專業排版與設計 期中考題

LINE Messaging API 教學文件製作

座號: 017

班級: 資管 4A

學號: 411631277

姓名: 張誠恩

November 9, 2025

目錄

圖目錄

程式碼目錄

第 1 章　LINE Messaging API 簡介與應用場景

1.1　LINE Messaging API 概述

LINE Messaging API 是由 LINE 公司提供的雲端服務介面，旨在讓開發者能以程式方式操作 LINE 聊天平台的互動功能。透過此 API，開發者可以建立能夠主動推播訊息、回覆使用者輸入、甚至與後端資料庫互動的自動化聊天系統。

Messaging API 採用 RESTful 架構，所有互動皆以 HTTPS 為基礎的 POST/GET 請求完成。開發者透過 LINE Developers 平台建立一個「Channel」，並使用「Channel Access Token」來進行身分驗證與授權。LINE 平台再透過 Webhook 機制，將使用者事件（例如訊息、加入好友、按鈕互動等）傳送至開發者設定的伺服器。

此技術的誕生使得企業不再需要自行開發聊天系統，而能以 LINE 為前端通道，將客戶溝通、行銷與服務集中於單一平台。這大幅降低開發成本，同時擴大應用覆蓋範圍。

1.2　常見應用場景與發展趨勢

Messaging API 的應用非常廣泛，以下列出幾個具代表性的實際場景：

1. 智慧客服機器人: 許多企業利用 LINE Bot 回覆常見問題，整合 FAQ、訂單查詢、報修狀態查詢等功能，減少人力成本。
2. 行銷與推播通知: 透過 Broadcast Message 或 Multicast API，企業可向特定用戶群發送促銷訊息，並根據互動紀錄分析點擊率。
3. 服務整合平台: 結合 Google Sheets、Firebase 或 MySQL，可實現會員查詢、預約系統、積分兌換等功能。
4. 教育與社群應用學校可利用 LINE Bot 提供課程公告、作業提醒，社群團體亦能透過 Bot 管理活動報名。

展望未來，LINE Messaging API 將持續朝「智慧化」、「個人化」與「跨平台整合」方向發展。AI 對話與使用者行為分析將成為關鍵技術，使 Bot 不僅能回覆問題，更能主動理解與預測使用者需求。

第 2 章　LINE Developers 帳號註冊申請與 Channel 建立流程

2.1　LINE Developers 帳號建立

要使用 Messaging API，第一步是註冊 LINE Developers 帳號。開發者可前往 https://developers.line.biz/ 網站，以現有的 LINE 帳號登入。進入後可建立一個或多個「Provider」——代表開發者或公司身分。

建立 Provider 後，平台會要求設定名稱、聯絡信箱與用途描述。Provider 相當於開發者群組，而每個 Provider 底下可以擁有多個 Channel，用以區分不同應用程式或專案。

在帳號註冊完成後，開發者即可在主控台中看到專案清單與狀態，並能管理權限、API 使用紀錄等資訊。

2.2　建立 Messaging API Channel 流程

1. 點擊「Create a new channel」
2. 選擇類型：Messaging API
3. 填寫 Channel 名稱、描述與類別
4. 勾選同意使用條款
5. 提交後即可在控制台看到新建的 Channel

