使用ASP.NET(C#)建立LineChatBot服務

Day 2

第1步

閱讀官方文件，重點：

1. **運作流程**
2. **有哪些API**
3. **API的Spec**

第2步

測試API

1. **準備好要測試的API的Spec**
2. **使用工具測試(Postman)**

第3步

把API實作成程式(以我們的C#範例，使用RestSharp Library)

Open Weather API

使用第3方Web API查詢城市氣象

開通及閱讀Open Weather API文件

- Open Weather API：提供一組Web API讓開發者可以介接取得地點氣象資料
  - 申請使用Open Weather API
    - https://openweathermap.org/
    - 啟用API Key
  - 計費：標準
    - 可使用API：
      - 目前天氣資訊
      - 5天內的3小時天氣資訊

- Tips：Open Weather API取得天氣資訊是用GPS位置(lat/lon)。所以需要先用GeoLocation API取得城市的lat/lon，才能用lat/lon去request天氣API

每次介接API, 都需要以下資訊

- Request：
  - Url/host + endpoint
  - Http method
  - Http Header
  - Parameters (Querystring parameters)
  - Body Content-type
  - Body format
- Response:
  - Response Content Definition

練習：介接目前天氣資訊

- Request：
  - Url/host + endpoint：
  - Http method：
  - Http Header：
  - Parameters (Querystring parameters)：
  - Body Content-type：
  - Body format：
- Response:
  - Response Content Definition在哪？

練習(解答):介接目前天氣資訊

- Request:
  - Url/host + endpoint: `https://api.openweathermap.org/data/2.5/weather`
  - Http method: `GET`
  - Http Header: `N/A`
  - Parameters (Querystring parameters):
    - `lat={lat}`
    - `lon={lon}`
    - `appid={API key}`
    - `...(optional)`
  - Body Content-type: `N/A`
  - Body format: `N/A`
- Response:
  - 見文件的Response JSON

練習：Postman測試目前天氣資訊

Postman測試目前天氣資訊

Scu-LineChatbot / WeatherQuery / Open Weather API / **current weather**

Save

1 GET

2 https://api.openweathermap.org/data/2.5/weather?lat=25.0375198&lon=121.5636796&appid=5e5f17ffd5692cc262a291de05237aa6

Send

Params • Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

3

KEY	VALUE	DESCRIPTION
lat	25.0375198	
lon	121.5636796	
appid	5e5f17ffd5692cc262a291de05237aa6	
units	metric	
Key	Value	Description

Bulk Edit

Body Cookies Headers (9) Test Results

Status: 200 OK Time: 337 ms Size: 815 B Save Response

Pretty Raw Preview Visualize JSON

4


```
{
    "coord": {
        "lon": 121.5637,
        "lat": 25.0375
    },
    "weather": [
        {
            "id": 801,
            "main": "Clouds",
            "description": "few clouds",
            "icon": "02d"
        }
    ],
    "base": "stations",
    "main": {
        "temp": 29.59,
        "feels_like": 31.55,
        "temp_min": 27.79,
        "temp_max": 30.91,
```

練習：介接5天內的3小時天氣資訊

- Request：
  - Url/host + endpoint：
  - Http method：
  - Http Header：
  - Parameters (Querystring parameters)：
  - Body Content-type：
  - Body format：
- Response:
  - Response Content Definition在哪？

- Tips：Open Weather API取得天氣資訊是用GPS位置(lat/lon)。所以需要先用GeoLocation API取得城市的lat/lon，才能用lat/lon去request天氣API

練習(解答)：介接5天內的3小時天氣資訊

- Request：
  - Url/host + endpoint：`api.openweathermap.org/data/2.5/forecast`
  - Http method：`GET`
  - Http Header：`N/A`
  - Parameters (Querystring parameters)：
    - `lat={lat}`
    - `lon={lon}`
    - `appid={API key}`
    - `…optional`
  - Body Content-type：`N/A`
  - Body format：`N/A`
- Response:
  - 見文件的Response JSON

- Tips：Open Weather API取得天氣資訊是用GPS位置(lat/lon)。所以需要先用GeoLocation API取得城市的lat/lon，才能用lat/lon去request天氣API

實作程式-介接目前天氣資訊

- 目的：要**寫一支WebAPI**，用這支WebAPI去介接『台北市』目前的天氣資訊。並用Postman來測試此WebAPI
- 實作的WebAPI的Spec：
  - Request：
    - (Url/host) + endpoint：/OpenWeather/Current
    - Http method：GET
    - Parameters (Querystring parameters)：N/A
    - Body Content-type：N/A
    - Body format：N/A
  - Response:
    - 直接把OpenWeather API調到的Response打出來
    - 只取出 **實際溫度、體感溫度、天氣描述** 三欄(進階)

