CH06 Render Flask Web 表單應用

建立 Web 表單方式

- 使用 HTML 建立 Web 表單
- 使用 Flask-WTF 擴充套件

建立 HTML Web 表單

- 使用 POST 方式傳送資料，利用表單輸入資料
- 使用 POST 方式並且使用 Session，製作 Login 與 Logout 功能

登入表單製作

- 設定 base.html
- 設定 signin.html
- 設定 user.html
- 設定 app.py

設定 base.html

- 使用 Visual Studio Code 編輯「templates」資料夾下的「base.html」

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>{% block title %} {% endblock %}</title>
    <link href="{{ url_for('static', filename='css/bootstrap.min.css') }}" rel="stylesheet">
    {% block style %} {% endblock %}
</head>
<body>
    {% block content %}
    {% endblock %}

    <script src="{{ url_for('static', filename='js/jquery-3.7.1.min.js') }}"></script>
    <script src="{{ url_for('static', filename='js/popper-2.11.8.min.js') }}"></script>
    <script src="{{ url_for('static', filename='js/bootstrap.min.js') }}"></script>
</body>
</html>
```

設定 signin.html

- 使用 Visual Studio Code 編輯「templates/member」資料夾下的「signin.html」

```
{% extends "base.html" %}
{% block title %} GoGo智慧家庭-會員登入 {% endblock %}
{% block style %}
<style>
    div.container-fluid {
        display: flex;
        justify-content: center;
        align-items: center;
    }
    img {
        width: 30%;
        height: 50;
        margin: auto;
        display: block;
    }
</style>
{% endblock %}
```

設定 signin.html (Cont.)

```
18  {% block content %}
19  <div class="container-fluid">
20    <div class="card" style="width: 18rem;">
21      <img class="profile-img-card" src="{{ url_for('static', filename='img/user_2x.png') }}">
22      <div class="card-body">
23        <form class="form-horizontal" id="local-signin" action="/member/login" method="POST">
24          <div class="form-group">
25            <input type="text" class="form-control" id="login-username" name="username"
26              placeholder="帳號 ID" required autofocus>
27          </div>
28          <div class="form-group">
29            <input type="password" class="form-control" id="login-password" name="userpassword"
30              placeholder="密碼 Password" required>
31          </div>
32          <div id="remember" class="checkbox">
33            <label>
34              <input type="checkbox" value="remember-me">Remember me
35            </label>
36          </div>
37          <button type="submit" class="btn btn-lg btn-primary btn-block btn-signin">登入</button>
38        </form>
39      </div>
```

設定 signin.html (Cont.)

```
40              <div class="not-member">
41                  <p>還不是會員<a href="/member/signup">加入會員</a></p>
42                  <a href="/member/forgot-password" class="forgot-password">
43                      忘記密碼?
44                  </a>
45              </div>
46          </div>
47      </div>
48      {% endblock %}
```

設定 user.html

- 使用 Visual Studio Code 編輯「templates」資料夾下的「user.html」

```
1   {% extends "base.html" %}
2   {% block title %} GoGo智慧家庭-會員專區 {% endblock %}
3   {% block content %}
```

```
34      <h1 style="text-align: center">Hello, {{ name }}!</h1>
35      <p style="text-align: center">歡迎{{ name }}來到會員專區!</p>
36      <h1 style="text-align: center">{{ message }}</h1>
```

設定 app.py

- 使用 Visual Studio Code 編輯 app.py

```
1   from flask import Flask, render_template, request
2   from flask import redirect, url_for
3
4   app = Flask(__name__, template_folder='templates',
5               static_url_path='/static', static_folder='static')
6
```

