></p>    |
| 三、實作  |
| <p>LineBot 氣象小助理完整 demo：</p> <p>透過氣象資料開放平台的 json 爬取要的資料、開啟 ngrok 取得網址、授權碼貼在 code 上面及 LINE Developer 的 key 作串接並能夠完整的執行。</p> <p>程式碼：</p> <pre>from flask_ngrok import run_with_ngrok</pre>  |

```

from flask import Flask, request
# 載入 json 標準函式庫，處理回傳的資料格式
import requests, json, time, statistics # import statistics 函式庫
# 載入 LINE Message API 相關函式庫
from linebot import LineBotApi, WebhookHandler
from linebot.models import MessageEvent, TextMessage, TextSendMessage

# 地震資訊函式
def earth_quake():
    msg = ['找不到地震資訊', 'https://example.com/demo.jpg'] # 預設回傳的訊息
    try:
        code = '你的氣象資料授權碼'
        url = f'https://opendata.cwa.gov.tw/api/v1/rest/datastore/E-A0016-001?Authorization={code}'
        e_data = requests.get(url) # 爬取地震資訊網址
        e_data_json = e_data.json() # json 格式化訊息內容
        eq = e_data_json['records']['Earthquake'] # 取出地震資訊
        for i in eq:
            loc = i['EarthquakeInfo']['Epicenter']['Location'] # 地震地點
            val = i['EarthquakeInfo']['EarthquakeMagnitude']['MagnitudeValue'] # 芮氏規模
            dep = i['EarthquakeInfo']['FocalDepth'] # 地震深度
            eq_time = i['EarthquakeInfo']['OriginTime'] # 地震時間
            img = i['ReportImageURI'] # 地震圖
            msg = [f'{loc}，芮氏規模 {val} 級，深度 {dep} 公里，發生時間 {eq_time}。', img]
            break # 取出第一筆資料後就 break
        return msg # 回傳 msg
    except:
        return msg # 如果取資料有發生錯誤，直接回傳 msg

# 目前天氣函式
def get_data(url, address):
    w_data = requests.get(url) # 爬取目前天氣網址的資料
    w_data_json = w_data.json() # json 格式化訊息內容
    location = w_data_json['cwaopendata']['dataset']['Station'] # 取出對應地點的內容
    for i in location:

```

```

name = i['StationName'] # 測站地點
city = i['GeoInfo']['CountyName'] # 縣市名稱
area = i['GeoInfo']['TownName'].strip() # 鄉鎮行政區
temp = i['WeatherElement']['AirTemperature'] # 氣溫
humd = round(float(i['WeatherElement']['RelativeHumidity'])*1,1)# 相對濕
度

r24 = i['WeatherElement']['Now']['Precipitation'] # 累積雨量
print(city, area, name, f'{temp} 度', f'相對濕度 {humd}%', f'累積雨量
{r24}mm')

if city in address and area in address: # 如果使用者的地址包含縣市名稱
    msg = f'氣溫 {temp} 度，相對濕度 {humd}%，累積雨量 {r24}mm'
    break

return msg

def current_weather(address):
    msg = '找不到氣象資訊。' # 預設回傳訊息
    code = '你的氣象資料授權碼'
    url = f'https://opendata.cwa.gov.tw/fileapi/v1/opendataapi/O-A0001-
001?Authorization={code}&downloadType=WEB&format=JSON'
    url = f'https://opendata.cwa.gov.tw/fileapi/v1/opendataapi/O-A0003-
001?Authorization={code}&downloadType=WEB&format=JSON'
    msg = get_data(url, address)
    return msg

# 空氣品質函式
def aqi(address):
    city_list, site_list = {}, {}
    msg = '找不到空氣品質資訊。'
    try:
        # 2022/12 時氣象局有修改了 API 內容，將部份大小寫混合全改成小
        # 寫，因此程式碼也跟著修正
        url = 'https://data.moenv.gov.tw/api/v2/aqx_p_432?api_key=e8dd42e6-9b8b-
43f8-991e-b3dee723a52d&limit=1000&sort=ImportDate%20desc&format=JSON'
        a_data = requests.get(url) # 使用 get 方法透過空氣品質指
        # 標 API 取得內容
        a_data_json = a_data.json() # json 格式化訊息內容
        for i in a_data_json['records']: # 依序取出 records 內容的每個項目
            city = i['county'] # 取出縣市名稱
            if city not in city_list:

