



網路程式設計

網路應用服務

Instructor: 馬豪尚

網路應用

- › 網路應用程式就是一種n層的架構，最常見的是三層應用程式
- › 第一層為呈現
 - 如果呈現的是以網頁的方式，可以搭配使用一些動態的網頁內容技術（如ASP、JS、PHP、Python）
 - 圖形使用者介面，讓使用者方便操作
- › 第二層為中間層
 - 應用程式邏輯，接收使用者輸入之後，要如何做運算或從資料庫取出資料等
 - 網路爬蟲也是中間層的例子之一
- › 第三層為儲存層
 - 連接到資料庫儲存資料

LINE BOT 互動機器人應用 實作

- › 使用Flask和 ngrok 服務
 - 創建一個網路應用程式的伺服器端
 - 如果使用 Colab 須使用Flask_ngrok
- › 建立 LINE Channel
 - 申請LINE的開發者帳號
 - 建立自訂的服務頻道
- › 撰寫互動機器人應用程式
 - 建立並串接 Webhook



建立一個LINE的Channel

LINE BOT 創建頻道

- › 註冊LINE developer帳號
 - <https://developers.line.biz/zh-hant/>
- › Create a New Provider

Welcome to the LINE Developers Console!

Here, you'll find information and tools for connecting your technology to the LINE Platform, helping you build apps that connect people.

To get started, create a provider to represent your company, yourself, or another entity that provides the apps and services you're working on.



Start by creating a provider



Then, create a channel for each app you want to connect to the LINE Platform



Lastly, enter the necessary information to link your apps to your channels, and start building!

Create a new provider

LINE BOT 創建頻道

› Create a channel

Channels

Roles

Settings

This provider doesn't have any channels yet

To create one, choose a channel type below



Create a LINE Login channel



Create a Messaging API channel



Create a Blockchain Service channel



Create a LINE MINI App channel

Available features might defer based on the account with which you are currently logged in. For details, check the [document](#).

LINE BOT 創建頻道

- › 建立時需要輸入頻道的名稱和描述，以及使用下拉選擇所在的地區與類別

Create a new channel

Channel type

Messaging API

✓ Don't leave this empty

Provider

Professor Allen

✓ Don't leave this empty

Company or
owner's country or
region ?

Taiwan

Corporations should select their company's country or region. Individuals should select the country or region.

✓ Don't leave this empty

LINE BOT 頻道管理

- › 建立 Channel 後，前往「LINE 官方帳號管理頁面」
 - 可察看目前這個帳號的基本訊息、權限、回應設定、收費資訊等

回應功能



可透過聊天與好友互動。

[開啟聊天畫面](#)

加入好友的歡迎訊息



當用戶將本帳號加為好友時，可自動傳送訊息內容。

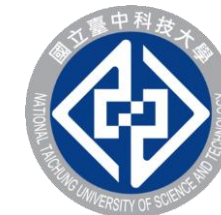
[開啟「加入好友的歡迎訊息」設定畫面](#)

Webhook



當用戶傳送訊息給本帳號或將本帳號加為好友時，從LINE平台傳送Webhook事件至Webhook網址。

[開啟Messaging API的設定畫面](#)



LINE BOT 頻道管理

› AI自動回覆聊天功能

AI自動回應訊息（快問快答）

[① 使用說明](#)[編輯](#)

收到用戶的訊息時，AI人工智慧技術將會判別內容並選擇合適的訊息來回覆。

已開啟本功能。若希望關閉本功能，請至「回應設定」畫面將「回應方式」變更為不含「AI自動回應訊息」的選項。[變更設定](#)

[一般問題](#)[基本資訊](#)[特色資訊](#)[預約資訊](#)

類型	訊息	預覽
歡迎	 謝謝您傳訊息給IP_Chat帳號！本系統可以自動回答關於營業時間、店家推薦、預...	預覽
說明	 本系統可以自動回覆一般基本疑問。若是稍微複雜的疑問，則會由客服人員...	預覽
感謝		預覽