實作程式-介接目前天氣資訊

寫一支Web API(依前頁的spec)

```
[HttpGet]
0 references
public async Task<IActionResult> CurrentWeather(string city)
{
    var cityLocation = this.geoLocationList.Where(geo => geo.name == city).FirstOrDefault();
    if (cityLocation == null)
    {
        return BadRequest("Cannot find city");
    }
    else
    {
        #region 使用RestSharp Library介接Open Weather API
        string apiEndpoint = $"https://api.openweathermap.org/data/2.5/weather?lat={cityLocation.lat}&lon={cityLocation.lon}&appid={apiKey}&lang=zh_tw&units=metric";

        RestClient client;
        RestRequest request;
        client = new RestClient(apiEndpoint);

        request = new RestRequest();
        request.Method = Method.GET;

        //呼叫API取得Response
        IRestResponse t = await client.ExecuteAsync(request, new CancellationTokenSource().Token);
        #endregion

        if (t.StatusCode == HttpStatusCode.OK)
        {
            //把API讀到的Body，轉型成自定的Model (Class Weather)
            Weather weather = JsonConvert.DeserializeObject<Weather>(t.Content);
            //把結果回應到 API的response
            return Ok(weather);
        }
        else
        {
            return BadRequest($"status: {t.StatusCode}, content: {t.Content}");
        }
    }
}
```

用Restsharp Library呼叫Open Weather API

練習Postman測試介接5天內的3小時天氣資訊

解答Postman測試介接5天內的3小時天氣資訊

1 GET

2 https://api.openweathermap.org/data/2.5/forecast?lat=25.0375198&lon=121.5636796&appid=5e5f17ffd5692cc262a291de05237a...

Send

3

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

KEY	VALUE	DESCRIPTION
lat	25.0375198	
lon	121.5636796	
appid	5e5f17ffd5692cc262a291de05237aa6	
units	metric	
Key	Value	Description

Body Cookies Headers (9) Test Results Status: 200 OK Time: 139 ms Size: 16.33 KB Save Response

Pretty Raw Preview Visualize JSON

4


```
{
    "cod": "200",
    "message": 0,
    "cnt": 40,
    "list": [
        {
            "dt": 1663912800,
            "main": {
                "temp": 29.75,
                "feels_like": 31.64,
                "temp_min": 28.26,
                "temp_max": 29.75,
                "pressure": 1012,
                "sea_level": 1012,
                "grnd_level": 1009,
                "humidity": 56,
                "temp_kf": 1.49
            },
            "weather": [
```

C#的JSON object字串與物件類的轉型

C# Web API input JSON Body自動轉型為C#物件

- 第1種-Web API的自動轉型：
  - Web API在接受JSON格式[FromBody]時，會直接把下面的request body轉型為C#物件
    - Content-Type=application/json
    - Body：JSON 物件
- 第2種-透過Library轉型：
  - 透過JsonConvert的library把『字串』型別的JSON物件內容，轉型為c#物件
- 以上兩種轉型，都需要把自訂的Class定義得與JSON Object一致

JSON Object字串與C# Class轉換

```
{
    "weather": [
        {
            "main": "Clouds",
            "description": "晴，少雲"
        }
    ],
    "main": {
        "temp": 24.21,
        "feels_like": 24.62,
        "temp_min": 22.94,
        "temp_max": 25.53,
        "humidity": 74
    },
    "visibility": 10000
}
```

JSON物件	C# Class
字串 1	string/string?
數值 2	double?/int?
陣列 3	List<T>/IEnumerable<T>
物件 4	自訂class
property名稱(大小寫需與c#物件一致)	property名稱(大小寫需與JSON物件一致)

JSON Object字串與C# Class轉換

1

```
{
    "weather": [
        {
            "main": "Clouds",
            "description": "晴，少雲"
        }
    ],
    "main": {
        "temp": 24.21,
        "feels_like": 24.62,
        "temp_min": 22.94,
        "temp_max": 25.53,
        "humidity": 74
    },
    "visibility": 10000
}
```

2

3

Models > OpenWeather > Weather.cs > {} HelloWorldMvc.Models.OpenWeather

```
1   namespace HelloWorldMvc.Models.OpenWeather
2   {
        12 references
3       public class Weather
4       {
            1 reference
5           public IEnumerable<WeatherDescription>? weather { get; set; }
            2 references
6           public Temperature? main { get; set; }
7           //public Wind? wind { get; set; }
            0 references
8           public double visibility { get; set; }
9
10      }
11
12      /*…
58  }
```

1

2

3

JSON Object字串與C# Class轉換比對範例

```json
{
    "weather": [
        {
            "main": "Clouds",
            "description": "晴，少雲"
        }
    ],
    "main": {
        "temp": 24.21,
        "feels_like": 24.62,
        "temp_min": 22.94,
        "temp_max": 25.53,
        "humidity": 74
    },
    "visibility": 10000
}
```

1

Models > OpenWeather > Temperature.cs > {} HelloWorldMvc.Models.OpenWeather

```
1   namespace HelloWorldMvc.Models.OpenWeather
2   
    1 reference
3   public class Temperature
4   {
        1 reference
5       public double temp { get; set; }
        1 reference
6       public double feels_like { get; set; }
        0 references
7       public double temp_min { get; set; }
        0 references
8       public double temp_max { get; set; }
        0 references
9       public double humidity { get; set; }
10  }
11  }
```

1

JSON Response與C# Class轉換比對範例

```
{
    "weather": [
        1 {
            "main": "Clouds",
            "description": "晴，少雲"
        }
    ],
    "main": {
        "temp": 24.21,
        "feels_like": 24.62,
        "temp_min": 22.94,
        "temp_max": 25.53,
        "humidity": 74
    },
    "visibility": 10000
}
```

Models > OpenWeather > WeatherDescription.cs > {} HelloWorldMvc.Models.OpenWeather

```
1  namespace HelloWorldMvc.Models.OpenWeather
2
   2 references
3  1 public class WeatherDescription
4  {
       0 references
5      public string? main { get; set; }
       1 reference
6      public string? description{get;set;}
7
8  }
9  }
```