```



```

city_list[city]=[] # 以縣市名稱為 key，準備存入串
列資料
site = i['sitename'] # 取出鄉鎮區域名稱
aqi = int(i['aqi']) # 取得 AQI 數值
status = i['status'] # 取得空氣品質狀態
site_list[site] = {'aqi':aqi, 'status':status} # 記錄鄉鎮區域空氣品質
city_list[city].append(aqi) # 將各個縣市裡的鄉鎮區域空氣
aqi 數值，以串列方式放入縣市名稱的變數裡
for i in city_list:
    if i in address: # 如果地址裡包含縣市名稱的 key，就直接使用對應的
內容
        #
https://airtw.epa.gov.tw/cht/Information/Standard/AirQualityIndicator.aspx
        aqi_val = round(statistics.mean(city_list[i]),0) # 計算平均數值，
如果找不到鄉鎮區域，就使用縣市的平均值
        aqi_status = " # 手動判斷對應的空氣品質說明文字
        if aqi_val<=50: aqi_status = '良好'
        elif aqi_val>50 and aqi_val<=100: aqi_status = '普通'
        elif aqi_val>100 and aqi_val<=150: aqi_status = '對敏感族群不健康
        ,
        elif aqi_val>150 and aqi_val<=200: aqi_status = '對所有族群不健康
        ,
        elif aqi_val>200 and aqi_val<=300: aqi_status = '非常不健康'
        else: aqi_status = '危害'
        msg = f'空氣品質{aqi_status} ( AQI {aqi_val} ) 。' # 定義回傳的訊
息
        break
    for i in site_list:
        if i in address: # 如果地址裡包含鄉鎮區域名稱的 key，就直接使用
對應的內容
            msg = f'空氣品質{site_list[i]["status"]} ( AQI {site_list[i]["aqi"]} ) 。'
            break
    return msg # 回傳 msg
except:
    return msg # 如果取資料有發生錯誤，直接回傳 msg

# 氣象預報函式
def forecast(address):

```

```

area_list = {}
# 將主要縣市個別的 JSON 代碼列出
json_api = {"宜蘭縣": "F-D0047-001", "桃園市": "F-D0047-005", "新竹縣": "F-D0047-009", "苗栗縣": "F-D0047-013",
            "彰化縣": "F-D0047-017", "南投縣": "F-D0047-021", "雲林縣": "F-D0047-025", "嘉義縣": "F-D0047-029",
            "屏東縣": "F-D0047-033", "臺東縣": "F-D0047-037", "花蓮縣": "F-D0047-041", "澎湖縣": "F-D0047-045",
            "基隆市": "F-D0047-049", "新竹市": "F-D0047-053", "嘉義市": "F-D0047-057", "臺北市": "F-D0047-061",
            "高雄市": "F-D0047-065", "新北市": "F-D0047-069", "臺中市": "F-D0047-073", "臺南市": "F-D0047-077",
            "連江縣": "F-D0047-081", "金門縣": "F-D0047-085"}
msg = '找不到天氣預報資訊。' # 預設回傳訊息
try:
    code = '你的氣象資料授權碼'
    url = f'https://opendata.cwa.gov.tw/fileapi/v1/opendataapi/F-C0032-001?Authorization={code}&downloadType=WEB&format=JSON'
    f_data = requests.get(url) # 取得主要縣市預報資料
    f_data_json = f_data.json() # json 格式化訊息內容
    location = f_data_json['cwaopendata']['dataset']['location'] # 取得縣市的預報內容
    for i in location:
        city = i['locationName'] # 縣市名稱
        wx8 = i['weatherElement'][0]['time'][0]['parameter']['parameterName']
# 天氣現象
        maxt8 = i['weatherElement'][1]['time'][0]['parameter']['parameterName'] # 最高溫
        mint8 = i['weatherElement'][2]['time'][0]['parameter']['parameterName']
# 最低溫
        ci8 = i['weatherElement'][3]['time'][0]['parameter']['parameterName']
# 舒適度
        pop8 = i['weatherElement'][4]['time'][0]['parameter']['parameterName']
# 降雨機率
        area_list[city] = f'未來 8 小時{wx8}，最高溫 {maxt8} 度，最低溫 {mint8} 度，舒適度{ci8}，降雨機率 {pop8} %' # 組合成回傳的訊息，存在以縣市名稱為 key 的字典檔裡
    for i in area_list:
        if i in address: # 如果使用者的地址包含縣市名稱