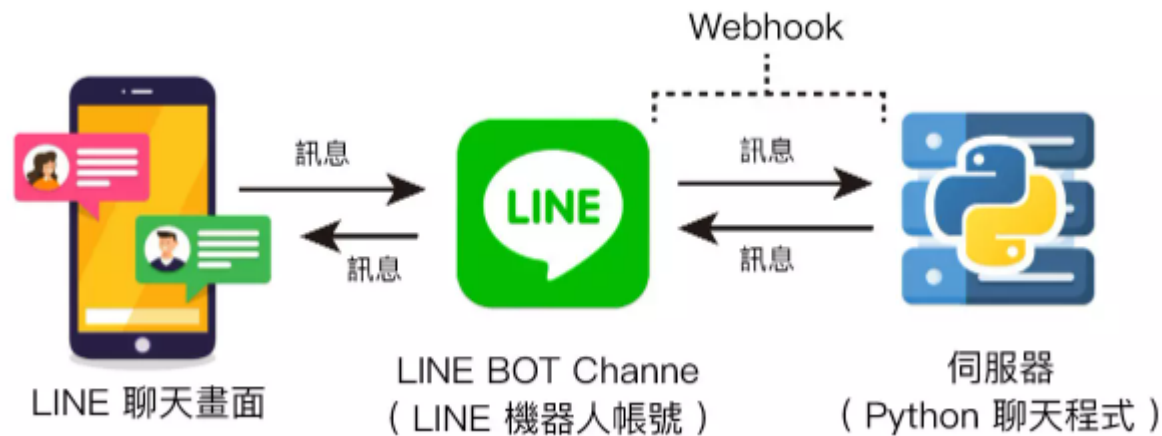
撰寫互動機器人應用程式

LINE BOT 與 WebHook

- › 因為LINE並不能提供給使用者上傳自己的程式碼來處理
- › 因此透過Webhook的串接，使得LINE BOT透過一個網址去和伺服器做溝通，向伺服器傳送或接收特定的資料，這個網址就是連結到伺服器上的某個程式
- › 讓 LINE 聊天的訊息能夠透過 Webhook 傳遞到用 Python或其他語言撰寫的程式去做處理

LINE BOT 與 WebHook 的溝通過程

- › 使用者向使用 Message API 所建立的 LINE BOT 發送訊息
- › 這個訊息透過 Webhook 傳遞到頻道管理者所設定連結的 伺服器上
- › 伺服器透過一個程式去處理訊息
- › 透過 Webhook 將處理結果回傳到 LINE BOT
- › LINE BOT 發送訊息到 LINE 聊天室裡



LINE 串接WEBHOOK

- › 在LINE Developers網站的頻道資訊內
- › Channel access token
 - Messaging API settings下
- › Channel secret
 - Basic settings下

LINE Official Account features

Edit the message text and other settings for these features in the LINE Official Account Manager

Allow bot to join
group chats ⓘ Disabled

Auto-reply
messages ⓘ Enabled

Greeting messages ⓘ Enabled

Channel access token

Channel access token (long-lived) ⓘ

LINE with WEBHOOK

- › 開啟一個用Flask建置的伺服器
- › 使用ngrok服務將伺服器位置公佈到網際網路上
- › 將伺服器網址儲存到LINE BOT的頻道WEBHOOK資訊
- › 在伺服器上使用LINE BOT API和WEBHOOK服務，並將兩項服務做連結
 - 接收從聊天室傳來的訊息(HTTP POST)，解析完並處理完之後用WEBHOOK傳遞給LINE BOT API
 - LINE BOT API將訊息回傳給聊天室

Flask建立一個網頁服務

- › 建立一個Flask 物件
 - `app = Flask(__name__)`
 - `__name__`表示目前執行的程式
- › 建立一個路由 (Routes)，定義主網域 / 請求資源的url路徑
 - `@app.route("/", methods)`
 - `"/"` 代表請求資源的url路徑，沒有指定代表主網域的路徑
 - `methods`為指定請求操作資源的方式(`get`、`post`等)

Flask建立一個網頁服務

- › 定義發出請求後執行的函式
 - def function():
 - 在函式需要中指定回傳給使用者的內容
 - › return 內容
- › 執行這個flask物件 → 啟動網路服務
 - app.run()
 - 設定連線埠號 port
 - › app.run(host="0.0.0.0", port=5555)
 - › 預設是127.0.0.1, port=5000

儲存LINE BOT的傳送訊息位置

› 在LINE Developers網站的頻道下的Messaging API

LINE Developers [News](#) [Products](#) [Documentation](#) [FAQ](#) [Glossary](#) [Community](#) [Blog](#)