實作程式-介接5天內的3小時天氣資訊

- 目的：要**寫一支WebAPI**，用這支WebAPI去介接『台北市』目前的天氣資訊。並用Postman來測試此WebAPI
- 實作的WebAPI的Spec：
  - Request：
    - (Url/host) + endpoint：/OpenWeather/Forecast
    - Http method：GET
    - Parameters (Querystring parameters)：N/A
    - Body Content-type：N/A
    - Body format：N/A
  - Response:
    - 直接把OpenWeather API調到的資料打出來

實作程式-介接5天內的3小時天氣資訊

- 程式解析

小結

- 實作介接第3方API，可以使用RestSharp Library
- 介接API時，要轉換JSON物件字串與C#物件，可使用Newton的JsonConvert library
- (彩蛋)上面的範例，如何可以不用寫這麼長的程式
  - 方法1：使用老師直接封裝的Library
    - 因為重覆的部分，已經被老師封裝在 Library裡面了)
  - 方法2：程式碼產生器
    - 透過工具直接把code產生出來

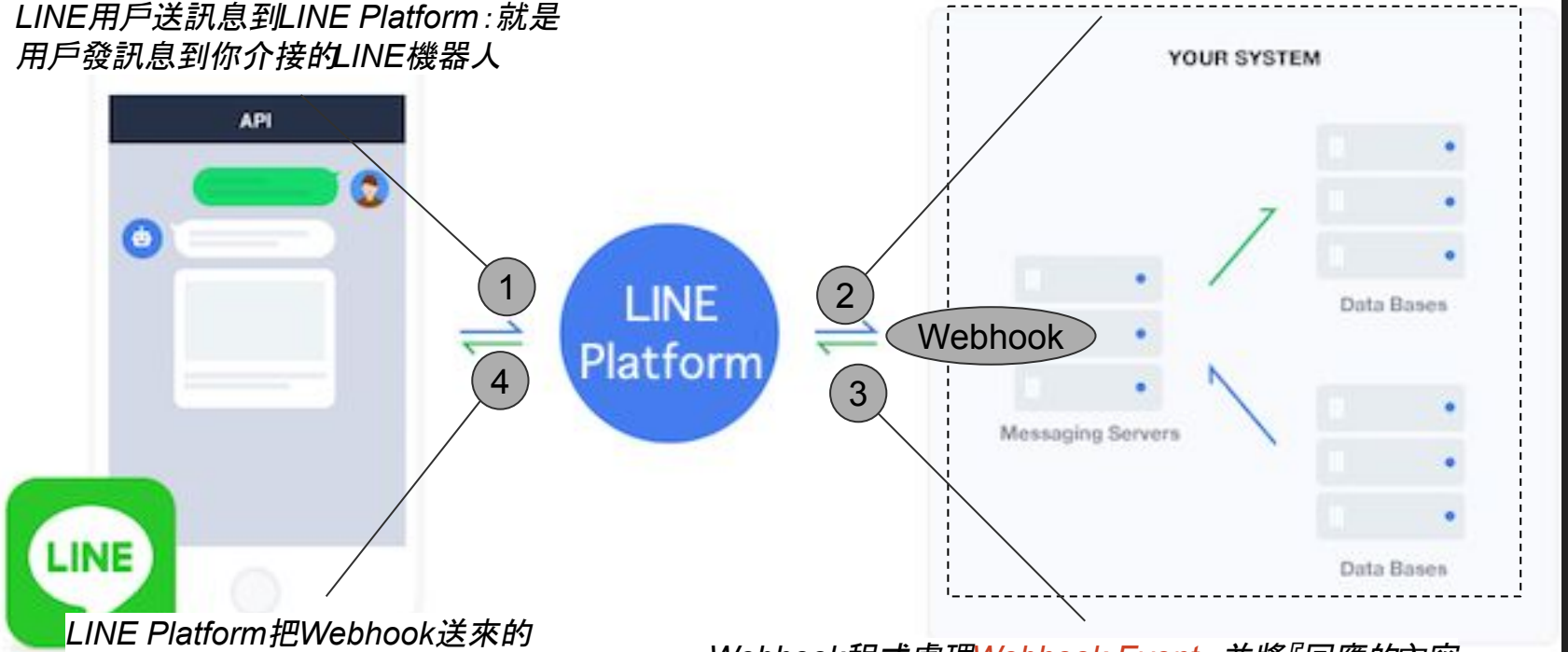
介接LINE Messaging API

開始接LINE機器人囉

Webhook基本運作解析

LINE Platform把訊息轉成Webhook Event送request到開發者的Webhook

LINE用戶送訊息到LINE Platform：就是用戶發訊息到你介接的LINE機器人



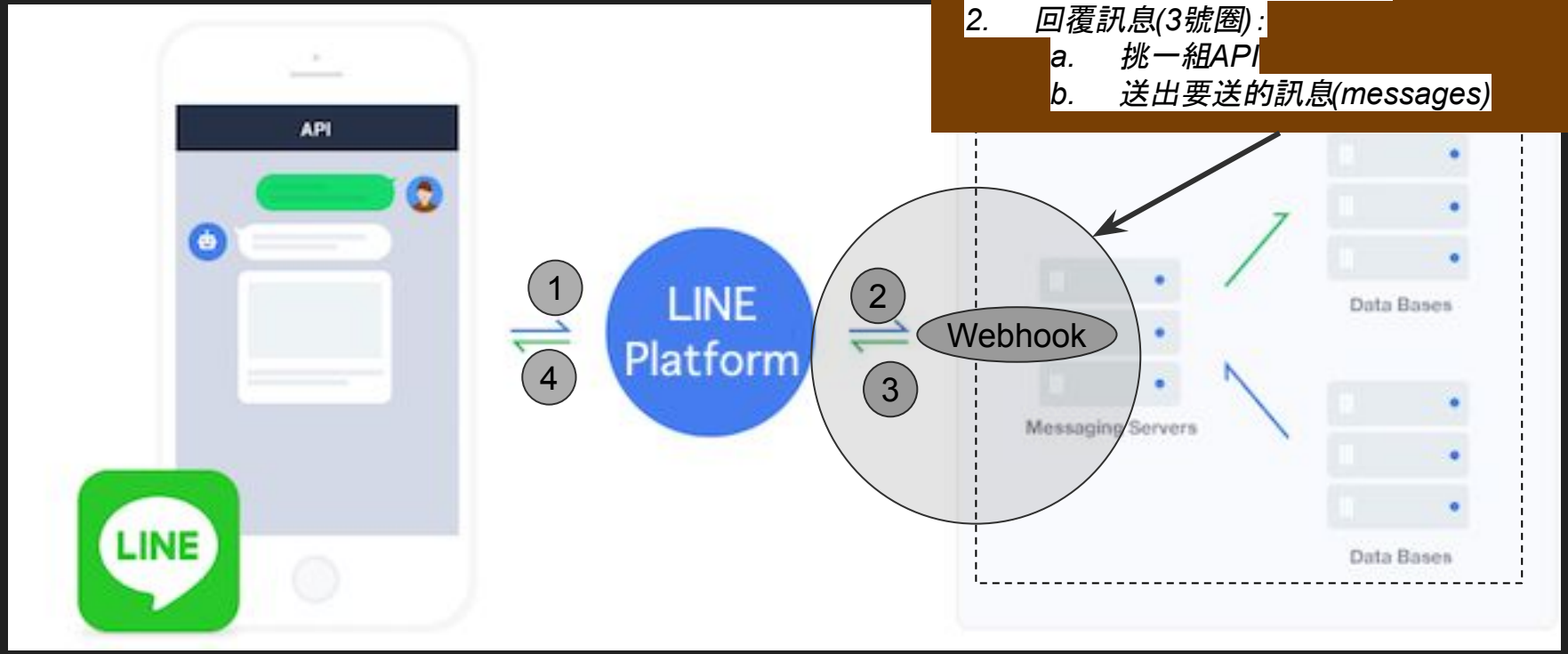
LINE Platform把Webhook送來的Message(s)轉送到對應的LINE 用戶

Webhook程式處理Webhook Event，並將『回應的內容message(s)』送出request到LINE Platform

Webhook基本運作解析

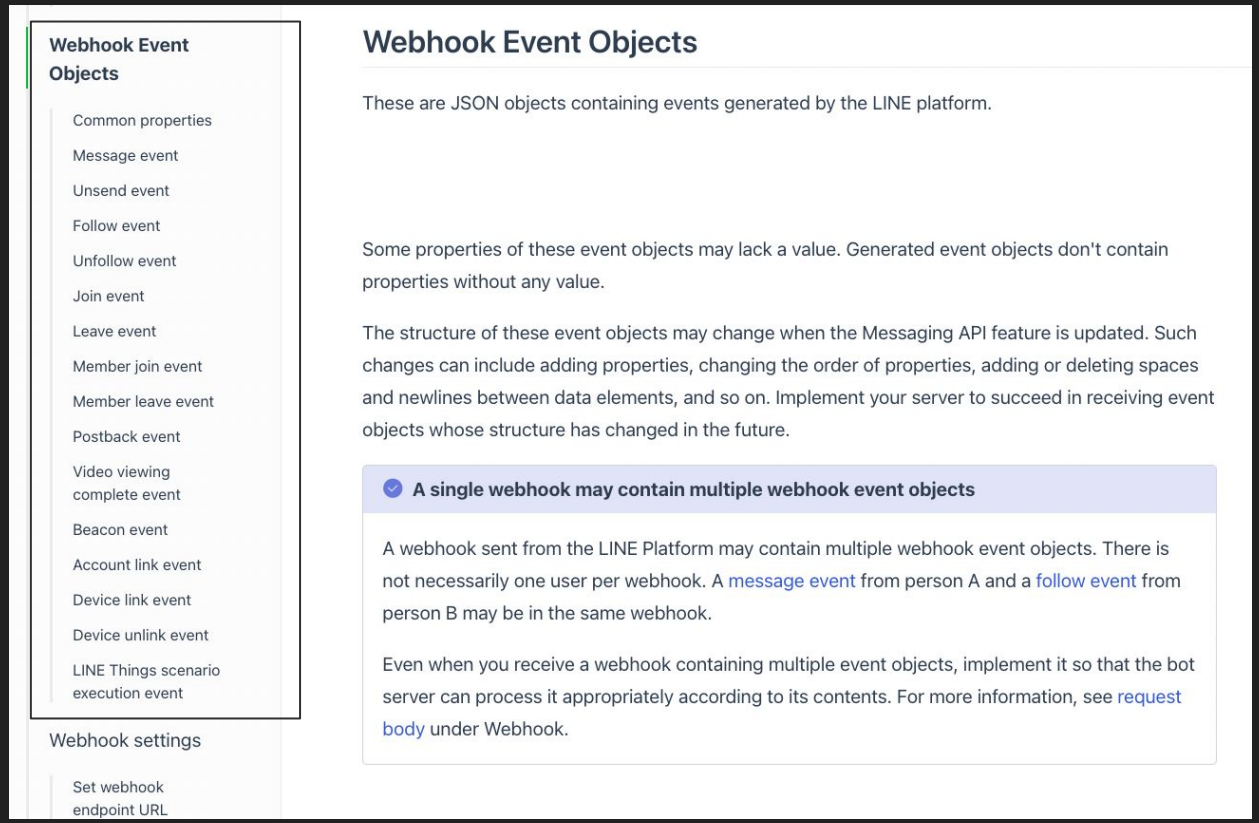
介接LINE Messaging API實作LINE Chatbot，實際上就是在寫Webhook這支API

