Web應用程式開發
Web Application Development

JSON & RESTFUL API

授課教師：林器弘

1

Outline

- JSON
- RESTFUL API

2

上課前準備

- 下載上次練習的註冊網頁。
- 上網下載postman API工具。
  - 測試API的神器 POSTMAN
  - https://www.postman.com/downloads/

3

JSON

4

JSON

- 主要函式：dumps()、loads()、dump()、load()

- dumps / loads：在記憶體內轉換
  - dumps()：將Python資料轉換為JSON格式
  - loads()：將JSON格式資料轉換為Python資料。
- dump / load：與檔案操作結合
  - dump()：將Python資料轉換為JSON檔案，資料將轉換為JSON格式。
  - load()：從JSON檔案中讀取資料，資料將轉換為Python格式。

5

JSON 資料轉為 Python 內部的資料格式

原始 JSON 類型	轉為 Python 的類型
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

6

Writing JSON to a file (List)

```
import json
#準備一個list的資料
person_list = [{"姓名": "林小弘", "性別": "男", "學號": "B1231330", "
專長": ["程式語言", "人工智慧"]}]
with open('personc.json', 'w', encoding="utf-8") as json_file:
  json.dump(person_list, json_file, ensure_ascii=False, indent=2)

print(type(person_list))
```

7

Open JSON file (List)

```
import json

with open("personc.json", "r", encoding="utf-8") as f:
    person = json.load(f)   # 將 JSON 資料讀取出來

print(type(person))
print(person)
```

8

Try and Except

- try 和 except 是 Python 的錯誤處理機制（Exception Handling）
  - 作用為： 當程式發生錯誤時，不讓整個程式崩潰，改由你自己決定怎麼處理。
  - except 會捕捉錯誤並顯示訊息，不會讓程式整個中止。

```
try:
   # 嘗試執行這裡的程式
   ...
except:
   # 如果上面發生錯誤，就執行這裡
   ...
```

9

Try and Except

- try 和 except 是 Python 的錯誤處理機制（Exception Handling）
  - 作用為： 當程式發生錯誤時，不讓整個程式崩潰，改由你自己決定怎麼處理。
  - except 會捕捉錯誤並顯示訊息，不會讓程式整個中止。
  - else：若沒發生錯誤就執行。（可省略）
  - finally：無論是否出錯都會執行（常用來關檔案或關連線，（最後清理動作））。

```
try:
   f = open("person.json", "r",
encoding="utf-8")
   content = f.read()
except FileNotFoundError as e:
   print("找不到檔案！")
else:
   print("檔案內容：", content)
finally:
   print("執行結束。")
```

10

定義函式（define function）

- 在 Python 裡，def 是 定義函式（define function），可以讓你把一段重複會用到的程式包起來、取個名字、以後直接呼叫。

```
沒有回傳值（只執行動作）

def hi(name):
    print(f"你好，{name}！")


hi("小庚")
```

11

練習一

- 請使用第六周的網頁為基礎，修改填寫結果除了呈現在result.html網頁上，另外寫入一個students.json的檔案。
- 每次輸入一筆資料，就會不斷往下增加註冊資料。
- 請至少輸入三筆資料(當中三筆資料學號為:123, 345, 567)，其他資料隨意輸入。

12

RESTFUL API

13

RESTFUL API

- 應用程式介面（英語：application programming interface，縮寫作API）
- 我們直接來舉個實際例子吧
- *我去麥當當買吃的，我只需要知道我要點什麼吃什麼並且告訴櫃檯人員，我不需要了解漢堡薯條怎麼做的，我只負責吃!*

14

RESTFUL API

- 想要吃勁辣雞腿套餐，所以我櫃台店員(API)點餐，店員就會將你的餐點告知給廚房(後台)，廚房製作完成餐點送出來，並且店員將餐點送到你手上。

15

RESTFUL API

- 使用者將資料傳遞給 API，API 會進行資料邏輯處理，處理完後產生的Respond 再透過 API 回傳給使用者。所以 API 就是我們常說的接口。
- 還有幾個最常看到的例子就是像Facebook、Google、Line、淘寶、微博登入、Line支付功能、Youtube數據等等一堆都是API。

- REST全名Representational State Transfer，中文是表現層狀態轉換，是一種「設計網路服務（API）架構風格，他並不是一種標準。
- REST強調用 HTTP 的標準方法（GET、POST、PUT、DELETE…）來操作「資源（Resource）」。
- 當你的 API 遵守 REST 原則設計時，就稱為 RESTful API。

16

RESTFUL API

- RESTful API是推薦(建議)你照著他們的設計方式。
- 所以表示並不是強制性，如果沒照著他們規定的方式做也不會怎麼樣。(當然不照做就不是RESTful API)

這是早期設計API的樣子	這是Restful API的樣子
/api/get_file/ (得到檔案)	/api/files/ (GET -> 得到檔案)
/api/post_file/ (新增檔案)	/api/files/ (POST -> 新增檔案)
/api/update_file/ (更新檔案)	/api/files/ (PUT -> 更新檔案)
/api/delete_file/ (刪除檔案)	/api/files/ (DELETE -> 刪除檔案)