Console home

Providers

Admin

Professor Allen

TOP > Professor Allen > IP_Chat > Messaging API

Scan this QR code with LINE to add your LINE Official Account as a friend

Webhook settings

Webhook URL [?](#)

☒ Don't leave this empty

☒ Enter a valid HTTPS URL

☒ Enter no more than 500 characters

Update **Cancel**

使用LINE API和WEBHOOK服務

- › 宣告LINE BOT 物件
 - line_bot_api = LineBotApi(access_token)
 - › access_token為LINE的Channel access token
- › 宣告WEBHOOK物件
 - handler = WebhookHandler(secret)
 - › secret為LINE的Channel secret

LINE BOT 應用的路由設定

- › `@app.route("/", methods=['POST'])`
 - LINE BOT 是透過POST 方法將資料放在 message-body 進行傳送，無法單純透過網址列傳送
 - 第一個參數對應到LINE WEBHOOK的欄位，"/"，代表LINE BOT 綁定的WEBHOOK為伺服器的網址

Example

```
@app.route("/", methods=['POST'])  
def linebot():  
    task...
```

取得LINE BOT傳來的訊息

- › 取得LINE聊天室傳來的訊息，取回來是一個類似json格式的形式
 - `request.get_data(as_text=True)`

Example

```
body = request.get_data(as_text=True)
```

```
json_data = json.loads(body)
```

```
msg = json_data['events'][0]['message']['text']
```

取得LINE BOT傳來的訊息

- › 首先要先驗證訊息來源，為了確認該事件訊息是否真的來自LINE平台，避免有安全性問題
- › 驗證方式
 - 檢查所收到HTTP請求標頭(Header)中的數位簽章
 - 驗證header中必須要包含X-Line-Signature項目，該項目的內容值是即為數位簽章
 - › `signature = request.headers['X-Line-Signature']`
- › 驗證完就交由WEBHOOK的handler去處理
 - `handler.handle(body, signature)`

WEBHOOK Handler設定

- › @handler.add(MessageEvent, message=TextMessage)
- › Handler加入一個事件處理的設定
 - 當收到 LINE 的某一種事件，(MessageEvent)，且訊息是屬於 TextMessage 的時候，就執行以下程式
 - LINE 的事件包括有：MessageEvent (訊息事件)、FollowEvent (加好友事件)、UnfollowEvent (刪好友事件)、JoinEvent (加入聊天室事件)、LeaveEvent (離開聊天室事件)、MemberJoinedEvent (加入群組事件)、MemberLeftEvent (離開群組事件)等
 - 其中訊息事件又包含TextMessage、ImageMessage、VideoMessage、StickerMessage、FileMessage等

WEBHOOK Handler設定

› 定義要執行的函式

–def handle_message(event):

- › event即為傳入handler的發生事件，聊天室傳來的訊息也包含在裡面
- › 我們在這個函式裡面做所有要處理的事情

```
event = {"reply_token":"就是代表reply_token",  
        "type":"message",  
        "timestamp":"1462629479859",  
        "source":{"type":"user",  
                  "user_id":"就是代表user的一串亂碼"},  
        "message":{"id":"就是代表這次message的一串代碼",  
                  "type":"text",  
                  "text":"使用者傳來的文字信息內容"}}
```


LINE BOT回傳訊息給聊天室

- › 事情都處理完之後用LINE BOT API傳訊息
 - `line_bot_api.reply_message(reply_token, TextSendMessage(text='回傳給聊天室的訊息'))`
 - › `reply_token`為event中的`reply_token`，要用這個值才能回傳到原本的聊天室，因為有可能很多個使用者在聊天
 - › `TextSendMessage`裡面設定要回傳的訊息

WEBHOOK串接Google Map API

- › 使用者在聊天室輸入關鍵字
 - 將文字訊息傳到伺服器
- › 定義WEBHOOK handler發生使用者傳訊息事件
 - 載入google map api
 - 用使用者的文字訊息當關鍵字查詢
 - 解析查詢結果，取得需要的資訊
- › 用LINE BOT API將結果回傳給使用者顯示在聊天室




Google Cloud Function


移植到Google Cloud Function

- › 因為我們沒有伺服器
 - 用Flask+ngrok來架設虛擬伺服器
- › 因為Colab免費使用而有些限制
 - 固定時間會斷線
- › Google Cloud Functions 是一個無伺服器的雲端執行環境，可以架設一些簡單或單一用途的程式，當監聽的事件被觸發時，就會觸發 Cloud Function 裡所部署的程式
 - 監聽的事件如
 - › LINE BOT做HTTP的POST請求
 - › 有外部的程式做HTTP的GET請求

啟用Google Function API

› 在API程式庫裡選擇Cloud Build API


 The page did not load correctly. Please reload the page or, if the issue persists, try again later [重新載入](#)



Cloud Build API

[Google Enterprise API](#)

Continuously build, test, and deploy.