1. 接收訊息(2號圈)：
   a. 怎麼接Webhook event
   b. 解析Webhook event
2. 回覆訊息(3號圈)：
   a. 挑一組API
   b. 送出要送的訊息(messages)



Webhook Events

- 怎麼接Webhook events
- 解析Webhook events(有哪些events)



Webhook Event Objects

- Common properties
- Message event
- Unsend event
- Follow event
- Unfollow event
- Join event
- Leave event
- Member join event
- Member leave event
- Postback event
- Video viewing complete event
- Beacon event
- Account link event
- Device link event
- Device unlink event
- LINE Things scenario execution event

Webhook settings

- Set webhook endpoint URL

Webhook Event Objects

These are JSON objects containing events generated by the LINE platform.

Some properties of these event objects may lack a value. Generated event objects don't contain properties without any value.

The structure of these event objects may change when the Messaging API feature is updated. Such changes can include adding properties, changing the order of properties, adding or deleting spaces and newlines between data elements, and so on. Implement your server to succeed in receiving event objects whose structure has changed in the future.

A single webhook may contain multiple webhook event objects

A webhook sent from the LINE Platform may contain multiple webhook event objects. There is not necessarily one user per webhook. A message event from person A and a follow event from person B may be in the same webhook.

Even when you receive a webhook containing multiple event objects, implement it so that the bot server can process it appropriately according to its contents. For more information, see request body under Webhook.

送Message的方式

developers.line.biz/en/reference/messaging-api/#messages

LINE Developers News Products Documentation FAQ Glossary More

Message

- Send reply message
- Send push message
- Send multicast message
- Send narrowcast message
- Get narrowcast message status
- Send broadcast message
- Get the target limit for sending messages this month
- Get number of messages sent this month
- Get number of sent reply messages
- Get number of sent push messages
- Get number of sent multicast messages
- Get number of sent broadcast messages
- Retrying an API request

Managing Audience

Message

You can send a message and obtain information about the sent message.

挑一組適合的API

Endpoints

```sh
POST /v2/bot/message/reply
POST /v2/bot/message/push
POST /v2/bot/message/multicast
POST /v2/bot/message/narrowcast
GET  /v2/bot/message/progress/narrowcast
POST /v2/bot/message/broadcast
GET  /v2/bot/message/quota
GET  /v2/bot/message/quota/consumption
GET  /v2/bot/message/delivery/reply
GET  /v2/bot/message/delivery/push
GET  /v2/bot/message/delivery/multicast
GET  /v2/bot/message/delivery/broadcast
```

Send reply message

有哪些Message種類可以送

LINE Developers News Products Documentation FAQ Glossary More

Message objects

- Common properties for messages
- Text message
- Sticker message
- Image message
- Video message
- Audio message
- Location message
- Imagemap message
- Template messages
- Flex Message

挑要送的訊息種類

Action objects

- Postback action
- Message action
- URI action
- Datetime picker

Message objects

JSON object which contains the contents of the message you send.

Common properties for messages

The following properties can be specified in all the message objects.

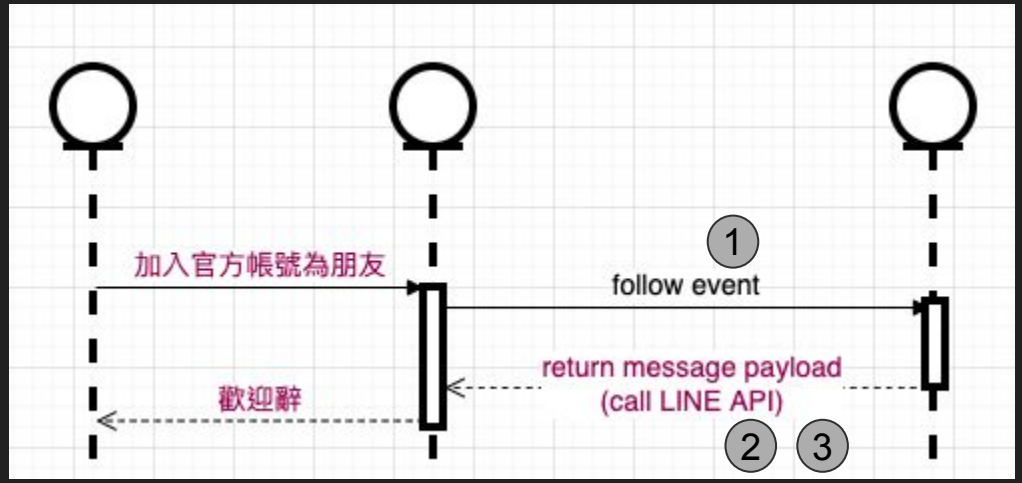
Quick reply

These properties are used for the quick reply feature. For more information, see Using quick replies.