建立完成後，平台會自動生成：

- Channel ID
- Channel Secret
- Channel Access Token（可重新產生）

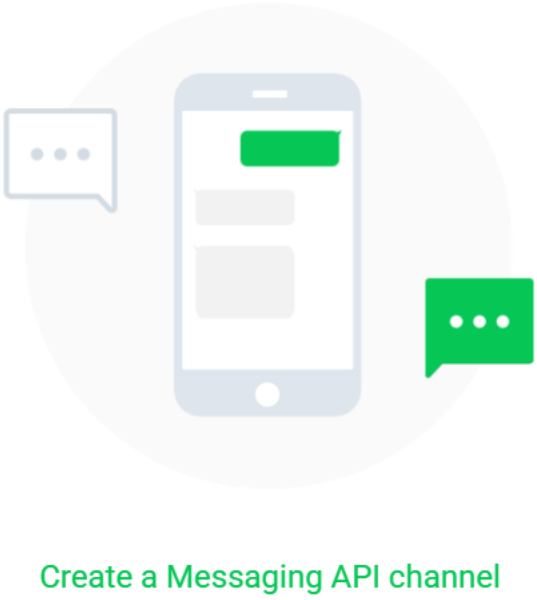

圖 2.1: Create a new channel 圖示

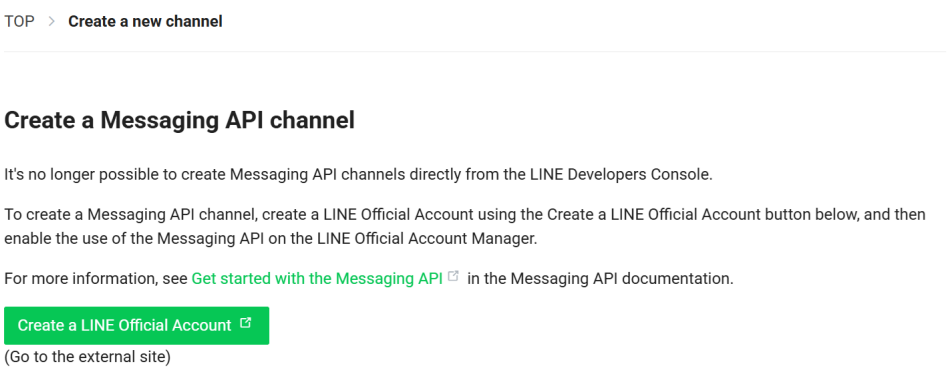

圖 2.2: 創建帳號

圖 2.3: 填寫 Channel 名稱、描述與類別

這些資訊是後續整合 API 的關鍵。此外，Channel 頁面也提供多項設定，如 Webhook URL、功能權限、回覆模式（自動或手動）等。

在實務開發中，建議將 Channel 區分為「開發測試用」與「正式營運用」兩組，以便版本控制與權限管理。

第 3 章　Channel Secret 及 Access Token 取得與設定

3.1　建立 Messaging API Channel 流程

Channel Secret 是由 LINE 平台自動產生的一串字串，用來驗證 Webhook 的來源真偽。每當 LINE 伺服器發送事件通知至你的伺服器時，HTTP Header 內會附帶一個簽章（Signature）。伺服器端應以 Channel Secret 進行 HMAC-SHA256 驗證，以確認請求未被偽造。

若 Secret 外洩，惡意者可能偽造事件請求並誘使伺服器誤判，因此務必妥善保管。常見作法是使用環境變數儲存，或以.env 文件方式讀取。

3.2　Access Token 的產生與管理

Channel Access Token 是讓伺服器代表 LINE Bot 呼叫 API 的授權金鑰。在 Messaging API 頁面中，開發者可選擇「Issue」產生新的 Token。建議選用長期有效（Long-lived）的 Token，以避免頻繁更新。

範例設定方式如下（Python Flask 環境）：

程式碼 3.1: Python Flask 環境範例

```
export LINE_CHANNEL_SECRET="abcdef1234567890"
export LINE_CHANNEL_ACCESS_TOKEN="E9qU3Y3xyz ... "
```

此 Token 在應用程式中用於初始化 LineBotApi 物件。若需要強化安全性，可定期重新產生 Token 並更新伺服器設定。

此外，也建議啟用伺服器防火牆與 HTTPS 憑證，以確保通訊安全。

第 4 章　Webhook 的設定與測試方式（可使用 ngrok 或伺服器）

4.1　Webhook 機制說明

Webhook 是 Messaging API 的核心之一。當使用者在 LINE 對話中輸入文字或觸發事件時，LINE 伺服器會立即將該事件以 JSON 格式發送至你設定的 Webhook URL。