```
25  @app.route('/user/<username>', methods=['GET'])
26  def user(username):
27      message = request.args.get('message')
28      return render_template('user.html', name=username, message=message)
29
```

設定 app.py (Cont.)

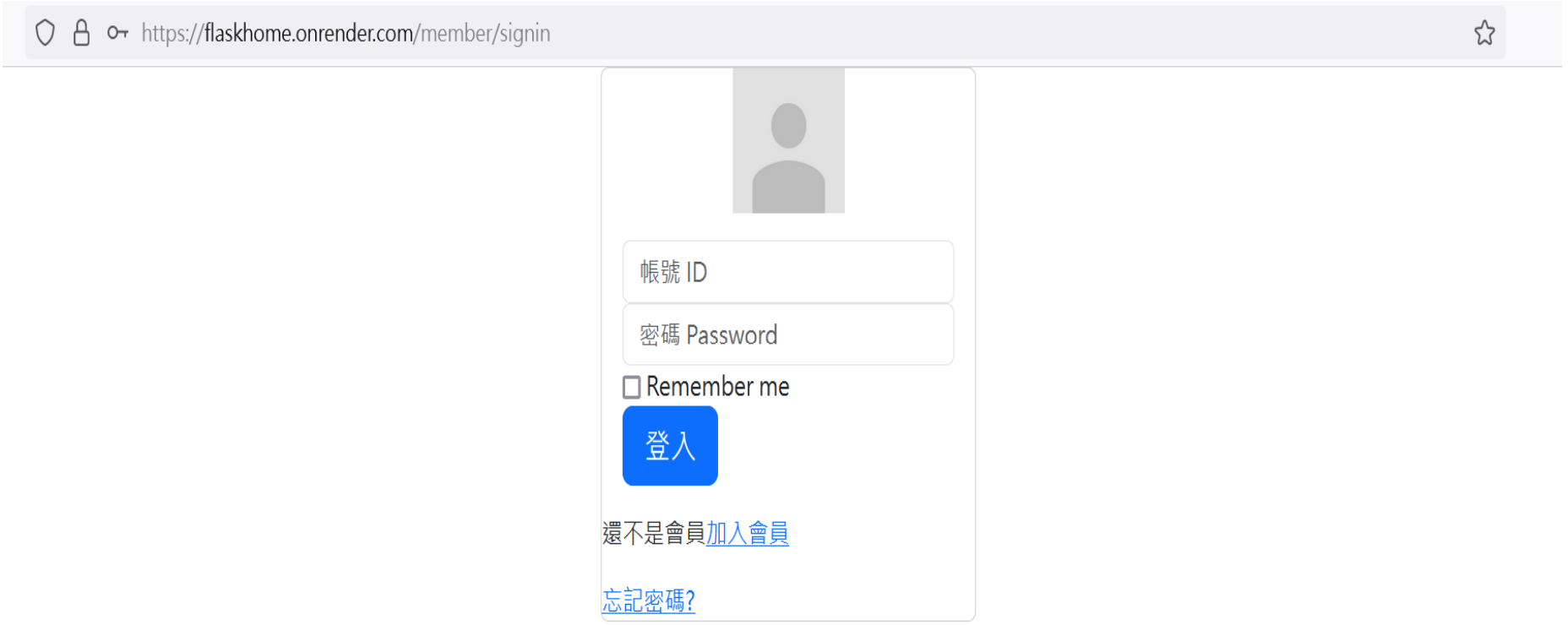
```
49  @app.route('/member/signin')
50  def signin():
51      return render_template('member/signin.html')
52
53  @app.route('/member/login', methods=["POST"])
54  def login():
55      if request.method == 'POST':
56          username = request.form['username']
57          userpassword = request.form['userpassword']
58          message = '登入成功!'
59          return redirect(url_for('user', username=username, message=message))
60      else:
61          render_template("member/signin.html")
```

上傳檔案

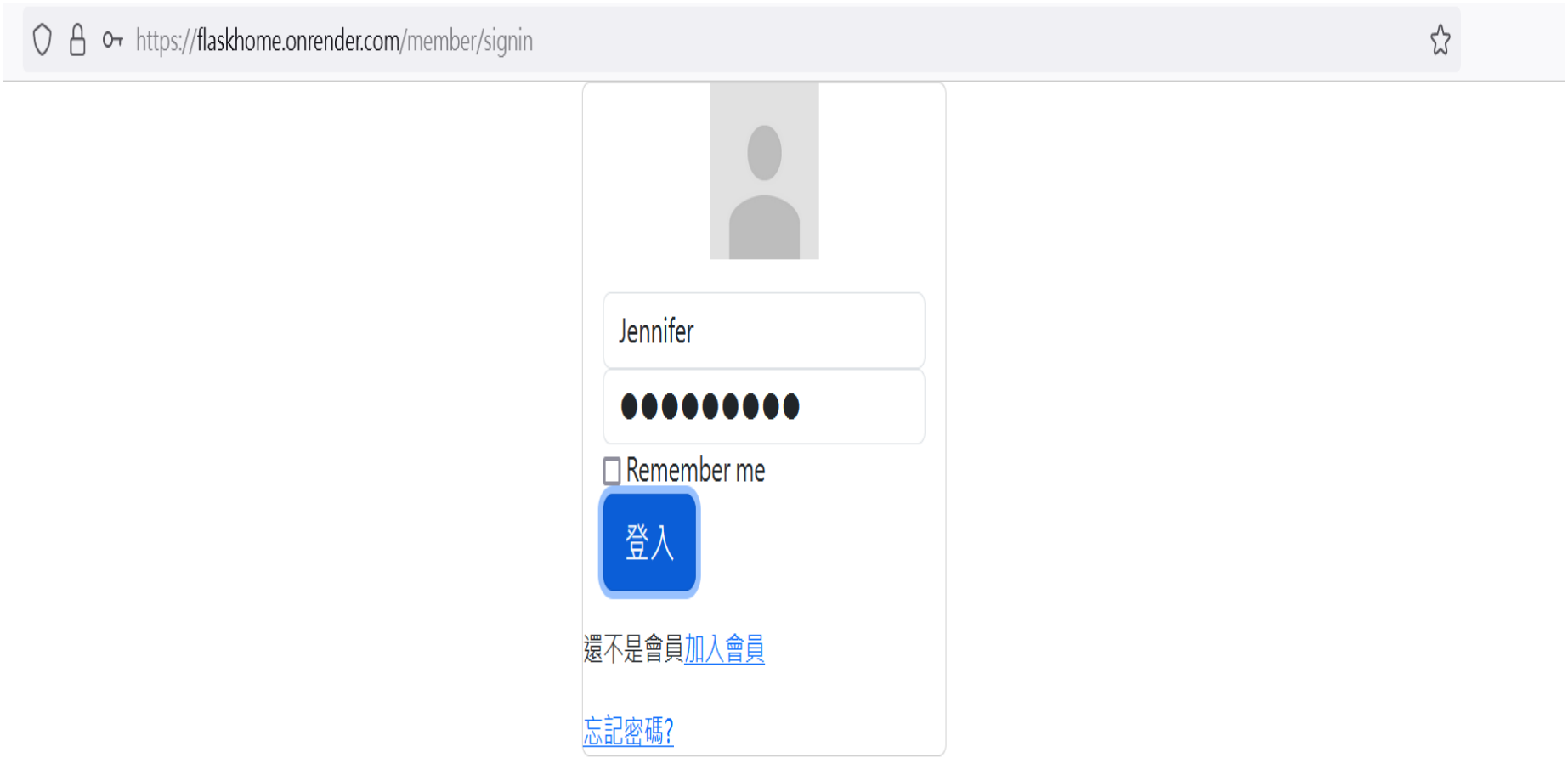
- $ git add .
- $ git commit -m "Tenth Commit"
- $ git push -u origin main

檢視 /member/signin 網頁

- https://flaskhome.onrender.com/member/signin



檢視 /member/signin 網頁 (Cont.)



檢視 /user/<username> 網頁

- https://flaskhome.onrender.com/user/Jennifer?message=%E7%99%BB%E5%85%A5%E6%88%90%E5%8A%9F!

使用 Session

- 當使用者登入一個網站時，Session 可以紀錄使用者登入 (Login) 的帳號資訊，當使用者瀏覽網站的其他頁面時，可以繼續使用 Session 資訊作為判斷
  - 使用者用每次進入一個頁面，不用重新再登入一次
- 當使用者離開網站或登出 (Logout) 網站時，Session 儲存的資訊會消失無蹤
