資料庫

關聯式資料庫

- 關聯式資料庫（Relational Database）是基於關聯模型實作的資料庫
- 常見的關聯性資料庫有
  - Microsoft SQL Server
    - 由Microsoft開發
  - MySQL
    - 原來由瑞典的MySQL AB公司開發，後來被Oracle收購
  - Oracle Database
    - 由Oracle開發
    - 使用費用較高，功能全面
  - SQLite
    - SQLite是一個零配置的數據庫，這意味著與其他數據庫不一樣，您不需要在系統中安裝程式

關聯資料庫語言：SQL及T-SQL

- SQL(Structured Query Language)結構化查詢語言
- 1986年成為美國國家標準學會（ANSI）的一項標準語言
- 用於操作關聯式資料庫，透過SQL可以修改資料庫的內容
- 基本上的功能就是：SELECT、INSERT、UPDATE、DELETE...等
- Microsoft 將SQL擴增為T-SQL(Transact-SQL)
- Transact-SQL又稱為T-SQL

SQL的四種運算子

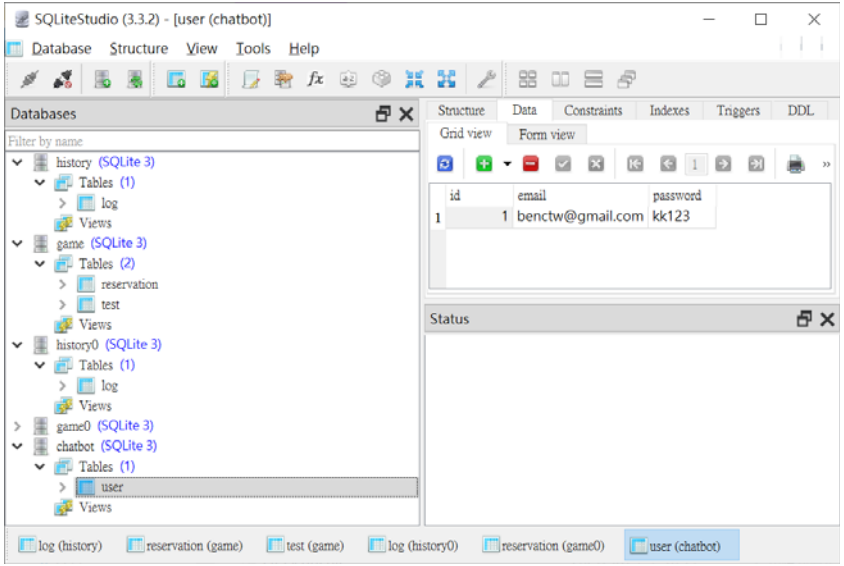
種類	運算元
算術運算子	+，-，*，/，%
邏輯運算子	AND，OR，NOT，LIKE，BETWEEN、IN
指派運算子	=
比較運算子	=，>，<，<>，>=，<=

SQLite開發工具簡介

- 使用「SQLiteStudio」做SQLite資料庫的管理
- https://sqlitestudio.pl/ 可下載「SQLiteStudio」綠色軟體
- SQLite資料庫以單一檔案形式存在，可以使用SQLiteStudio軟體建立一個空SQLite資料庫，隨之進行資料表規劃
- 一個資料庫內，可建置多個資料表，如：user(會員資料)、article(網站文章)、product(商品資訊)等

下載並使用SQLiteStudio

- 到官網下載SQLiteStudio：https://sqlitestudio.pl/
- 建立一個資料庫，內含一個資料表，名稱為user，並建立四個欄位，分別為id、name、email、password
- 嘗試在資料表中新增一些資料



使用Python讀寫SQLite資料庫 - 1

- Python 3 已經內建讀寫SQLite資料庫的模組
- 官方文件： https://docs.python.org/3/library/sqlite3.html
- 寫入資料到SQLite：