應該明顯看出差異性了吧~

17

RESTFUL API

- RESTful API有以下設計要點:
  - 統一介面
  - 對資源的 增/刪/改/查，分別用HTTP的Method POST/DELETE/PUT/GET 對應
  - 分層系統（Layered System）
  - 客戶端—伺服器分離（Client-Server Separation）
  - 無狀態
  - Cache
- **統一介面**
  - 也就是說他只有**唯一**的URI，可以看到上述了例子/api/files/，所有的動作(新、更、刪等等)僅透過唯一的/api/files/進行。然而就不會像早期API那般有多URI卻都在對同一個地方進行動作。
  - 另外，通常在命名URI時都會使用複數名詞，像範例中**files**或**Users**等等，不會出現動詞型態，如**get_files**，因為操作上會使用HTTP中的動詞(下方會敘述)。這樣不僅省時省力，設計人員也能方便閱讀，還能提對Server端提高效率。

18

RESTFUL API

- 對資源的 增/刪/改/查，分別用HTTP的Method POST/DELETE/PUT/GET 對應
- REST風格所有的東西皆都在HTTP協議之下完成，所有動作(動詞)都會對應到HTTP當中POST/DELETE/PUT/GET。這樣一來在設計時，就能避免多次上述所說的多URI卻都在對同一個地方進行動作等等問題。
- 常見的method：
  - GET : 取得資料或是狀態
  - POST : 新增一項資料
  - PUT/ PATCH : 更新資料。
  - DELETE : 指定資料刪除。

19

RESTFUL API

- 由下表可以看到，若以 RESTful API 風格開發的話，網址幾乎沒有什麼改變：

功能	方法	網址
新增使用者資料	POST	/user/<username>
取得特定使用者資料	GET	/user/<username>
取得全部使用者資料	GET	/users
更新使用者資料	PUT / PATCH	/user/<username>
刪除使用者資料	DELETE	/user/<username>

20

RESTFUL API

- 通過URI操作資源，改變資源狀態
- 簡單說就是對/api/files/進行post(/put/delete等)資料的動作。

所以照著REST的風格特點所設計出來的API就是RESTful囉~~

21

RESTFUL API

- 網址要長怎樣
  - 首先要決定這個 API 放哪裡。通常如果我們要架設新的服務，例如 cgmh.ai.tw 的話，我們的 API 網址可能會是：

    https://cgmh.ai.tw/api

- **Endpoints**
  - 決定了 API 的網址後，要先規劃一下 API 大概會分幾個部分。假設我們要做一個簡單的部落格系統，可能會分成 **使用者** 部分和 **文章** 的部分。
  - Endpoints 可以說是 RESTful 的「名詞」。

22

RESTFUL API

- 實作簡易版RESTful API
  - Flask-RESTful 是一個輕量級的Flask擴充套件，讓我們可以快速建立 REST API。這個套件，可以與ORM資料庫一起使用。如果本身對於 Flask熟悉的話，要上手這個套件並不難。
  - 接下來我們要透過Flask-RESTful實際製作一個簡易版的RESTful API。
- 建立FLASK應用程式
  - 在進行這個簡易版RESTful API前，需要先建立一個FLASK應用程式。

```
from flask import Flask
app = Flask(__name__)
if __name__ == '__main__':
   app.run()
```

23

RESTFUL API

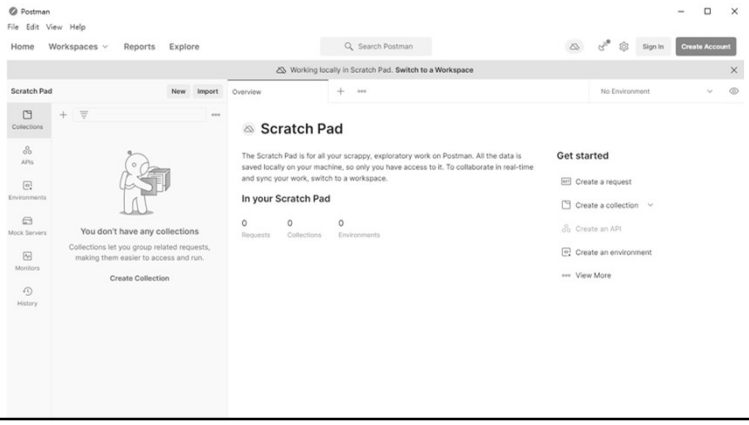
- 建立一個Restful API 基本框架，可以從下面的FLASK應用程式開始。下面的應用程式裡，已經匯入了flask_restful套件，並且將app實體化並包入API了。

```
from flask import Flask

# 建立 Flask 應用
app = Flask(__name__)

# 定義路由（GET 方法）
@app.route("/hello", methods=["GET"])

def hello_world():
   return {"message": "Hello, World!"}

# 啟動伺服器
if __name__ == "__main__":
   app.run()
```