- 不像 Cookie，Session 資訊儲存在伺服器 (Server) 端，相對來說比較安全

使用 Session (Cont.)

- 還需要加上一個 secret_key 的設定，否則視圖函式執行後會出現錯誤訊息
  - app.secret_key = 'Anything You Want'

帳號資料表

- 帳號資料表規劃
- 帳號資料表建立

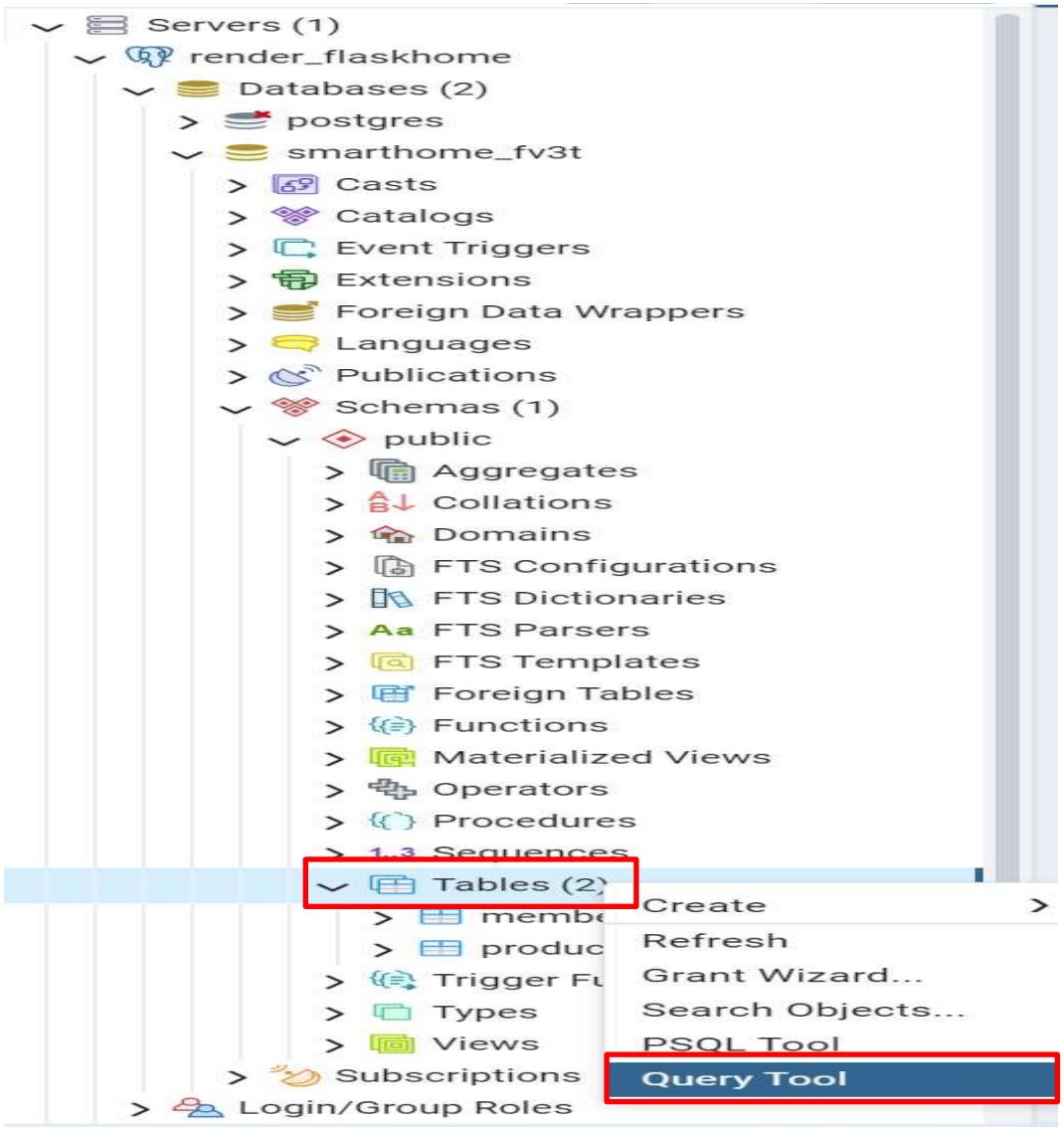
帳號資料表規劃

欄位名稱	資料型態	長度	是否允許 NULL	說明
aid (帳號編號)	INT		☐	自動遞增
mid (會員編號)	CHAR	5	☐	外來鍵
username (帳號)	VARCHAR	50	☐	
userpass (密碼)	VARCHAR	50	☐	

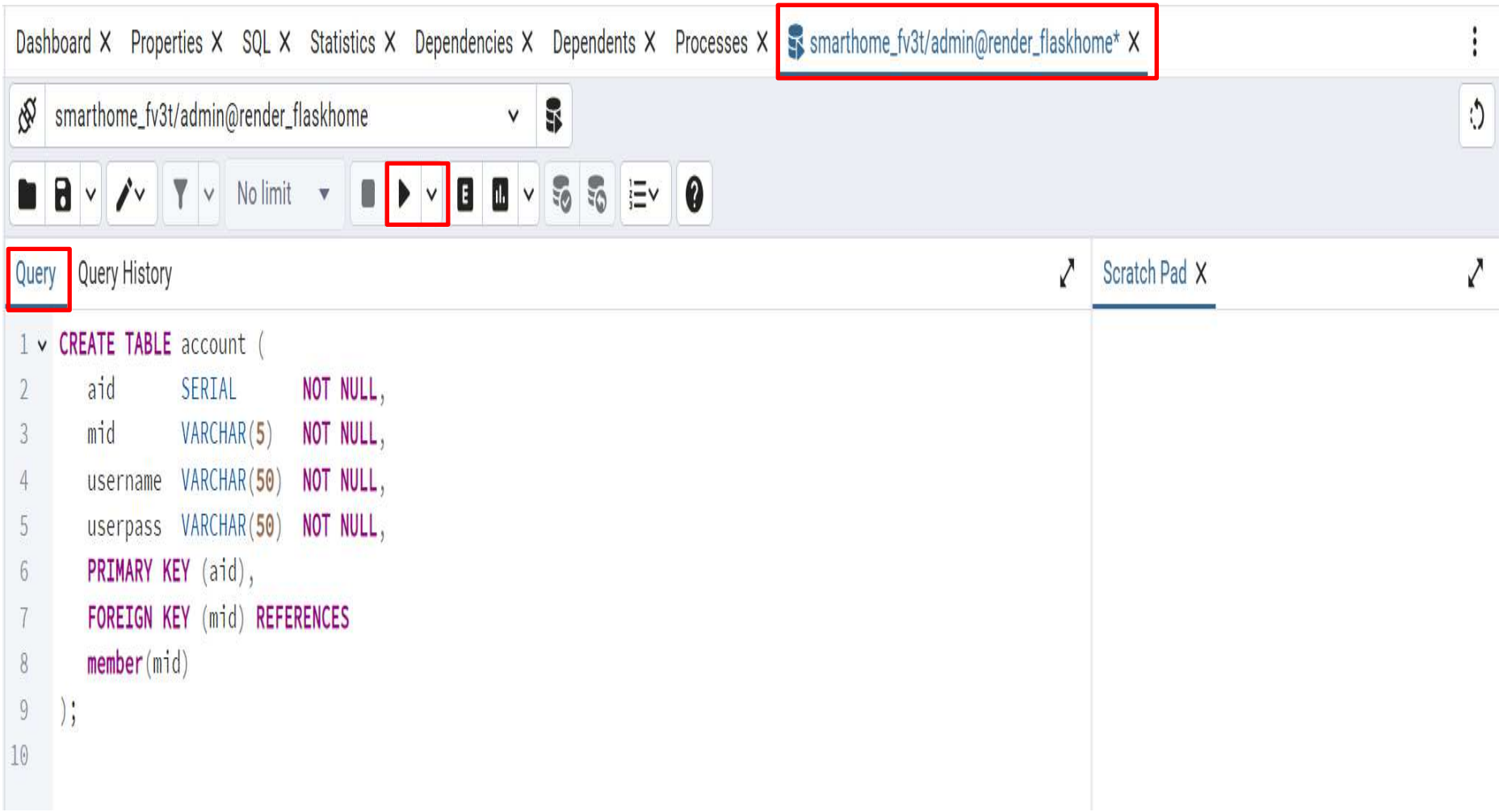
帳號資料表建立

- **CREATE TABLE** account (
  aid SERIAL NOT NULL,
  mid VARCHAR(5) NOT NULL,
  username VARCHAR(50) NOT NULL,
  userpass VARCHAR(50) NOT NULL,
  PRIMARY KEY (aid),
  FOREIGN KEY (mid) REFERENCES
  member(mid));

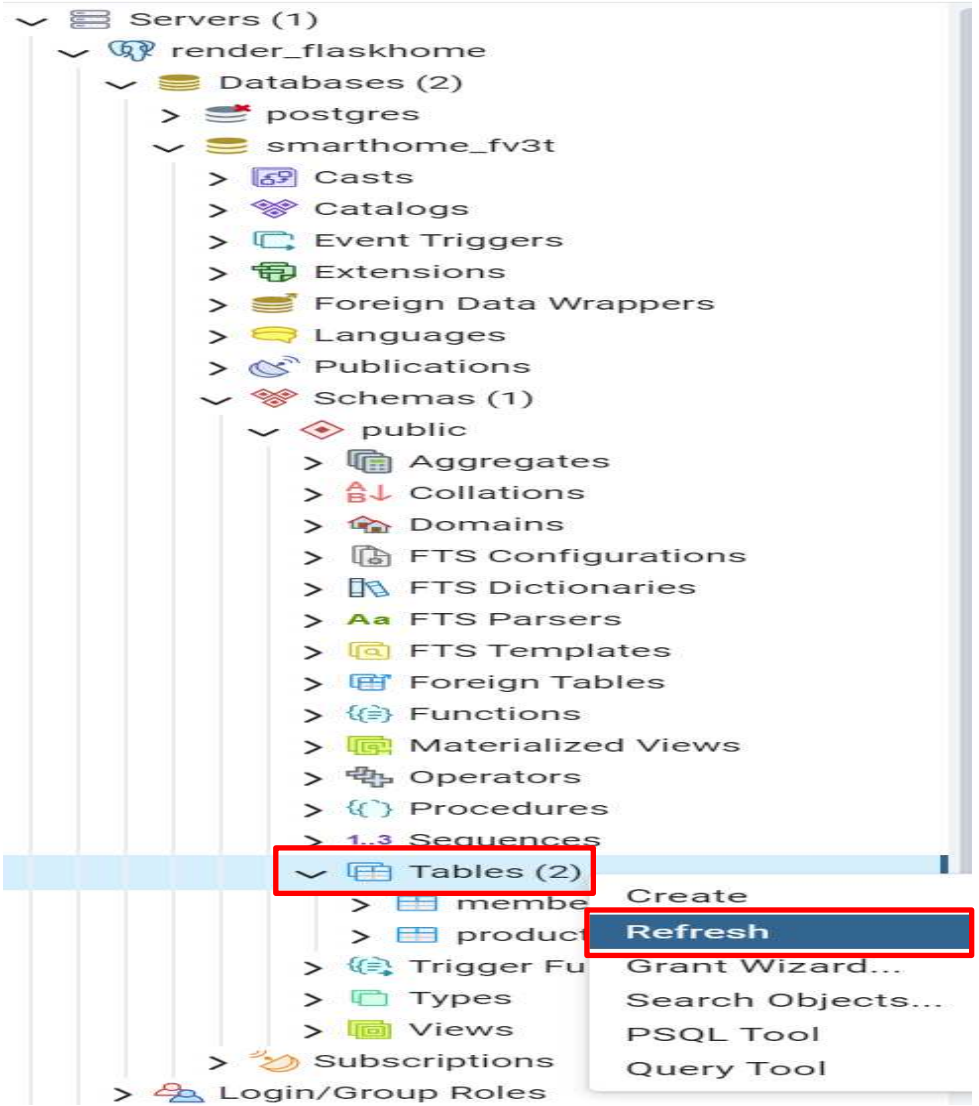
帳號資料表建立 (Cont.)



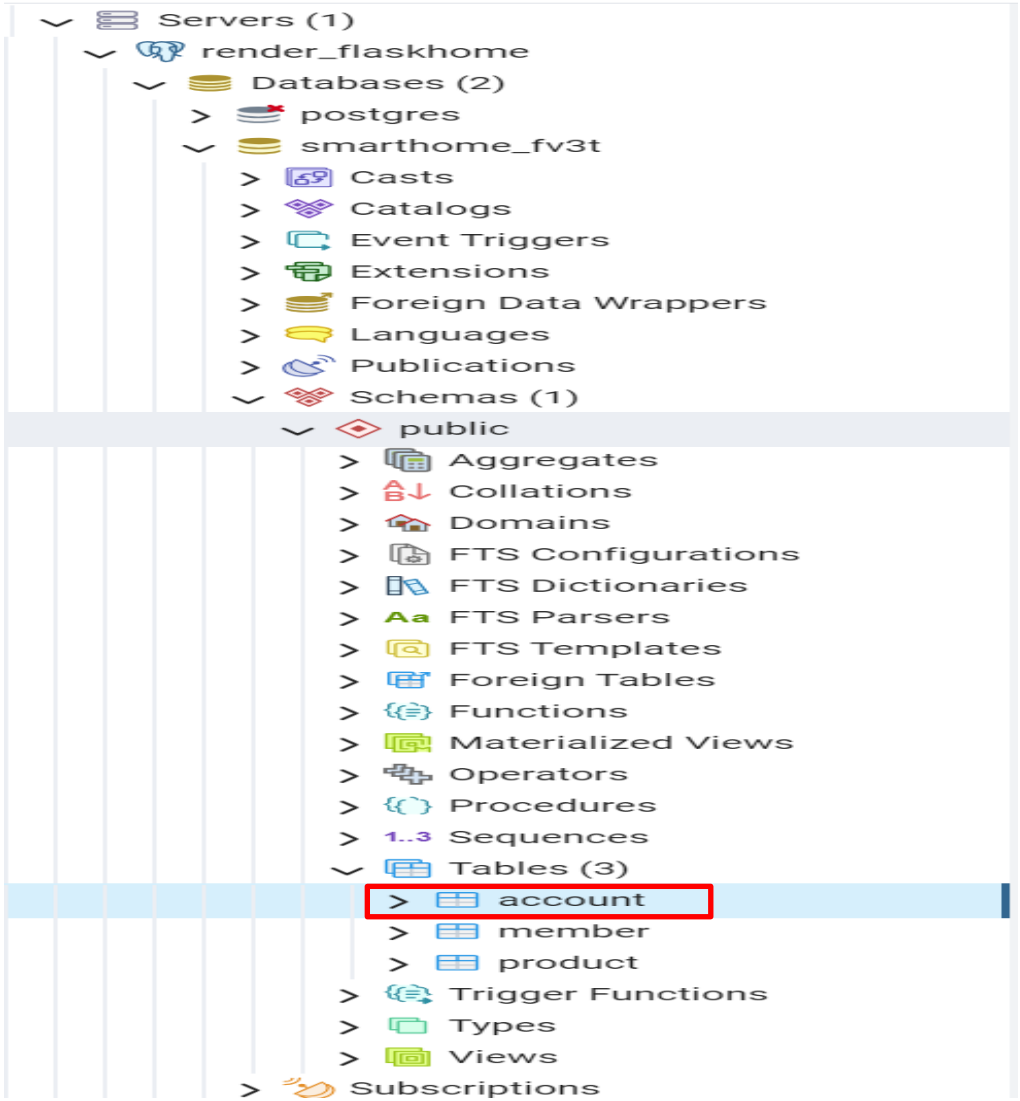
帳號資料表建立 (Cont.)



帳號資料表建立 (Cont.)



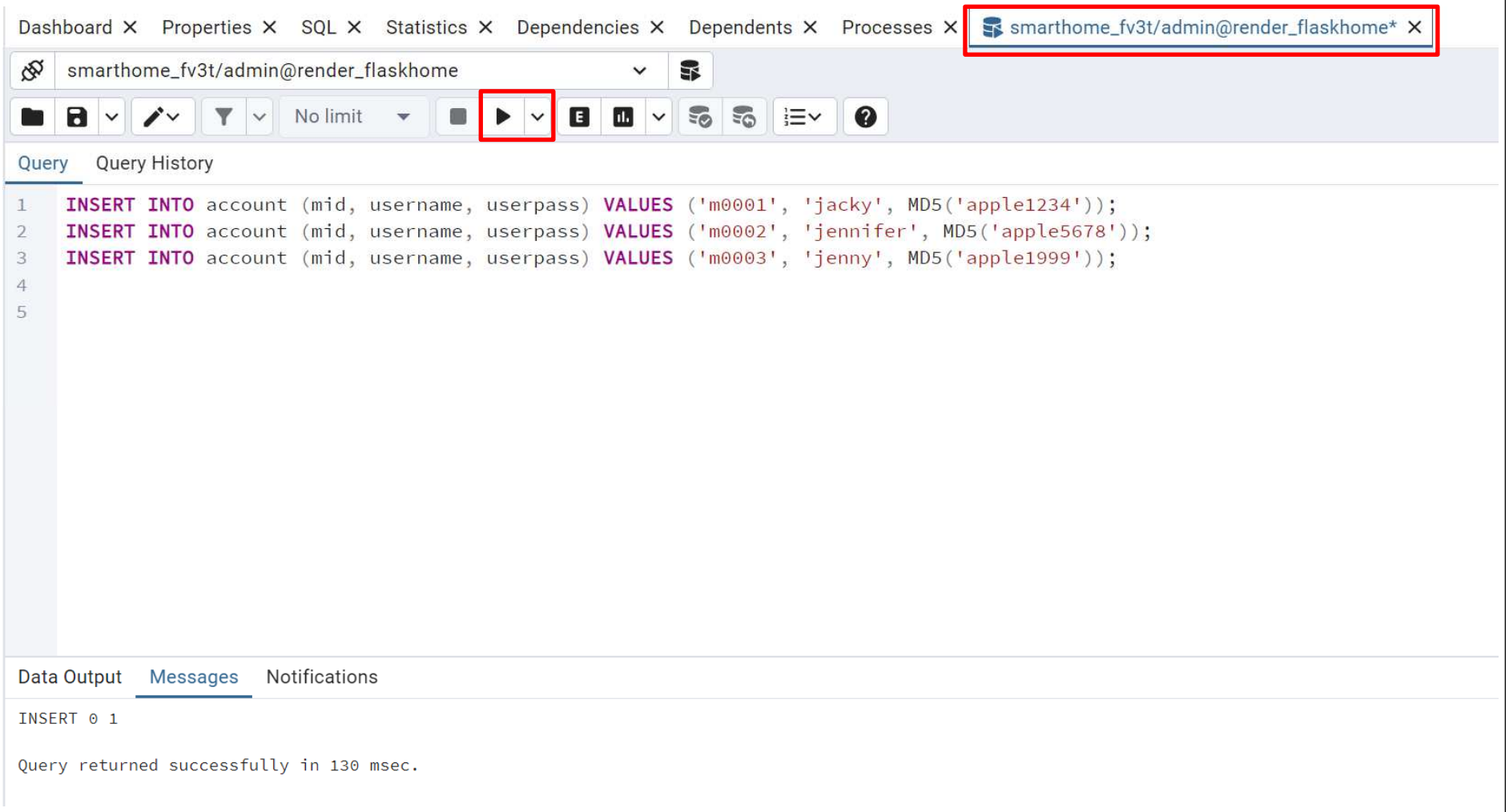
帳號資料表建立 (Cont.)



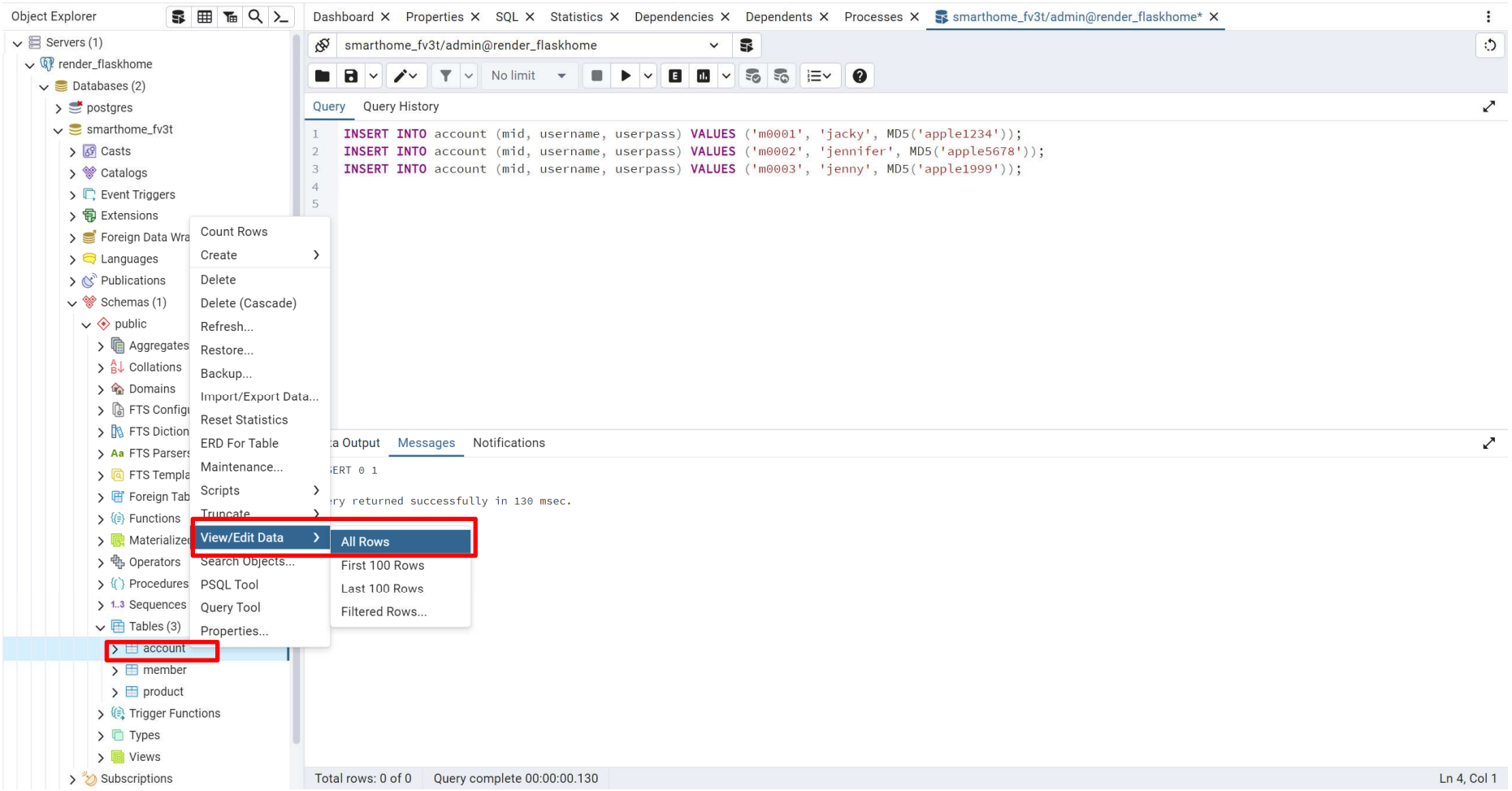
帳號資料表新增資料

- **INSERT INTO** account (mid, username, userpass) **VALUES** ('m0001', 'jacky', MD5('apple1234'));
- **INSERT INTO** account (mid, username, userpass) **VALUES** ('m0002', 'jennifer', MD5('apple5678'));
- **INSERT INTO** account (mid, username, userpass) **VALUES** ('m0003', 'jenny', MD5('apple1999'));

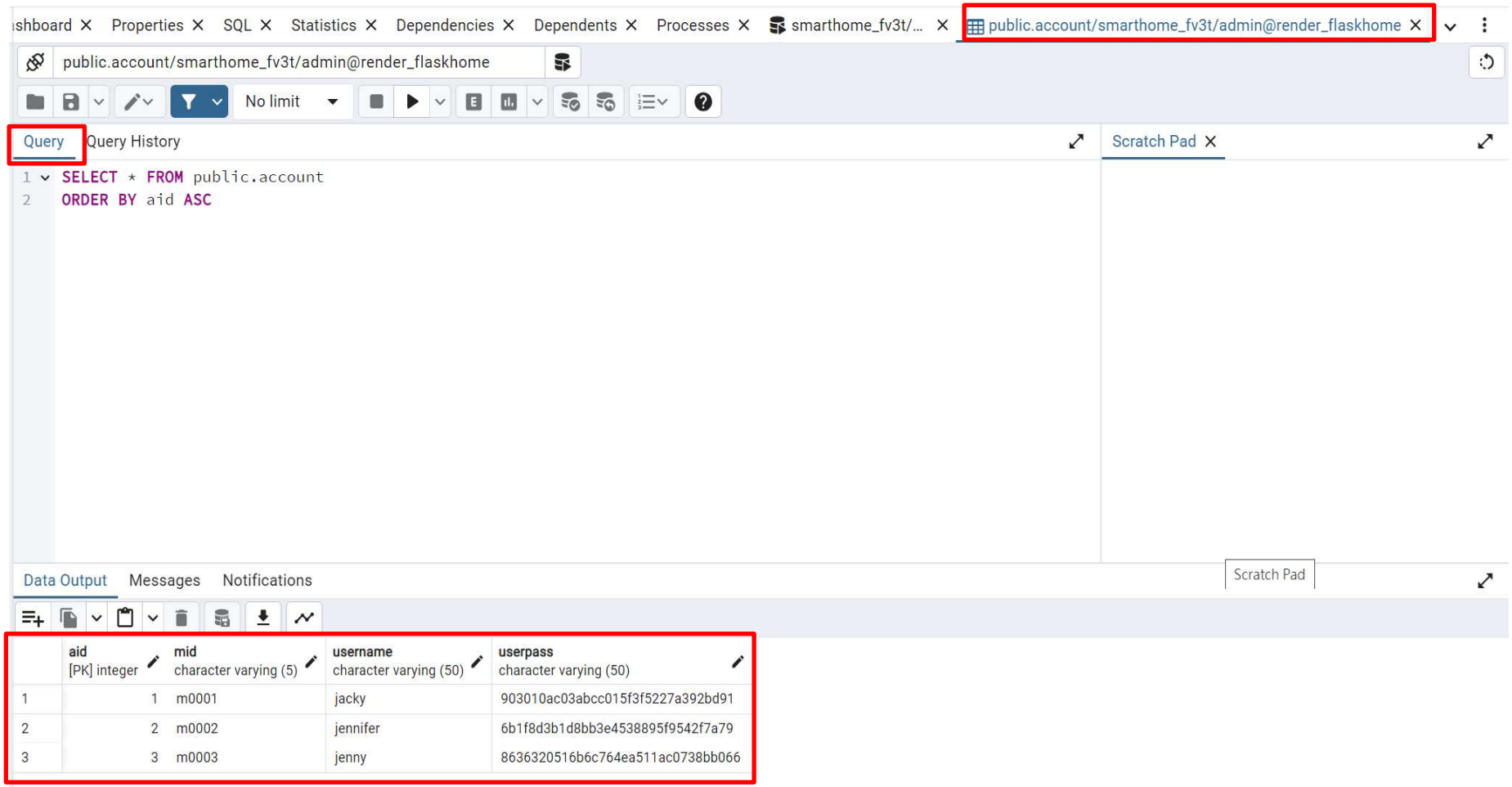
帳號資料表新增資料 (Cont.)



查詢帳號資料表


Object Explorer
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