LINE Chatbot - Hello Visitor

LINE Chatbot-Hello Visitor

- 1 判斷是收到"follow event"：表示有使用者把我們的Chatbot加入好友
- 2 挑選要送訊息的API(方式)：reply message
- 3. 挑選要送的訊息(message)種類



Welcome Visitor

- 修改Webhook：
  - 接受LINE訊息
    - Follow Event
      - 取得reply token
  - Reply Messages
    - Message類型：Text
    - Message內容：
      - "歡迎加入氣象 查詢機器人“

Part 1(1/2)：接收Follow Event並解析

- 解析Webhook Follow Event
  - https://developers.line.biz/en/reference/messaging-api/#follow-event
- 準備Model對應上面的JSON格式，並設定為Webhook action的Input parameter

```
public class LINEMessagingAppController : Controller
{
    0 references
    public LINEMessagingAppController()
    {
    }

    [HttpPost]
    0 references
    public async Task<IActionResult> Webhook([FromBody] ReceivedFollowEvent webhookEvent)
    {
        //for the very 1st test from LINE Developer Console
        //our webhook can just return HTTP 200
        //to test don't forget to bridge local web app to the Internet via ngrok
        // and to do that, need to comment out app.UseHttpsRedirection(); in Program.cs
        //return Ok();

        //to parse coming parameter (JSON data) to string
        // and using Console.WriteLine to show on the terminal
        //Console.WriteLine(JsonConvert.SerializeObject(webhookEvent));

        if (webhookEvent.events != null)
        {
            //查詢是否有follow event
            var followEvent = webhookEvent.events.Where(obj => obj.type == "follow").FirstOrDefault();
            if (followEvent != null)
            {
                string? replyToken = followEvent.replyToken;

                var response = await this.replyWelcomeMessage(replyToken);
                Console.WriteLine($"send reply message & get status code: {response.StatusCode}, content:
            }
        }

        return Ok();
    }
```

1 2 3 4 5

- Tips：要確認傳進來的payload是否是follow event，使用Linq的Where方式查詢

Follow event example

JSON

```json
{
  "destination": "xxxxxxxxxx",
  "events": [
    {
      "replyToken": "nHuyWiB7yP5Zw52FIkcQobQuGDXCTA",
      "type": "follow",
      "mode": "active",
      "timestamp": 1462629479859,
      "source": {
        "type": "user",
        "userId": "U4af4980629..."
      },
      "webhookEventId": "01FZ74A0TDDPYRVKNK77XKC3ZR",
      "deliveryContext": {
        "isRedelivery": false
      }
    }
  ]
}
```

Models > LINEPayload > FollowEvent.cs > {} HelloWorldMvc.Models.LINEPayload > HelloWorldMvc.Models.LINEPayload.ReceivedFollowE

```
namespace HelloWorldMvc.Models.LINEPayload;

1 reference
public class ReceivedFollowEvent
{
    0 references
    public string? destination { get; set; }
    2 references
    public IEnumerable<FollowEvent> events { get; set; }
}

1 reference
public class FollowEvent
{
    1 reference
    public string? replyToken { get; set; }
    1 reference
    public string? type { get; set; }
    0 references
    public string? mode { get; set; }
    0 references
    public double timestamp { get; set; }
    0 references
    public Source? source { get; set; }
    // "source": {
    // "type": "user",
    // "userId": "U4af4980629..."
    // },
    0 references
    public string? webhookEventId { get; set; }
    // "deliveryContext": {
    // "isRedelivery": false
    // }
}
1 reference
public class Source
{
    0 references
    public string? type { get; set; }
    0 references
    public string? userId { get; set; }
}
```

- Tips: 將WebAPI的JSON物件轉換為C# Model，可以參考以下原則
  - JSON Object -> C# Class
  - JSON Array -> C# IEnumerable<T> or List<T>

Part 1(2/2)：回覆Follow Event歡迎訊息

- Send Reply Message
  - https://developers.line.biz/en/reference/messaging-api/#send-reply-message
- 準備Class對應"Reply Message"的JSON格式
- 使用Restsharp library發訊息給LINE Platform
- 測試：
  - 未加官方好友時，掃描 QrCode加入好友
  - 已加過官方帳號好友時，先『封鎖』，再重新加入好友

- Tips：要對LINE Platform發訊息，記得要確認以下 API資訊：
  - Channel access token
  - Api Url

Example request

Shell Java Go Ruby PHP Perl Python Node.js

sh

```
curl -v -X POST https://api.line.me/v2/bot/message/re
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer {channel access token}' \
-d '{
    "replyToken":"nHuyWiB7yP5Zw52FIkcQobQuGDXCTA",
    "messages":[
        {
            "type":"text",
            "text":"Hello, user"
        },
        {
            "type":"text",
            "text":"May I help you?"
        }
    ]
}'
```

