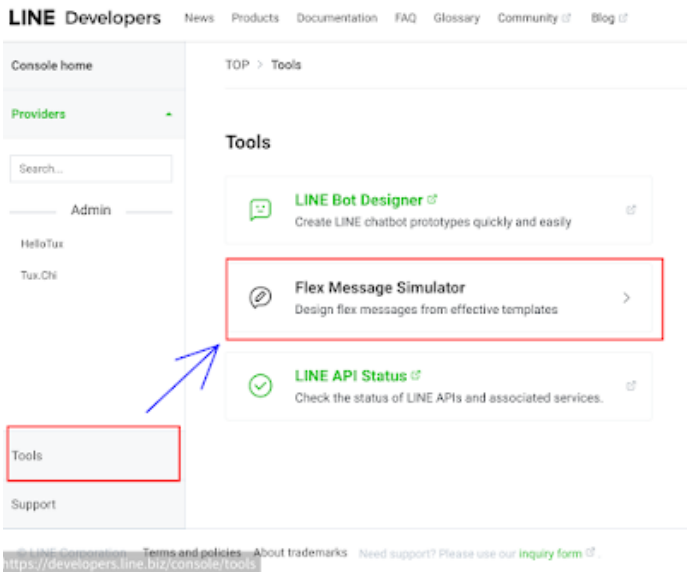
Line Bot 上的 FlexMessage 用法

學習目標：

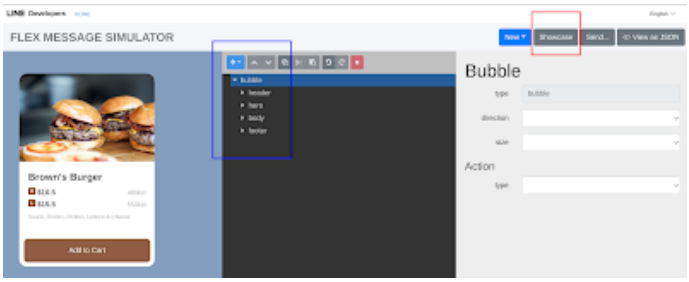
- 了解 Line Bot 上的 FlexMessage 用法！
- 設定 FlexMessage 連結 Line Bot 機器人資料庫！

產生 FlexMessage 程式碼

1. 登入 Line Developers ，查看 Flex Message Simulator



2. 利用「Showcase」選擇自己想要的範例！利用「View as JSON」可以查看檔案內容！



3. 新增 app/flexmodules 子目錄，方便存放Flex Message 程式！
4. 利用 Python 程式，產生所需要的 JSON 檔案格式，傳送給 Line 平台！
   例：flexmessages.py

```
def fleximage(url):
    return { "type": "image",
             "url": url,
             "size": "full",
             "aspectRatio": "20:13",
             "aspectMode": "cover"}
```

```
def flextext(text,size,color,weight='regular',wrap=False):
    return { "type": "text",
             "text": text,
             "size": size,
             "color": color,
             "weight": weight,
             "wrap": wrap }

def flexlogo(text='大學麵店'):
    return flextext(text, size='md',color='#066BAF', weight='bold')

def flextitle(text):
    return flextext(text, size='xl',color='#066BAF', weight='bold')

def flexhead(text):
    return flextext(text, size='xl',color='#555555')

def flexnote(text):
    return flextext(text, size='md',color='#AAAAAA', wrap=True)

def flexseparator(margin='xl'):
    return { "type": "separator", "margin": margin }

def flexbutton(label, data, display_text):
    return { "type": "button",
             "action": {
                 "type": "postback",
                 "label": label,
                 "data": data,
                 "display_text": display_text },
                 "style": "link",
                 "color": "#066BAF",
                 "height": "sm" }

def flex_index():
    hero_url = "https://scdn.line-apps.com/n/channel_devcenter/img/fx/01_2_restaurant.png"
    bodys = [flexlogo(),
                 flextitle('歡迎光臨'),
             flexseparator(),
             flexhead('菜單列表'),
             flexnote('# 查詢所有資料'),
             flexseparator()]
    footers = [flexbutton('菜單列表','菜單','菜單列表'),
               flexbutton('單項查詢','單項','單項菜單查詢') ]

    FlexMessage = {'type': 'bubble',
                   'hero': fleximage(hero_url),
                   'body': {
                       'type': 'box',
                       'layout': 'vertical',
                       'spacing': 'md',
                       'contents': bodys},
                   'footer': {
                       'type': 'box',
                       'layout': 'vertical',
                       'contents': footers}}
    return FlexMessage
```