smarthome_fv3t/admin@render_flaskhome*
Servers (1)
render_flaskhome
Databases (2)
postgres
smarthome_fv3t
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wra
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Config
FTS Diction
FTS Parsers
FTS Templa
Foreign Tab
Functions
Materialized
Operators
Procedures
Sequences
Tables (3)
account
member
product
Trigger Functions
Types
Views
Subscriptions
smarthome_fv3t/admin@render_flaskhome
No limit
Query
Query History
INSERT INTO account (mid, username, userpass) VALUES ('m0001', 'jacky', MD5('apple1234'));
INSERT INTO account (mid, username, userpass) VALUES ('m0002', 'jennifer', MD5('apple5678'));
INSERT INTO account (mid, username, userpass) VALUES ('m0003', 'jenny', MD5('apple1999'));
Count Rows
Create
Delete
Delete (Cascade)
Refresh...
Restore...
Backup...
Import/Export Data...
Reset Statistics
ERD For Table
Maintenance...
Scripts
Truncate
View/Edit Data
Search Objects...
PSQL Tool
Query Tool
Properties...
All Rows
First 100 Rows
Last 100 Rows
Filtered Rows...
Output
Messages
Notifications
ERT 0 1
ry returned successfully in 130 msec.
Total rows: 0 of 0
Query complete 00:00:00.130
Ln 4, Col 1

查詢帳號資料表 (Cont.)



設定 requirements.txt

- 使用 Visual Studio Code 編輯 requirements.txt
- 新增以下套件名稱
  - psycopg2-binary