1 2 3 4

Part 1(2/2): 回覆Follow Event歡迎訊息

eatherController.cs 6 | LINEMessagingAppWeatherQueryController.cs 9+ | LINEMessagingAppController.cs 9+ | WebhookEve

Controllers > LINEMessagingAppController.cs > { } HelloWorldMvc.Controllers > HelloWorldMvc.Controllers.LINEMessagingAppController > Webl

```
100     #region 這段:使用RestSharp Library call LINE Messaging API的Reply Message
101     //************** 使用RestSharp Library call LINE Messaging API的Reply Message
102     RestClient client;
103     RestRequest request;
104
105     //PPTX: Url/host + endpoint
106     client = new RestClient("https://api.line.me/v2/bot/message/reply");
107
108     //PPTX: Http method
109     request = new RestRequest(Method.POST);
110
111     //PPTX: Body format
112     //依LINE Messaging API的Reply Message格式，準備要 POST 到 "Reply Message API" 的 "Text Message" --begin
113     ReplyMessage replyMessage = new ReplyMessage()
114     {
115         replyToken = replyToken
116     };
117
118     List<TextMessage> replyMessageList = new List<TextMessage>();
119     replyMessageList.Add(
120             new TextMessage()
121             {
122                 type = "text"
123                 , text = $"歡迎加入{this.chatbotName} (RestSharp from scratch)"
124             }
125     );
126     replyMessage.messages = replyMessageList;
127     //依LINE Messaging API的Reply Message格式，準備要 POST 到 "Reply Message API" 的 "Text Message" --end
128
129     //PPTX: Request Header
130     //PPTX: Body Content-type: Json Body =>
131     request.AddHeader("Content-Type", "application/json");
132     request.AddHeader("Authorization", $"Bearer {lineChannelAccessToken}");
133
134     //將資料附加在 reuqest
135     request.AddJsonBody(replyMessage);
136
137     IRestResponse t = await client.ExecuteAsync(request, new CancellationTokenSource().Token);
138     Console.WriteLine($"status: {t.StatusCode}, content: {t.Content}");
139     //************** 使用RestSharp Library call LINE Messaging API的Reply Message
140     #endregion
```

Example request

Shell | Java | Go | Ruby | PHP | Perl | Python | Node.js

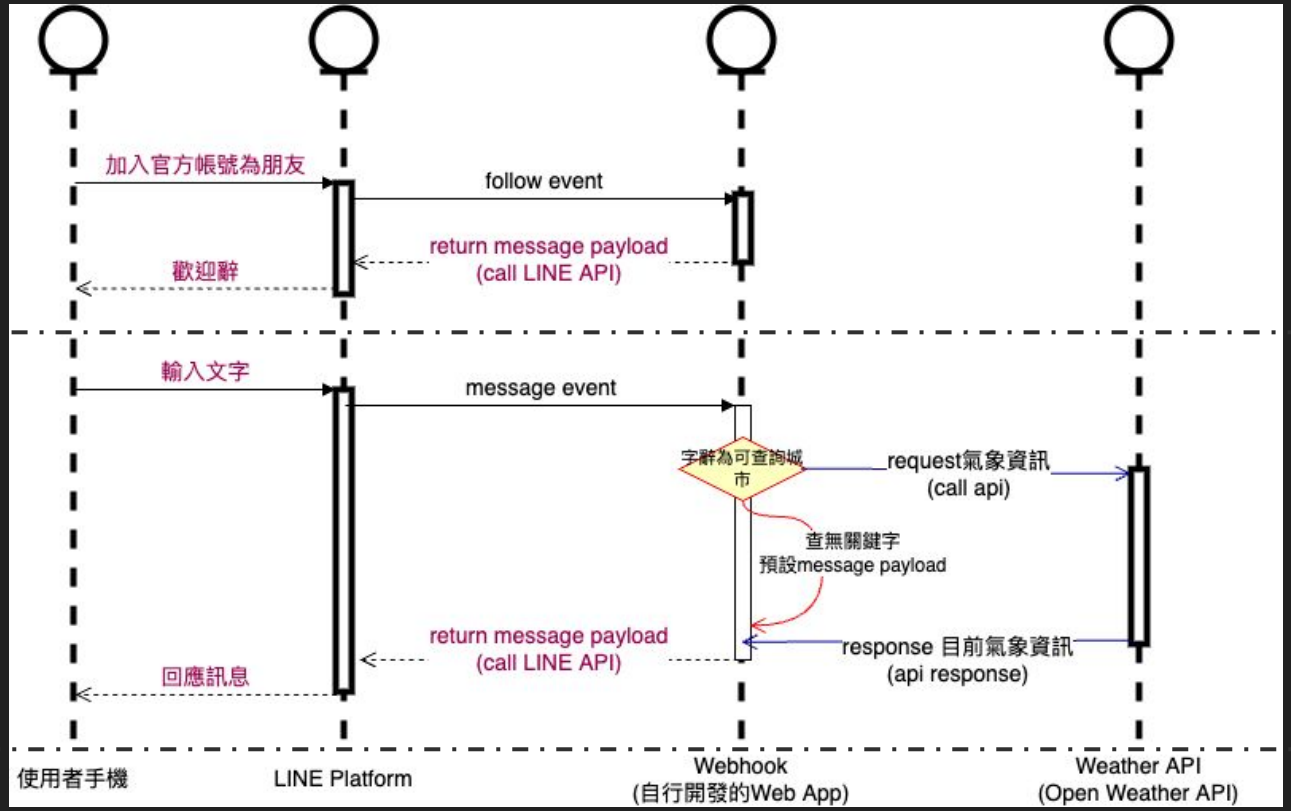
sh

```
curl -v -X POST https://api.line.me/v2/bot/message/re
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer {channel access token}' \
-d '{
    "replyToken":"nHuyWiB7yP5Zw52FIkcQobQuGDXCTA",
    "messages":[
        {
            "type":"text",
            "text":"Hello, user"
        },
        {
            "type":"text",
            "text":"May I help you?"
        }
    ]
}'
```