Webhook 的優點在於：

- 即時性高：可立即回應使用者操作。
- 節省輪詢資源：伺服器不需主動查詢，僅在事件發生時處理。
- 易於整合：可與資料庫、AI 模組或外部 API 結合。

Webhook 的資料結構包括事件類型（如 message、follow、unfollow、join 等）以及使用者 ID、訊息內容、時間戳記等欄位。

4.2　Webhook 機制說明

若開發環境在本地端，無法被外部連線，此時可利用 ngrok 工具。

安裝後，執行以下命令：

程式碼 4.1: ngrok 指令 1

```
ngrok http 5000
```

ngrok 會建立一條安全通道，並產生一個 HTTPS 公開網址，例如：

程式碼 4.2: ngrok 指令 2

```
https://abcd1234.ngrok.io
```

將此網址填入 LINE Developers 控制台的「Webhook URL」欄位，然後點擊「Verify」按鈕。若顯示成功，即表示本地伺服器能正常接收事件。

在測試過程中，可於 Flask 伺服器輸出 print() 訊息以觀察事件內容，協助除錯。

第 5 章　範例程式說明（接收及回覆 LINE 訊息）

5.1　程式架構介紹

本範例使用 Python Flask 框架，搭配官方提供的”line-bot-sdk” 套件。整體流程如下：

1. 使用者在 LINE 中傳送訊息給 Bot。
2. LINE 伺服器透過 Webhook 將事件傳送至 Flask /callback 路由。
3. Flask 接收到事件後交由 WebhookHandler 驗證簽章。
4. 若為文字訊息事件，則以相同文字回覆。

此結構簡單明瞭，適合初學者快速上手。

5.2　程式碼與執行步驟

安裝必要套件：

程式碼 5.1: 安裝必要條件

```
pip install flask line-bot-sdk
```

範例主程式如下：

程式碼 5.2: 範例主程式

```
from flask import Flask, request, abort
from linebot import LineBotApi, WebhookHandler
from linebot.exceptions import InvalidSignatureError
from linebot.models import MessageEvent, TextMessage, TextSendMessage

app = Flask(__name__)

line_bot_api = LineBotApi('YOUR_CHANNEL_ACCESS_TOKEN')
handler = WebhookHandler('YOUR_CHANNEL_SECRET')

@app.route("/callback", methods=['POST'])
def callback():
    signature = request.headers['X-Line-Signature']
    body = request.get_data(as_text=True)
    try:
        handler.handle(body, signature)
    except InvalidSignatureError:
```

```
        abort(400)
    return 'OK'

@handler.add(MessageEvent, message=TextMessage)
def handle_message(event):
    msg = event.message.text
    reply = f"你說了: {msg}"
    line_bot_api.reply_message(
        event.reply_token,
        TextSendMessage(text=reply)
    )

if __name__ == "__main__":
    app.run(port=5000)
```

執行後搭配 ngrok 測試，即可於手機上體驗 Bot 的回覆行為。

若要擴充，可加入圖片訊息（ImageMessage）、按鈕模板或 Flex Message。

第 6 章　結論與未來發展

6.1　開發成果總結

本文件完整說明了 LINE Messaging API 的基礎原理、開發流程、安全設定與測試方式，並提供實際程式範例。透過此流程，開發者能從無到有建立一個能與使用者互動的 LINE Bot，並掌握從 Channel 設定到 Webhook 驗證的關鍵步驟。

本範例雖屬基礎型聊天機器人，但其架構具延展性，可應用於各類智慧服務，例如即時客服、智慧助理、行銷推播與 IoT 裝置整合等。

6.2　未來應用方向與挑戰

未來的 LINE Bot 開發趨勢將聚焦於以下方向：

1. 人工智慧整合：結合 ChatGPT、Dialogflow 等 NLP 模型，提供更自然的對話體驗。
2. 多媒體互動：支援圖片、影片、地圖、音訊回覆，豐富使用者體驗。
3. 資料分析與行銷決策：藉由使用者互動紀錄進行行為分析，優化產品策略。
4. 跨平台串接：將 LINE 與網站、App、CRM 系統互通，形成完整的服務生態。

不過，開發者也必須面對如使用者隱私保護、API 權限管理與伺服器安全等挑戰。唯有兼顧便利與安全，才能打造穩定、可信賴的智慧對話應用。