```
1    flask
2    gunicorn
3    psycopg2-binary
```

設定 dbconn.py

- 使用 Visual Studio Code 在「flaskhome」資料夾下新增 dbconn.py 檔案

```
host = 'dpg-coj83lol6cac739sjuu0-a.singapore-postgres.render.com'
database = 'smarthome_fv3t'
user = 'admin'
password = 'Your Render PostgreSQL Password'
```

設定 app.py

- 使用 Visual Studio Code 編輯 app.py

```
1   from flask import Flask, render_template, request
2   from flask import redirect, url_for, session
3   import hashlib
4   import psycopg2
5   import dbconn
6
7   app = Flask(__name__, template_folder='templates',
8               static_url_path='/static', static_folder='static')
9   app.secret_key = 'fd4723e200261a2271ea912571eaaa1d'
10
```

設定 app.py (Cont.)

```
11    # DB Connection
12    def get_db_connection():
13        conn = psycopg2.connect(
14            host=dbconn.host,
15            database=dbconn.database,
16            user=dbconn.user,
17            password=dbconn.password)
18
19        return conn
20
```

設定 app.py (Cont.)

```
39  @app.route('/user')
40  def user():
41      if 'username' in session:
42          username = session['username']
43          return render_template('user.html', name=username)
44      else:
45          return redirect(url_for('signin'))
46
```

設定 app.py (Cont.)

```
47  @app.route('/search', methods=['GET', 'POST'])
48  def search():
49      if 'username' in session:
50          username = session['username']
51
52      if request.method == 'POST':
53          keyword = request.values['keyword']
54          if keyword == '紅燈':
55              message = '紅燈停!'
56          elif keyword == '黃燈':
57              message = '加速通過馬路或停下等候綠燈!'
58          elif keyword == '綠燈':
59              message = '綠燈行!'
60          else:
61              message = '請重新輸入!'
62          return render_template('user.html', name=username, message=message)
63      else:
64          message = '請使用HTTP POST傳送資料!'
65          return render_template('result.html', message=message)
66
```

設定 app.py (Cont.)

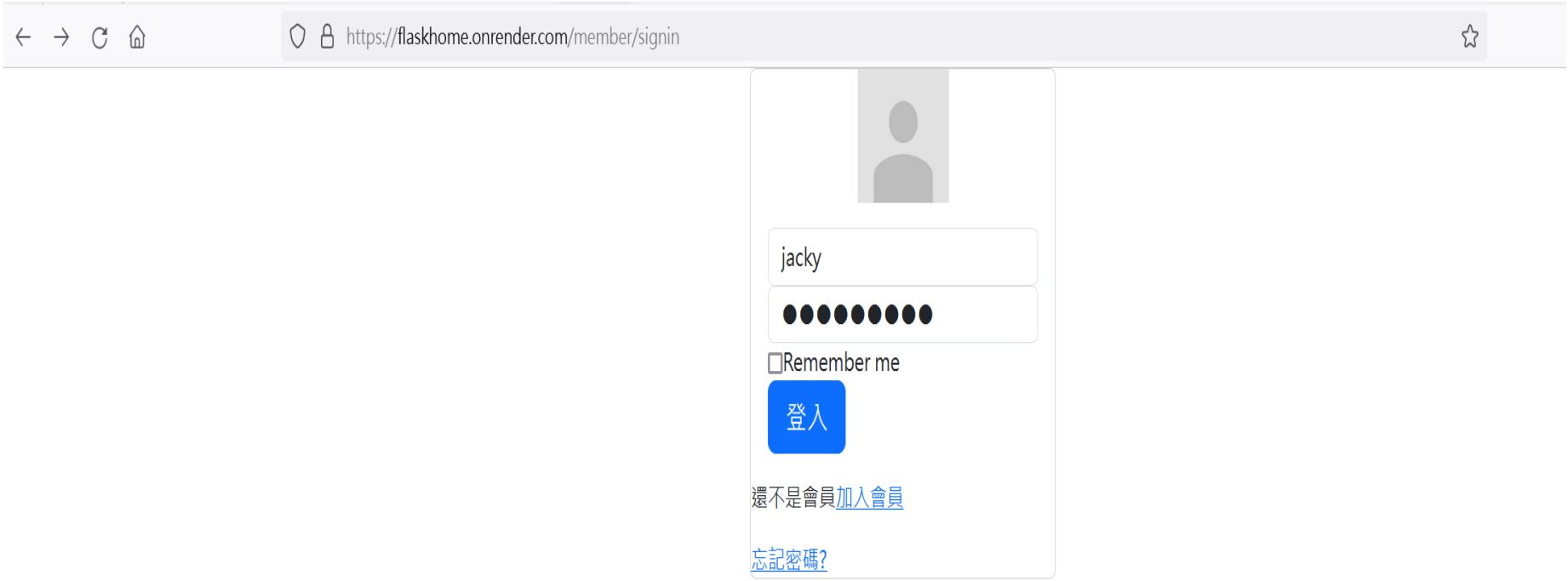
```
71  @app.route('/member/login', methods=["POST"])
72  def login():
73      if request.method == 'POST':
74          username = request.form['username']
75          userpass = request.form['userpassword']
76          md = hashlib.md5()
77          md.update(userpass.encode('utf-8'))
78          hashpass = md.hexdigest()
79
80          conn = get_db_connection()
81          cursor = conn.cursor()
82          SQL = f"SELECT username,userpass FROM account WHERE username='{username}';"
83          cursor.execute(SQL)
84          user = cursor.fetchone()
85          cursor.close()
86          conn.close()
87
88          if (username == user[0] and hashpass == user[1]):
89              session['username'] = username
90              return redirect(url_for('user'))
91      else:
92          render_template("member/signin.html")
```

上傳檔案

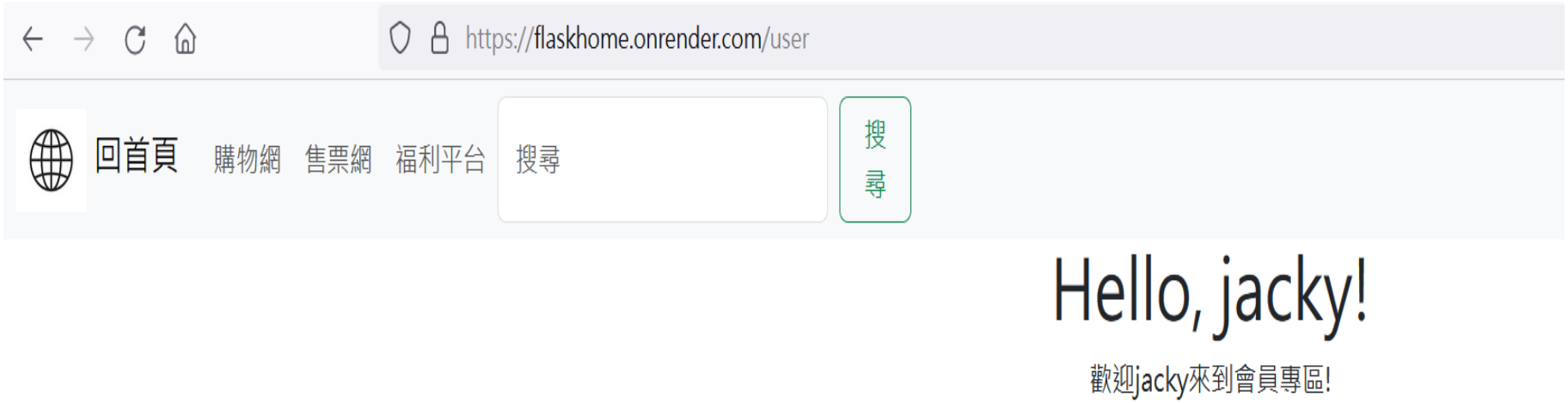
- $ git add .
- $ git commit -m "Eleventh Commit"
- $ git push -u origin main

檢視 /member/signin 網頁

- https://flaskhome.onrender.com/member/signin



檢視 /user 網頁


https://flaskhome.onrender.com/user
回首頁
購物網
售票網
福利平台
搜尋
搜尋
Hello, jacky!
歡迎jacky來到會員專區!

檢視 /user 網頁 (Cont.)

檢視 /search 網頁


https://flaskhome.onrender.com/search
回首頁
購物網
售票網
福利平台
搜尋
搜尋
Hello, jacky!
歡迎jacky來到會員專區!
加速通過馬路或停下等候綠燈!

關閉 Session

- 使用者手動關閉瀏覽器即可關閉 Session
- 若希望在不關閉瀏覽器的狀態下關閉 Session，可以使用「登出」的方式達成
- 當使用者進入 Logout 頁面時，使用 pop() 函式將 Session 裡面記錄的 username 清除
  - session.pop('username', None)
- 當 Session 紀錄清除後，將頁面導向 signin 頁面

關閉 Session (Cont.)

- 為配合 logout() 視圖函式的使用，需要修改 signin() 視圖函式，使已經有 Session 紀錄 (aession[username]) 的使用者，從路由 signin() 視圖函式，直接被導向 user() 視圖函式載入的頁面

設定 app.py

- 使用 Visual Studio Code 編輯 app.py

