LINE Office Account Manger (Easy-to-use graphical interface)

- https://manager.line.biz
- rich menu 圖文選單

Rich Menus of Messaging API

- Massaging API(Advanced customization)

create-rich-menu

- You can create up to 1000 rich menus for one LINE Official Account with the Messaging API.
- Create a rich menu with up to 20 different tappable areas.

Steps

- Creates a rich menu.
- upload a rich menu image,
- set the rich menu as the default rich menu

Set the default rich menu

Sets the **default** rich menu. The default rich menu is displayed to **all users** who have added your LINE Official Account as a friend and aren't linked to any per-user rich menu.
https://api.line.me/v2/bot/user/all/richmenu/richmenu-{rich-menu_id}

message api(python)

- create_rich_menu(self, rich_menu, timeout=None)
- set_rich_menu_image(self, rich_menu_id, content_type, content, timeout=None)
- set_default_rich_menu(self, rich_menu_id, timeout=None)

Tools

- flex-simulator web
- opening-the-location-screen

line-bot python SDK

- https://github.com/line/line-bot-sdk-python

STEPS

- see Synopsis need to consider part:

```
@handler.add(MessageEvent, message=TextMessage)
def handle_message(event):
    line_bot_api.reply_message(
```

```
        event.reply_token,
        TextSendMessage(text=event.message.text))
```

Message objects

The following classes are found in the linebot.models package.

- TextSendMessage

```
text_message = TextSendMessage(text='Hello, world')
```

- ImageSendMessage

```
image_message = ImageSendMessage(
    original_content_url='https://example.com/original.jpg', #實際點開來的圖片
    preview_image_url='https://example.com/preview.jpg' #預覽圖片
)
```

VideoSendMessage video_message = VideoSendMessage(original_content_url='https://example.com/original.mp4', preview_image_url='https://example.com/preview.jpg')

- AudioSendMessage

```
audio_message = AudioSendMessage(
    original_content_url='https://example.com/original.m4a',
    duration=240000 #Length of audio file (milliseconds)
)
```

- LocationSendMessage(google map)

```
location_message = LocationSendMessage(
    title='my location',
    address='Tokyo',
    latitude=35.65910807942215,
    longitude=139.70372892916203
)
```

- You can pass a dict(JSON) to FlexSendMessage#contents as follows:

```
flex_message = FlexSendMessage(
    alt_text='hello',
    contents={
        'type': 'bubble',
```

```
        'direction': 'ltr',
        'hero': {
            'type': 'image',
            'url': 'https://example.com/cafe.jpg',
            'size': 'full',
            'aspectRatio': '20:13',
            'aspectMode': 'cover',
            'action': { 'type': 'uri', 'uri': 'http://example.com',
'label': 'label' }
        }
    }
)
```

Thus, You can send a JSON designed with Flex Message Simulator.

WebhookHandler

- default(self) Set the **default handler** method by using this decorator.

```
@handler.default()
def default(event):
    print(event)
```

If there is **no handler for an event**, this default handler method is called.

If the arity of the **handler method** is more than one, a destination property in a webhook request is passed to it as the second argument.

```
@handler.add(FollowEvent)
def handle_follow():
    # do something
```

If the arity of the handler method is zero, the handler method is called with no arguments.

example

```
@handler.add(FollowEvent)
def handle_follow_event(event):
    line_user = LineUser(event.source.sender_id)
    message = TextSendMessage(
        text='Nice to meet You，{name}'.format(name=line_user.name)
    )
    line_bot_api.reply_message(event.reply_token, message) # key part
```

Well... What's the LineUser Model?! Oh... That's mine model. Let's take a look!

```
import json
import requests
class LineUser:
    def __init__(self, user_id):
        self.user_id = user_id
        json = self.__user_profile()
        self.name = json['displayName']
        self.pirture_url = json['pictureUrl']
        self.status_message = json['statusMessage']
    def __user_profile(self):
        """ You can access user profile via LINE_API """
        line_profile_get_url =
            'https://api.line.me/v2/bot/profile/{}'.format(
                self.user_id
            )
        headers = {'Authorization': 'Bearer {}'.format(
            'Your Channel Access Token')}
        json = requests.get(line_profile_get_url,
            headers=headers).json()
        return json
```

Yep, So you can access user profile (just name, picture, status messasge NO MORE!!!)

Webhook event object(使用者傳送過來的資料)

- https://developers.line.biz/en/reference/messaging-api/#webhook-event-objects

The following classes are found in the linebot.models package.

MessageEvent

- type
- mode
- timestamp
- source: Source
- reply_token
- message: Message

```
@handler.add(MessageEvent, message=TextMessage)
def handle_message(event):
    line_bot_api.reply_message(
        event.reply_token,
        TextSendMessage(text=event.message.text))
```

Source

- SourceUser
- - type

- user_id
- SourceGroup
- type
- group_id
- user_id
- SourceRoom
- type
- room_id
- user_id