```

```

        msg = area_list[i] # 將 msg 換成對應的預報資訊
        break

    return msg # 回傳 msg
except:
    return msg # 如果取資料有發生錯誤，直接回傳 msg

# LINE push 訊息函式
def push_message(msg, uid, token):
    headers = {'Authorization': f'Bearer {token}', 'Content-Type': 'application/json'}
    body = {'to': uid, 'messages': [{"type": "text", "text": msg}]}
    req = requests.request('POST', 'https://api.line.me/v2/bot/message/push',
headers=headers, data=json.dumps(body).encode('utf-8'))
    print(req.text)

# LINE 回傳訊息函式
def reply_message(msg, rk, token):
    headers = {'Authorization': f'Bearer {token}', 'Content-Type': 'application/json'}
    body = {'replyToken': rk, 'messages': [{"type": "text", "text": msg}]}
    req = requests.request('POST', 'https://api.line.me/v2/bot/message/reply',
headers=headers, data=json.dumps(body).encode('utf-8'))
    print(req.text)

# LINE 回傳圖片函式(雷達回波圖)
def reply_image(msg, rk, token):
    headers = {'Authorization': f'Bearer {token}', 'Content-Type': 'application/json'}
    body = {'replyToken': rk, 'messages': [{"type": 'image', 'originalContentUrl':
msg, 'previewImageUrl': msg}]}
    req = requests.request('POST', 'https://api.line.me/v2/bot/message/reply',
headers=headers, data=json.dumps(body).encode('utf-8'))
    print(req.text)

app = Flask(__name__)

@app.route("/", methods=['POST'])
def linebot():
    body = request.get_data(as_text=True) # 取得收到的訊息內容
    try:

```

```

access_token = '你的 LINE Channel access token'
secret = '你的 LINE Channel secret'
line_bot_api = LineBotApi(access_token)
handler = WebhookHandler(secret)
signature = request.headers['X-Line-Signature'] # 加入回傳的 headers
handler.handle(body, signature) # 綁定訊息回傳的相關資訊
json_data = json.loads(body) # 轉換內容為 json 格式
reply_token = json_data['events'][0]['replyToken'] # 取得回傳訊息的
Token ( reply message 使用 )
user_id = json_data['events'][0]['source']['userId'] # 取得使用者 ID ( push
message 使用 )
print(json_data) # 印出內容
if 'message' in json_data['events'][0]:
    if json_data['events'][0]['message']['type'] == 'location':
        address = json_data['events'][0]['message']['address'].replace('台','
臺')

reply_message(f'{address}\n\n{current_weather(address)}\n\n{aqi(address)}\n\n{forec
ast(address)}',reply_token, access_token) # 回覆爬取到的相關氣象資訊
        print(address)
    if json_data['events'][0]['message']['type'] == 'text': # 如果 message
的類型是文字 text
        text = json_data['events'][0]['message']['text'] # 取出文字
        if text == '雷達回波圖' or text == '雷達回波': # 如果是雷達回波
圖相關的文字
            reply_image(f'https://cwaopendata.s3.ap-northeast-
1.amazonaws.com/Observation/O-A0058-001.png?{time.time_ns()}',reply_token,
access_token)
        elif text == '地震資訊' or text == '地震': # 如果是地震相關的文
字
            msg = earth_quake() # 爬取地震資訊
            push_message(msg[0], user_id, access_token) # 傳送地震資
訊 ( 用 push 方法, 因為 reply 只能用一次 )
            reply_image(msg[1], reply_token, access_token) # 傳送地震
圖片 ( 用 reply 方法 )
        else:
            reply_message(text, reply_token, access_token) # 如果是一
般文字, 直接回覆同樣的文字

```



```
except Exception as e:  
    print('error', e)  
    return 'OK'
```

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

執行結果：

### 1. 雷達回波圖



### 2. 地震資訊




### 3. 天氣、預報及空氣品質



### 四、結論

完成後，在氣象機器人的聊天畫面下方，就會出現圖文選單，點擊圖文選單，就會出現對應的動作。



## 五、參考

(STEAM 教育學習網)

<https://steam.oxxostudio.tw/category/python/example/line-bot-weather-1.html>