```
94    @app.route('/member/logout')
95    def logout():
96        session.pop('username', None)
97        return redirect(url_for('signin'))
```

設定 app.py (Cont.)

```
71  @app.route('/member/login', methods=["POST"])
72  def login():
73      if request.method == 'POST':
74          username = request.form['username']
75          userpass = request.form['userpassword']
76          md = hashlib.md5()
77          md.update(userpass.encode('utf-8'))
78          hashpass = md.hexdigest()
79
80          conn = get_db_connection()
81          cursor = conn.cursor()
82          SQL = f"SELECT username,userpass FROM account WHERE username='{username}';"
83          cursor.execute(SQL)
84          user = cursor.fetchone()
85          cursor.close()
86          conn.close()
87
```

設定 app.py (Cont.)

```
88          if (username == user[0] and hashpass == user[1]):
89              session['username'] = username
90              return redirect(url_for('user'))
91      else:
92          if 'username' in session:
93              return redirect(url_for('user'))
94
95      render_template('member/signin.html')
96
```

上傳檔案

- $ git add .
- $ git commit -m "Twelfth Commit"
- $ git push -u origin main

持續 Session 效果一段時間

- 目前使用的 Session，只要關閉瀏覽器，Session 的記憶即會消失
- 有時候，希望 Session 不因為關閉瀏覽器而消失並且效力可以持續一段特定的時間
- 首先，需要匯入 timedelta 套件
  - from datetime import timedelta

持續 Session 效果一段時間 (Cont.)

- 接著，設定 Session 的存續期間
  - 以「分」(minutes) 為單位設定 (minutes=5)
  - 以「天」(days) 為單位設定 (days=7)
  - app.permanent_session_lifetime = timedelta(minutes=3)
- 接著，在 signin() 視圖函式的加入以下設定使生效
  - session.permanent = True

設定 app.py

- 使用 Visual Studio Code 編輯 app.py

```
1   from flask import Flask, render_template, request
2   from flask import redirect, url_for, session
3   from datetime import timedelta
4   import hashlib
5   import psycopg2
6   import dbconn
7
8   app = Flask(__name__, template_folder='templates',
9               static_url_path='/static', static_folder='static')
10  app.secret_key = 'fd4723e200261a2271ea912571eaaa1d'
11  app.permanent_session_lifetime = timedelta(minutes=3)
12
```

設定 app.py (Cont.)

```
73  @app.route('/member/login', methods=["POST"])
74  def login():
75      if request.method == 'POST':
76          username = request.form['username']
77          userpass = request.form['userpassword']
78          md = hashlib.md5()
79          md.update(userpass.encode('utf-8'))
80          hashpass = md.hexdigest()
81
82          conn = get_db_connection()
83          cursor = conn.cursor()
84          SQL = f"SELECT username,userpass FROM account WHERE username='{username}';"
85          cursor.execute(SQL)
86          user = cursor.fetchone()
87          cursor.close()
88          conn.close()
89
```

設定 app.py (Cont.)

```
90          if (username == user[0] and hashpass == user[1]):
91              session.permanent = True
92              session['username'] = username
93              return redirect(url_for('user'))
94      else:
95          if 'username' in session:
96              return redirect(url_for('user'))
97
98      render_template('member/signin.html')
99
```

WTForms

- 一個 Python 套件
- 提供 Python 開發者在開發網站時，需要的表單呈現以及表單的驗證等功能
- 可以與選擇的網站框架與樣板引擎一起使用，並支援資料的驗證、CSRF (Cross-Site Request Forgery，跨站請求偽造) 的保護、I18N 多國化等等功能

WTForms 核心概念

- Forms 表單為 WTForms 最主要的核心容器
  - Forms 表單為 Fields 欄位的集合
  - 可以選擇透過字典 (dictionary) 或屬性 (attribute) 的方式接觸
- Fields 將大部分粗重的工作攬在身上
  - 每個 Fields 欄位都代表某一種資料的類型
  - Fields 欄位限制使用者僅能夠輸入符合該資料類型的資料
  - Fields 欄位除包含資料外，還包含 label (標籤)，description (描述) 與錯誤驗證等屬性

上傳檔案

- $ git add .
- $ git commit -m "Thirteenth Commit"
- $ git push -u origin main

WTForms 核心概念 (Cont.)

- 每個 Fields 欄位都有一個 widget 實例
  - Widget 的工作為呈現該 Fields 欄位對應的 HTML 標籤
  - 可以指定 Widget 實例給每個特定的 Fields 欄位
  - 預設情況下每個 Fields 欄位都有一個 Widget 實例
- 有一些 Fields 欄位設置的目的只是為使工程師使用上方便
  - TextAreaField 只是一個字串欄位 (StringField)，預設的 Widget 為 TextArea

WTForms 核心概念 (Cont.)

- 為提供各種驗證規則，Fields 欄位包含一系列的驗證方式

WTForms 支援欄位清單

類型	說明
StringField	字串欄位
TextAreaField	多行字串欄位
PasswordField	密碼欄位 (出現 *)
HiddenField	隱藏欄位
DateField	日期欄位 datetime.date
DateTimeField	日期時間欄位 datetime.datetime

WTForms 支援欄位清單 (Cont.)

類型	說明
IntegerField	整數欄位
DecimalField	精度數值欄位 decimal.Decimal
FloatField	浮點數欄位
BooleanField	布林欄位
RadioField	單選 Radio
SelectField	下拉表單

WTForms 支援欄位清單 (Cont.)

類型	說明
SelectMultipleField	下拉表單 (可多選)
FileField	文件上傳欄位
SubmitField	提交表單按鈕
FormField	Form 中有 Form
FieldList	可提交多行表單資料

WTForms 驗證函式

函式	說明
Email	驗證電子郵件欄位
EqualTo	比較兩欄位數值是否相同
IPAddress	驗證 IP 位址
Length	驗證字串長度
NumberRange	驗證數值區間
Optional	欄位沒有被填寫時，不要進行其他的驗證

WTForms 驗證函式 (Cont.)

函式	說明
DataRequired	必填欄位
Regexp	以正則式驗證
URL	驗證網址
AnyOf	驗證輸入值在列表中
NoneOf	驗證輸入值不在列表中

Flask-WTF

- 一個 Python 套件
- 將 Flask 應用程式與 WTForms 整合在一起
- 包括檔案上傳、CSRF 以及 reCAPTCHA 驗證等，解決 WTForms 無檔案上傳功能的問題

使用 Flask-WTF

- 匯入 Flask-WTF 套件
  - from flask_wtf import FlaskForm
- 匯入 WTForms 套件
  - 只要匯入欄位，例如 StringField、SubmitField 等
  - 如果需要欄位驗證的話，匯入 validators 驗證欄位
  - from wtforms import StringField, SubmitField
  - from wtforms.validators import DataRequired
- 建立表單

Flask-WTF 會員註冊表單實作

- 設定 requirements.txt
- 設定 app.py
- 設定 signup.html

設定 requirements.txt

- 使用 Visual Studio Code 編輯 requirements.txt
- 新增以下套件名稱
  - WTForms
  - Flask-WTF

```
1    flask
2    gunicorn
3    psycopg2-binary
4    WTForms
5    Flask-WTF
```

設定 app.py

- 使用 Visual Studio Code 編輯 app.py

```
1   from flask import Flask, render_template, request
2   from flask import redirect, url_for, session
3   from flask_wtf import FlaskForm
4   from wtforms import StringField, DateField, PasswordField, SubmitField, validators
5   from datetime import timedelta
6   import hashlib
7   import psycopg2
8   import psycopg2.extras
9   import dbconn
10
11  app = Flask(__name__, template_folder='templates',
12              static_url_path='/static', static_folder='static')
13  app.secret_key = 'fd4723e200261a2271ea912571eaaa1d'
14  app.permanent_session_lifetime = timedelta(minutes=3)
15
```

設定 app.py (Cont.)