5. 新增程式 app/flexmodules/flextalks.py ，回應 line bot 平台的訊息！

```
from app.flexmodules import flexmessages
from linebot.models import FlexSendMessage
```

```
from app import line_bot_api, handler

def query_menu(event):
    if '菜單查詢' in event.message.text:
        line_bot_api.reply_message(
            event.reply_token,
            FlexSendMessage(alt_text='query record: index',contents=flexmessages.flex_index())
        )
        return True
    else:
        return False
```

6. 修改 app/linebotmodules.py 裡的程式，加上呼叫 flextalks.py 的程式！

```
(前面略過....)
        replay = False

        if not replay:
            reply = flextalks.query_menu(event)
(後面略過....)
```

7. 上傳至 Heroky 專案內！進行測試！

製作 PostbackEvent 程式碼

1. 修改 app/linebotmodules.py 裡的程式！在最後一行加上下列程式碼：

```
(前面略過....)
# line 回應的訊息
@handler.add(PostbackEvent)
def handle_postback(event):
    print(event)
    flextalks.query_menu_back(event)
```

2. 修改程式 app/flexmodules/flextalks.py ，import 資料庫連結程式，並加入回應 line bot 平台的訊息！

```
(前面略過....)
from app import line_bot_api, handler
from app.dataSQL import connectDB
(中間略過....)
def query_menu_back(event):
    query = event.postback.data
    print(query)
    if '=' in query:
        print(query.split('=')[1])
        data = connectDB.queryItem(query.split('=')[1])
        menu_name = [i[2] for i in data]
        line_bot_api.reply_message(
            event.reply_token,
            FlexSendMessage(
                alt_text=f"query record: column {query}",
                contents= flexmessages.flex_menu_prize(query,menu_name)
            )
        )
        return True
    elif '菜單' in query:
        data = connectDB.showallMenu()
        menu_name = [i[1] for i in data]
        line_bot_api.reply_message(
            event.reply_token,
```

```
            FlexSendMessage(
                alt_text=f"query record: column {query}",
                contents= flexmessages.flex_menu(query,menu_name)
            )
        )
        return True
    else:
        return False
```

3. 修改程式 app/flexmodules/fflexmessages.py，在檔案最後面，加上下列程式：

```
(前面略過....)
def flex_menu(keywords, data):
    bodys = [flexlogo(),
                 flextitle(f'{keywords}'),
             flexseparator()]

    footers = [flexbutton(f"{i}",f"{keywords}={i}",f"查詢 {i}") for i in data]

    FlexMessage = {'type': 'bubble',
                   'body': {
                       'type': 'box',
                       'layout': 'vertical',
                       'spacing': 'md',
                       'contents': bodys},
                   'footer': {
                       'type': 'box',
                       'layout': 'vertical',
                       'contents': footers}}
    return FlexMessage

def flex_menu_prize(keywords, data):
    keyword = keywords.split('=')[1]
    bodys = [flexlogo(),
                 flextitle(f'{keyword}'),
             flexseparator(),
             flexnote(f'價格:{data}'),
             flexseparator()]

    footers = [flexbutton("菜單查詢","菜單查詢","菜單列表")]

    FlexMessage = {'type': 'bubble',
                   'body': {
                       'type': 'box',
                       'layout': 'vertical',
                       'spacing': 'md',
                       'contents': bodys},
                   'footer': {
                       'type': 'box',
                       'layout': 'vertical',
                       'contents': footers}}
    return FlexMessage
```

4. 修改 app/dataSQL/connectDB.py，加入單項查詢的資料庫語法：

```
(前面略過....)
def queryItem(keyword):
    DATABASE_URL = os.environ['DATABASE_URL']
    connection_db = psycopg2.connect(DATABASE_URL,sslmode='require')
    cursor = connection_db.cursor()
    SQL_cmd = f"""SELECT * FROM menu WHERE menuname = %s;"""
    cursor.execute(SQL_cmd,[keyword])
    connection_db.commit()
```

```
        raws = cursor.fetchall()
        data = []
        for raw in raws:
            data.append(raw)

        cursor.close()
        connection_db.close()
        return data
    (後面略過....)
```

5. 上傳 Heroku 專案後，進行測試！