 [試用這個 API](#)

[總覽](#) [定價](#) [說明文件](#) [相關產品](#)

Google Cloud Function 建立函式

› 搜尋cloud function

 × 🔍 搜尋

搜尋結果

目前顯示 30 筆「cloud function」的結果，共有 150 筆。

(...) [Cloud Functions](#)

可擴充的函式即服務 (FaaS)，讓您以即付即用的方式執行程式碼，完全不必管理伺服器

類型： 產品或頁面



歡迎使用 Cloud Functions !

Cloud Functions is a lightweight, event-based, asynchronous compute solution that allows you to create small, single-purpose functions that respond to cloud events - without the need to manage a server or a runtime environment.

[Learn More](#)

建立函式

[參加快速入門導覽課程](#)

Google Cloud Function 建立函式

› 選擇函式基本資訊和觸發條件

基本

環境
第 2 代

函式名稱 *
LINEBOT_callback

地區
asia-east1

觸發條件

☒ HTTPS

驗證

☒ 允許未經驗證的叫用
如要建立公用 API 或網站，請勾選這個選項。

☐ 需要驗證
請透過 Cloud IAM 管理已獲授權的使用者。

+ 新增 EVENTARC 觸發條件

執行階段、建構作業、連線和安全性設定

撰寫Python處理程式

› Cloud function會給一個內建的template

```
1  import functions_framework
2
3  @functions_framework.http
4  def hello_http(request):
5      """HTTP Cloud Function.
6      Args:
7          request (flask.Request): The request object.
8          <https://flask.palletsprojects.com/en/1.1.x/api/#incoming-request-data>
9      Returns:
10         The response text, or any set of values that can be turned into a
11         Response object using `make_response`
12         <https://flask.palletsprojects.com/en/1.1.x/api/#flask.make\_response>.
13      """
14      request_json = request.get_json(silent=True)
15      request_args = request.args
16
17      if request_json and 'name' in request_json:
18          name = request_json['name']
19      elif request_args and 'name' in request_args:
20          name = request_args['name']
21      else:
22          name = 'World'
23      return 'Hello {}'.format(name)
24
```


設定需要的模組

- › 編輯 requirements.txt
 - 安裝line-bot-sdk 函式庫
 - 安裝googlemap函式庫

執行階段
Python 3.10

原始碼
內嵌編輯器

+

main.py

requirements.txt

進入點 *
linebot_map

按下 Alt+F1 鍵即可查看無障礙工具選項。

```
1 functions-framework==3.*  
2 line-bot-sdk  
3 googlemaps
```

撰寫Python處理程式

› 載入需要的模組

- from linebot import LineBotApi, WebhookHandler
- from linebot.models import TextSendMessage, StickerSendMessage, ImageSendMessage, LocationSendMessage
- import googlemaps
- import json

撰寫Python處理程式

- › 建立LINEBOT物件並輸入access token
 - access_token = 'token'
 - line_bot_api = LineBotApi(access_token)
- › 從HTTPS的POST取得使用者輸入的文字訊息
 - body = request.get_data(as_text=True)
 - json_data = json.loads(body)
 - msg = json_data['events'][0]['message']['text']
 - tk = json_data['events'][0]['replyToken']

撰寫Python處理程式

- › 將文字訊息當關鍵字用google map api查詢
 - `gmaps=googlemaps.Client(key="google map api key")`
 - `geocode_result = gmaps.geocode(msg)`
- › 分析查詢結果並將需要的資訊傳入LINE BOT API
 - `mes = geocode_result[0]['place_id']`
- › 用LINE BOT API將最後的結果回傳給使用者
 - `line_bot_api.reply_message(tk, TextSendMessage(text=mes))`