```
26  # Registration Form
27  class RegistrationForm(FlaskForm):
28      username = StringField('帳號', [validators.DataRequired(),
29          validators.Length(min=4, max=50)])
30      userpass = PasswordField('密碼', [validators.DataRequired(),
31          validators.Length(min=8, max=50),
32          validators.EqualTo('confirm', message='密碼必須與確認密碼一樣')])
33      confirm = PasswordField('確認密碼')
34      name = StringField('姓名', [validators.DataRequired(), validators.Length(min=4, max=8)])
35      birthday = DateField('生日', format='%Y-%m-%d')
36      phone = StringField('電話', [validators.DataRequired(),
37          validators.Length(min=7, max=13)])
38      address = StringField('地址',  [validators.Length(min=6, max=50)])
39      email = StringField('電子郵件', [validators.DataRequired(), validators.Length(min=6, max=50)])
40      submit = SubmitField('立即註冊')
41
```

設定 app.py (Cont.)

```
92  @app.route('/member/login', methods=["POST"])
93  def login():
94      if request.method == 'POST':
95          username = request.form['username']
96          userpass = request.form['userpassword']
97          md = hashlib.md5()
98          md.update(userpass.encode('utf-8'))
99          hashpass = md.hexdigest()
100
101         conn = get_db_connection()
102         cursor = conn.cursor(cursor_factory=psycopg2.extras.RealDictCursor)
103         SQL = f"SELECT username,userpass FROM account WHERE username='{username}';"
104         cursor.execute(SQL)
105         user = cursor.fetchone()
106         cursor.close()
107         conn.close()
108
```

設定 app.py (Cont.)

```
109             if (username == user['username'] and hashpass == user['userpass']):
110                 session.permanent = True
111                 session['username'] = username
112                 return redirect(url_for('user'))
113         else:
114             if 'username' in session:
115                 return redirect(url_for('user'))
116 
117         render_template('member/signin.html')
118 
```

設定 app.py (Cont.)

```
119  @app.route('/member/signup')
120  def signup():
121      regform = RegistrationForm()
122      return render_template('member/signup.html', form=regform)
123
124  @app.route('/member/join', methods=["POST"])
125  def join():
126      if request.method == 'POST':
127          regform = RegistrationForm()
128          if regform.validate_on_submit():
129              username = regform.username.data
130              userpass = regform.userpass.data
131              name = regform.name.data
132              birthday = regform.birthday.data
133              phone = regform.phone.data
134              address = regform.address.data
135              email = regform.email.data
136
```

設定 app.py (Cont.)

```
137             conn = get_db_connection()
138             cursor = conn.cursor()
139 
140             SQL = "SELECT COUNT(*) FROM member"
141             cursor.execute(SQL)
142             count = cursor.fetchone()[0]
143             mid = 'm' + str(count + 1).zfill(4)
144 
145             SQL2 = f"INSERT INTO member VALUES('{mid}','{name}','{birthday}','{phone}','{address}','{email}');"
146             cursor.execute(SQL2)
147 
148             SQL3 = f"INSERT INTO account(mid,username,userpass) VALUES('{mid}','{username}','{userpass}');"
149             cursor.execute(SQL3)
150 
151             conn.commit()
152             cursor.close()
153             conn.close()
154 
155         return redirect(url_for('signin'))
156 
```

設定 app.py (Cont.)

```
157    @app.route('/member/forgot')
158    def forgot():
159        return 'Pass'
160
161    @app.route('/member/logout')
162    def logout():
163        session.pop('username', None)
164        return redirect(url_for('signin'))
```

設定 signup.html

- 使用 Visual Studio Code 編輯「templates/member」資料夾下的「signup.html」

```
{% extends "base.html" %}
{% block title %} GoGo智慧家庭-會員註冊 {% endblock %}
{% block content %}
<div class="container-fluid">
    <h1 style="text-align: center;">GoGo智慧家庭-填寫註冊資料</h1>
    <form method="POST" action="{{ url_for('join') }}">
        {{ form.hidden_tag() }}
        {{ form.username.label(class='form-label') }}
        {{ form.username(class='form-control') }}
        <br>
        {{ form.userpass.label(class='form-label') }}
        {{ form.userpass(class='form-control') }}
        <br>
        {{ form.confirm.label(class='form-label') }}
        {{ form.confirm(class='form-control') }}
        <br>
        {{ form.name.label(class='form-label') }}
        {{ form.name(class='form-control') }}
        <br>
```

設定 signup.html (Cont.)

```
20              {{ form.birthday.label(class='form-label') }}
21              {{ form.birthday(class='form-control') }}
22              <br>
23              {{ form.phone.label(class='form-label') }}
24              {{ form.phone(class='form-label') }}
25              <br>
26              {{ form.address.label(class='form-label') }}
27              {{ form.address(class='form-control') }}
28              <br>
29              {{ form.email.label(class='form-label')}}
30              {{ form.email(class='form-control') }}
31              <br>
32              {{ form.submit(class='btn btn-primary') }}
33          </form>
34      </div>
35      {% endblock %}
```

設定 signin.html

- 使用 Visual Studio Code 編輯「templates/member」資料夾下的「signin.html」