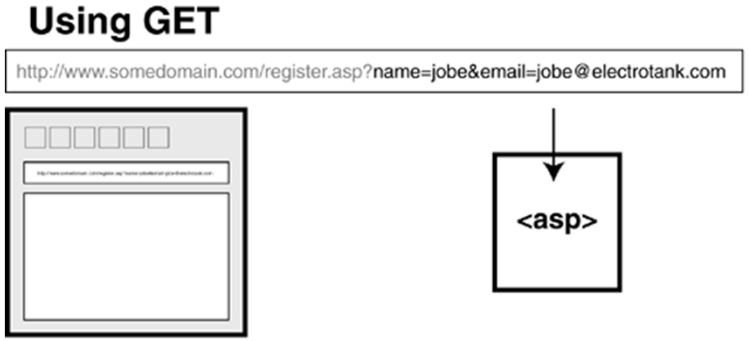
```
import sqlite3
con = sqlite3.connect('mywebsite.db')
cur = con.cursor()
# Insert a row of data
cur.execute("INSERT INTO user (`name`, `email`, `password`) VALUES
('John','john@gmail.com','333333')")
# Save (commit) the changes
con.commit()
# We can also close the connection if we are done with it.
# Just be sure any changes have been committed or they will be lost.
con.close()
```

使用Python讀寫SQLite資料庫 - 2

- 從SQLite資料庫讀出資料
-
```
import sqlite3
con = sqlite3.connect('mywebsite.db')
cur = con.cursor()
querydata = cur.execute('SELECT * FROM user ORDER BY id')
for row in querydata:
    print(row)
con.close
```

HTTP GET vs. POST

- GET
  - 不安全，傳遞的資料在網址列中
  - 只能發送有限量的資料
- POST
  - 較安全
  - 可發送大量數據



https://codebridgeplus.com/get-vs-post/

DEMO

- 安裝postman測試軟體，網址： https://www.postman.com/
- 使用此網站來測試HTTP GET與POST：https://httpbin.org
- 使用瀏覽器將瀏覽請求發送到https://httpbin.org/get
- 使用postman軟體將請求發送到https://httpbin.org/post

HTTP Methods in Flask

```
from flask import request

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return do_the_login()
    else:
        return show_the_login_form()
```

Request 用於取得變數

- 可以使用request取得來自GET或 POST的資料

```
from flask import request
@app.route('/login', methods=['POST', 'GET'])
def login():
    error = None
    if request.method == 'POST':
        if valid_login(request.form['username'],
                       request.form['password']):
            return log_the_user_in(request.form['username'])
        else:
            error = 'Invalid username/password'
    # the code below is executed if the request method
    # was GET or the credentials were invalid
    return render_template('login.html', error=error)
```

Request物件

- 取得POST的變數：
- request.form.get('username', '')

- 取得GET的變數：
- request.args.get('username', '')

下面的URL，即是由許多不同部分所構成：

https://docs.python.org/3.5/search.html?q=parse&check_keywords=yes&area=default

DEMO

- 開啟範例資料夾：Module_04/Example_01，啟動app.py
- 分別開啟以下兩個網址，觀察網頁呈現有何不同：
  - http://127.0.0.1:5000/register
  - http://127.0.0.1:5000/register?username=Benjamin&email=benctw@gmail.com
- 觀察app.py的register路由：
  -
```
@app.route('/register', methods=['GET','POST'])
def register():
    if request.method=='POST':
        do_the_register()
    else:
        # username = request.args.get('username', '') if request.args.get('username', '') else ''
        username = request.args.get('username', '')
        email = request.args.get('email', '')
        return render_template('register.html', username=username, email=email)
```

Cookies

- 如果要讀取cookie，可以使用cookies 屬性。如果寫入cookie，可以使用make_response 模組中的set_cookie方法。

- 寫入cookie

```
from flask import make_response

@app.route('/')
def index():
    resp = make_response(render_template(...))
    resp.set_cookie('username', 'the username')
    return resp
```

- 讀取cookie

```
from flask import request

@app.route('/')
def index():
    username = request.cookies.get('username')
    # use cookies.get(key) instead of
    # cookies[key] to not get a
    # KeyError if the cookie is missing.
```

重定向和錯誤

- 要將用戶重定向到另一個端點，請使用redirect() 函數；要使用錯誤代碼提前中止請求，請使用以下 abort()功能：

```
from flask import abort, redirect, url_for

@app.route('/')
def index():
    return redirect(url_for('login'))

@app.route('/login')
def login():
    abort(401)
    this_is_never_executed()
```

g物件

- Flask可以import一個g物件
- 使用g物件作為全域的臨時儲存的物件，每次重新連線都會重新設置