氣象查詢機器人

整合Open Weather與LINE Messaging API

氣象資訊查詢LINE Chatbot運作全圖



氣象查詢機器人

實作1:回覆使用者輸入的訊息

LINE Messaging API只能接受1個Webhook

- Webhook只有一個，怎麼接受兩種不同格式的JSON body input？
  - 先觀察兩個格式的**異同**
    - **events陣列**內的**物件格式**不同
    - 但events陣列內的物件，不論是Follow Event或Message Event都含有***string type***屬性
  - 把**不同處(events陣列內的物件)**設定為 **dynamic 型別**
  - 在**runtime**時把 dynamic 解析後使用
    - 先解出能讀出string type的物件
    - 再依type的值決定要把**events陣列內的物件**轉型成**Follow Event物件**或**Message Event物件**

LINE Messaging API只能接受1個Webhook

```
[HttpPost]
0 references
public async Task<IActionResult> Webhook([FromBody] ReceivedFollowEvent webhookEvent)
{
```

```
0 references
public class ReceivedFollowEvent
{
    0 references
    public string? destination { get; set; }
    0 references
    public IEnumerable<FollowEvent> events { get; set; }
}
```

```
[HttpPost]
0 references
public async Task<IActionResult> Webhook([FromBody] WebhookEvent webhookEvent)
{
    //for the very 1st test from LINE Developer Console
```

```
1 reference
public class WebhookEvent
{
    0 references
    public string? destination { get; set; }
    2 references
    public IEnumerable<dynamic> events { get; set; }
}
```

LINE Messaging API只能接受1個Webhook

Follow event

Follow event example

JSON

```json
{
  "destination": "xxxxxxxxxx",
  "events": [
    {
      "replyToken": "nHuyWiB7yP5Zw52FIkcQobQuGDXCTA",
      "type": "follow",
      "mode": "active",
      "timestamp": 1462629479859,
      "source": {
        "type": "user",
        "userId": "U4af4980629..."
      },
      "webhookEventId": "01FZ74A0TDDPYRVKNK77XKC3ZR",
      "deliveryContext": {
        "isRedelivery": false
      }
    }
  ]
}
```

Message event(Text Message)

JSON

```json
{
  "destination": "xxxxxxxxxx",
  "events": [
    {
      "replyToken": "nHuyWiB7yP5Zw52FIkcQobQuGDXCTA",
      "type": "message",
      "mode": "active",
      "timestamp": 1462629479859,
      "source": {
        "type": "user",
        "userId": "U4af4980629..."
      },
      "webhookEventId": "01FZ74A0TDDPYRVKNK77XKC3ZR",
      "deliveryContext": {
        "isRedelivery": false
      },
      "message": {
        "id": "325708",
        "type": "text",
        "text": "@example Hello, world! (love)",
        "emojis": [
          {
            "index": 23,
```

LINE Messaging API只能接受1個Webhook

```
if (webhookEvent.events != null)
{
    //查詢是否有follow event
    foreach (dynamic _event in webhookEvent.events)
    {
        WebhookEventBase eventBase = JsonConvert.DeserializeObject<WebhookEventBase>(_event.ToString());

        switch (eventBase.type)
        {
            case "follow":
                FollowEvent followEvent = JsonConvert.DeserializeObject<FollowEvent>(_event.ToString());
                if (followEvent != null)
                {
                    string? replyToken = followEvent.replyToken;

                    var response = await this.replyWelcomeMessage(replyToken);
                    Console.WriteLine($"send reply message & get status code: {response.StatusCode}, content: {respons
                }
                break;
            case "message":
                //parse coming event to Message Event
                MessageEvent messageEvent = JsonConvert.DeserializeObject<MessageEvent>(_event.ToString());

                //get user input text
                if (messageEvent.message != null)
                {
                    string? replyToken = messageEvent.replyToken;
                    string? userInputText = messageEvent.message.text;

                    //var response = await this.replyWelcomeMessage(replyToken, $"你輸入的訊息是{userInputText}");
                    string queryWeatherResult = await this.getCurrentWeatherResult(userInputText);
                    var response = await this.replyWelcomeMessage(replyToken, queryWeatherResult);

                    Console.WriteLine($"send reply message & get status code: {response.StatusCode}, content: {respons
                }

                //find if the input text in any GeoLocation city name

                //reply message

                break;
        }
    }
}
```

```
2 references
public class WebhookEventBase
{
    1 reference
    public string? type { get; set; }
}
```

小結

- 介接第三方Web API流程(經驗分享)
  -
- C#
  - dynamic型別
- JsonConvert Library
  - 可以做字串(string)與物件間的轉型
    - 物件轉字串：`JsonConvert.SerializeObject(obj)`
    - 字串轉物件：`JsonConvert.DeserializeObject<T>(stringVariable)`
- 物件導向
  - 繼承
  - 重構