```
18  {% block content %}
19  <div class="container-fluid">
20    <div class="card" style="width: 18rem;">
21      <img class="profile-img-card" src="{{ url_for('static', filename='img/user_2x.png') }}">
22      <div class="card-body">
23        <form class="form-horizontal" id="local-signin" action="{{ url_for('login') }}" method="POST">
24          <div class="form-group">
```

設定 signin.html (Cont.)

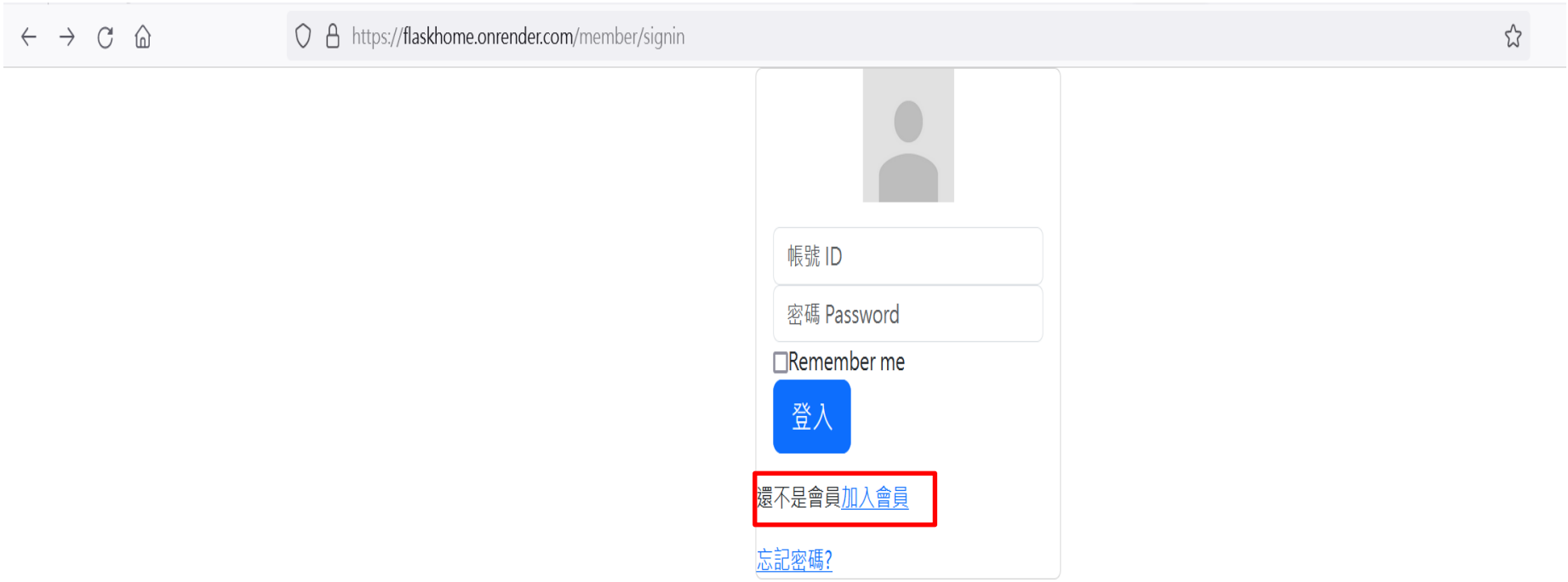
```
40              <div class="not-member">
41                  <p>還不是會員<a href="{{ url_for('signup') }}">加入會員</a></p>
42                  <a href="{{ url_for('forgot') }}" class="forgot-password">
43                      忘記密碼?
44                  </a>
45              </div>
46          </div>
47      </div>
48      {% endblock %}
```

上傳檔案

- $ git add .
- $ git commit -m "Fourteenth Commit"
- $ git push -u origin main

檢視 /member/signin 網頁

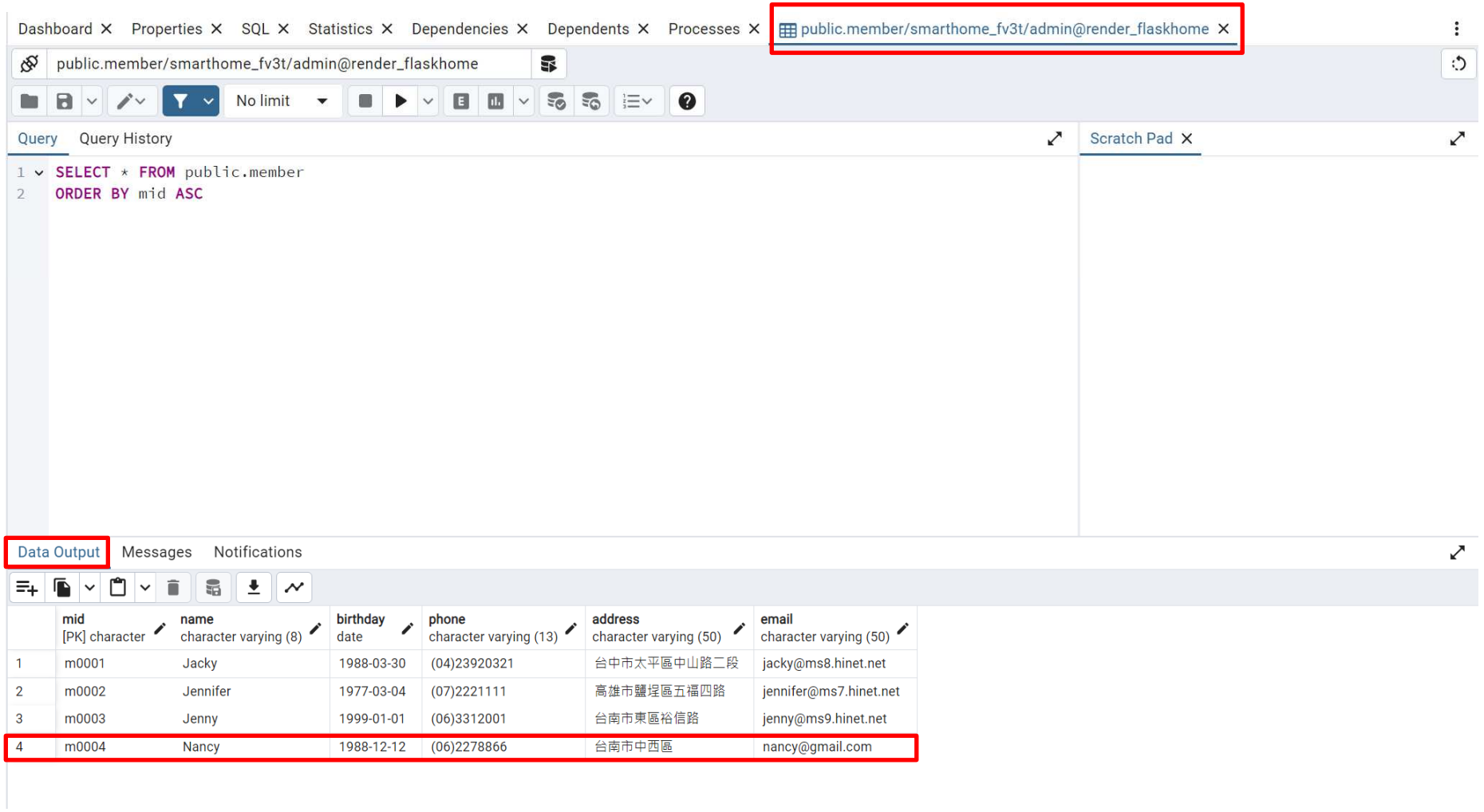
- https://flaskhome.onrender.com/member/signin



檢視 /member/signup 網頁

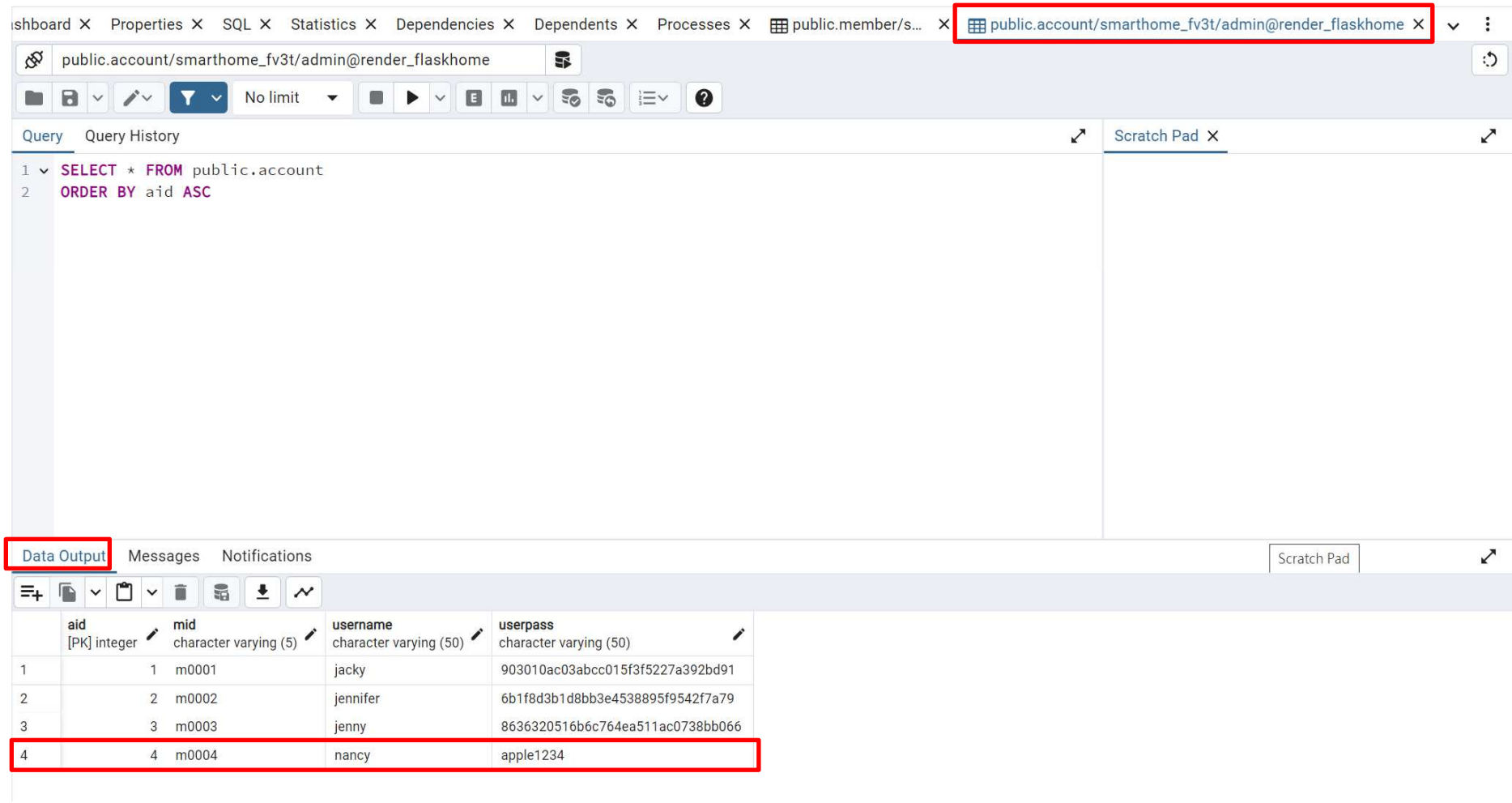
- https://flaskhome.onrender.com/member/signup

查詢 member 資料表



	mid [PK] character	name character varying (8)	birthday date	phone character varying (13)	address character varying (50)	email character varying (50)
1	m0001	Jacky	1988-03-30	(04)23920321	台中市太平區中山路二段	jacky@ms8.hinet.net
2	m0002	Jennifer	1977-03-04	(07)2221111	高雄市鹽埕區五福四路	jennifer@ms7.hinet.net
3	m0003	Jenny	1999-01-01	(06)3312001	台南市東區裕信路	jenny@ms9.hinet.net
4	m0004	Nancy	1988-12-12	(06)2278866	台南市中西區	nancy@gmail.com

查詢 account 資料表



	aid [PK] integer	mid character varying (5)	username character varying (50)	userpass character varying (50)
1	1	m0001	jacky	903010ac03abcc015f3f5227a392bd91
2	2	m0002	jennifer	6b1f8d3b1d8bb3e4538895f9542f7a79
3	3	m0003	jenny	8636320516b6c764ea511ac0738bb066
4	4	m0004	nancy	apple1234

參考文獻

- Python Web Flask — 如何透過Form取得資料
  - https://medium.com/seaniap/python-web-flask-%E5%A6%82%E4%BD%95%E9%80%8F%E9%81%8Eform%E5%8F%96%E5%BE%97%E8%B3%87%E6%96%99-7a63ebf9ff1f
- Python 計算 md5 hash 雜湊值
  - https://shengyu7697.github.io/python-md5/
- Flask實作_ext_03_Flask-WTF_表單建立
  - https://hackmd.io/@shaoeChen/ByofdR1XG?type=view

參考文獻 (Cont.)

- Python Web Flask — 用WTF Form製作表單
  - https://medium.com/seaniap/python-web-flask-%E7%94%A8wtf-form%E8%A3%BD%E4%BD%9C%E8%A1%A8%E5%96%AE-1f4af213ea88
- Flask-WTF 好用的表單渲染
  - https://medium.com/@jacky.goman/flask-wtf-%E6%88%91%E6%B2%92%E7%BD%B5%E4%BA%BA%E5%A5%BD%E7%94%A8%E7%9A%84%E8%A1%A8%E5%96%AE%E6%B8%B2%E6%9F%93-dc9e38e10265

參考文獻 (Cont.)

- Day 18 實作表單 (1)
  - https://ithelp.ithome.com.tw/articles/10266277
- Day 19 實作表單 (2)
  - https://ithelp.ithome.com.tw/articles/10267439
- Day 20 實作表單 (3)
  - https://ithelp.ithome.com.tw/articles/10268388

參考文獻 (Cont.)

- How To Use a PostgreSQL Database in a Flask Application
  - https://www.digitalocean.com/community/tutorials/how-to-use-a-postgresql-database-in-a-flask-application
- PostgreSQL Python – Querying Data
  - https://www.geeksforgeeks.org/postgresql-python-querying-data/

參考文獻 (Cont.)

- Use psycopg2 to return dictionary like values (key-value pairs)
  - https://varun-verma.medium.com/use-psycopg2-to-return-dictionary-like-values-key-value-pairs-4d3047d8de1b
- How to Use FieldList in Flask-WTF
  - https://prettyprinted.com/tutorials/how-to-use-fieldlist-in-flask-wtf/