重新設計應用程式

學習目標：

- 將檔案分解，藏暱重要的 Line Bot 聊天機器人機密
- 結構化開發的專案，以利將來的維護與發展

藏暱重要的機密

1. 在專案資料夾 LineBot下，新增 config.ini 檔案，寫人重要資訊

```
[LineBot]
channel_access_toke = Channel access token
channel_secret = Channel secret
```

2. 修改主要程式檔：main.py

```
(前面略過...)
import configparser

app = Flask(__name__)

# 導入 config.ini 檔案
config = configparser.ConfigParser()
config.read('config.ini')

# Line 聊天機器人的基本資料
line_bot_api = LineBotApi(config.get('LineBot','channel_access_toke'))
handler = WebhookHandler(config.get('LineBot','channel_secret'))
(以下略過...)
```

3. 將程式碼送上 Heroku 進行測試，應不會出現錯誤訊息

```
C:\LineBot>git add .
C:\LineBot>git commit -m "Modify Main.py"
C:\LineBot>git push keroku main
```

4. MessageEvent主要分類：
    - MessageEvent:訊息事件
        - TextMessage:文字訊息
        - ImageMessage:圖片訊息
        - VideoMessage:影音訊息
        - StickerMessage:貼圖訊息
        - FileMessage:檔案訊息
    - FollowEvent:加入好友事件

- UnfollowEvent:刪除好友事件
- 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

5. 試著加上一些程式碼，例：main.py

```
(前面的程式略過....)
# 鸚鵡學說話
@handler.add(MessageEvent, message=TextMessage)
def echo(event):
    # user id 在 Line Developers / Basic Setting 下
    if event.source.user_id == "Uf4a596a6eb65eabf52c003ffe325a21d":
        line_bot_api.reply_message(
            event.reply_token,
            TextSendMessage(text=event.message.text)
        )

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

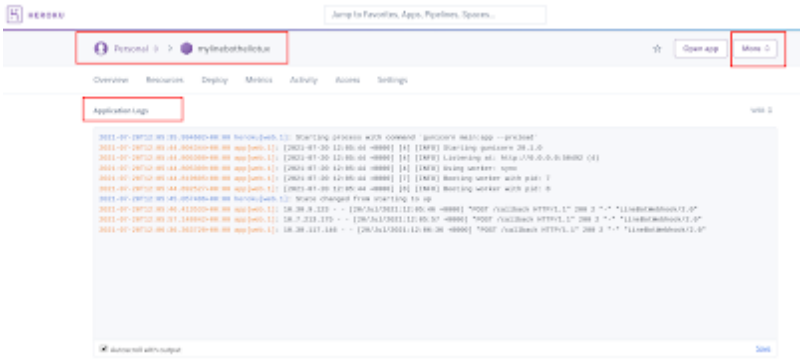
PS:將程式推上 Heroku ，再測看看！

學會查看重要記錄檔

1. 在 main.py 檔案中，寫入一小斷程式碼：

```
(前面略過....)
print(body)
try:
(後面略過....)
```

2. 利用 Heroku 的 app dashboard ，可查看 Server 上運作的訊息：



關閉自動訊息的回應

1. 如果不喜歡這個自動回應的訊息，可以關閉：

2. 靈活的改一段程式，顯現想要回應的訊息：main.py

```
(前面略過....)
# 靈活展現文字
```

```
@handler.add(MessageEvent, message=TextMessage)
def echo(event):
    # user id 在 Line Developers / Basic Setting 下
    if event.source.user_id == "Uf4a596a6eb65eabf52c003ffe325a21d":

        display_message = '你後面那位也想聽'

        if event.message.text == "tux來一個鬼故事":
            response_message = display_message

        line_bot_api.reply_message(
            event.reply_token,
            TextSendMessage(text=response_message)
        )

if __name__ == "__main__":
    app.run()
(後面略過....)
```

結構化開發專案的目錄設計

1. 將目錄結構化成下列形式，儘量把程式分門別類分開：

```
LineBot
  |-- Procfile
  |-- requirements.txt
  |-- runtime.txt
  |-- config.ini
  |-- main.py
  |-- app
       |-- __init__.py
       |-- router.py
       |-- linebotmodules.py
```

2. 修改 main.py ，將內容儘量清空：

```
from app import app

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

3. 新增 __init__.py 內容：

```
from flask import Flask, request, abort
from linebot import LineBotApi, WebhookHandler
import configparser

app = Flask(__name__)

# 導入 config.ini 檔案
config = configparser.ConfigParser()
config.read('config.ini')

# Line 聊天機器人的基本資料
line_bot_api = LineBotApi(config.get('LineBot','channel_access_token'))
handler = WebhookHandler(config.get('LineBot','channel_secret'))

# 導入其他的程式模組
from app import router, linebotmodules
```

4. 新增 router.py 內容：

```
from app import app, handler, request, abort
from linebot.exceptions import InvalidSignatureError

# 接收 Line 平台來的「通知」
@app.route("/callback", methods=['POST'])
def callback():
    signature = request.headers['X-Line-Signature']
    body = request.get_data(as_text=True)
    app.logger.info("Request body: " + body)

    print(body)
    try:
        handler.handle(body, signature)
    except InvalidSignatureError:
        abort(400)

    return 'OK'
```

5. 新增 linebotmodules.py 內容：

```
from app import line_bot_api, handler
from linebot.models import MessageEvent, TextMessage, TextSendMessage

# 靈活展現文字
@handler.add(MessageEvent, message=TextMessage)
def echo(event):
    # user id 在 Line Developers / Basic Setting 下
    if event.source.user_id == "Uf4a596a6eb65eabf52c003ffe325a21d":

        display_message = '你後面那位也想聽'

        if event.message.text == "tux來一個鬼故事":
            response_message = display_message

        line_bot_api.reply_message(
```

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

6. 將程式推上 Heroku ，再測看看！