24

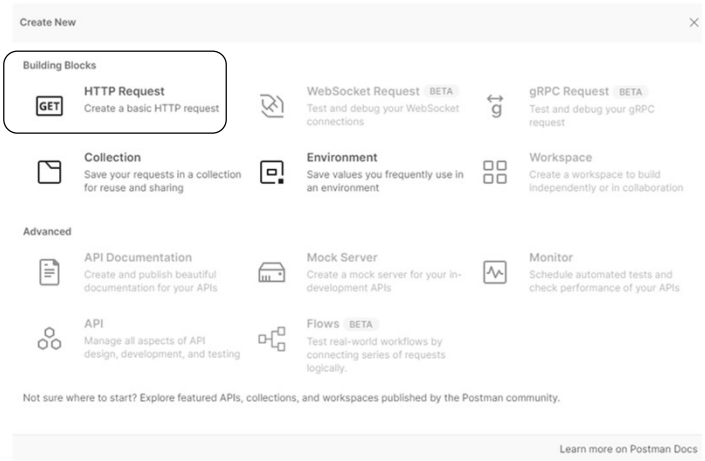
RESTFUL API

- 測試API的神器 POSTMAN
- https://www.postman.com/downloads/



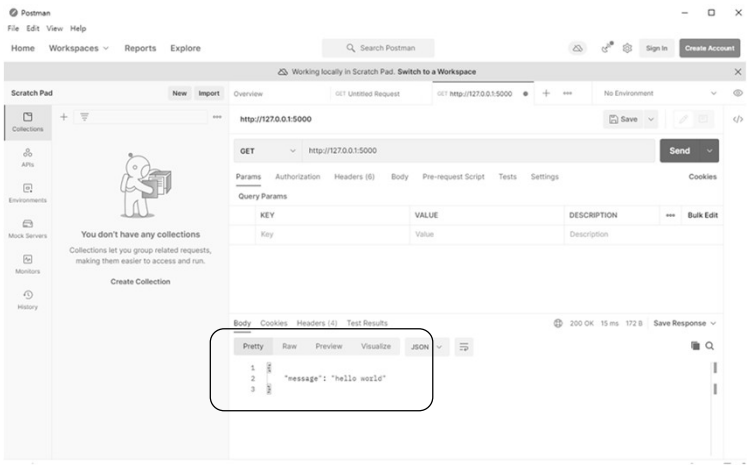
25

RESTFUL API



26

RESTFUL API



27

curl

- Curl http://127.0.0.1:5000/hello
- Powershell

```
StatusCode        : 200
StatusDescription : OK
Content           : {"message":"Hello, World!"}
```

- CMD

```
{"message":"Hello, World!"}
```

28

jsonify

- jsonify 是 Flask 提供的一個非常重要的工具，用來「回傳 JSON 格式的資料給前端或 API 使用者」。
- 它是建立 RESTful API 時最常用的回傳方式之一。
- 把 Python 的資料型別（如 dict、list）轉換成 **JSON 格式**，同時自動設定 HTTP 標頭（Content-Type: application/json）

```
在 Flask 裡：
from flask import jsonify
def error():
   return jsonify({"錯誤" : "沒有找到資料"})
```

jsonify() = Flask 幫你包好的「安全 JSON 回傳器」，不用手動轉字串、不用加 header、支援狀態碼。

- 可以搭配回傳狀態碼

```
def error():
   return jsonify({"錯誤": "沒有找到資料"}), 404
```

29

HTTP 狀態碼（Status Code）

狀態碼	說明	範例
200 OK	成功	資料查詢成功
201 Created	成功建立	新增成功
204 No Content	成功但無內容	刪除成功
400 Bad Request	用戶請求錯誤	參數錯誤
401 Unauthorized	未授權	Token 錯誤
404 Not Found	找不到資源	ID 不存在
500 Internal Server Error	伺服器錯誤	程式出錯

30

Jsonify

為何API上面常用jsonify() 而不是使用 json.dumps()

項目	jsonify()	json.dumps()
功能	產生 Flask 的 Response 物件	只轉成 JSON 字串
自動加 Header	是 (application/json)	否
支援 HTTP 狀態碼	可一起設定	需手動處理
常見用途	RESTful API 回傳資料	檔案寫入或資料序列化

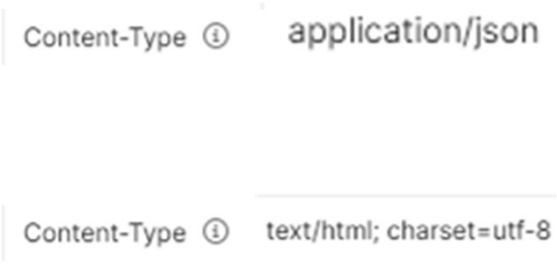
常見用途

場景	說明
RESTful API 回傳結果	統一格式，給前端或第三方系統用
回傳錯誤訊息	結合 HTTP 狀態碼
回傳查詢結果	JSON 是通用格式，跨平台易解析

31

Jsonify VS. dumps

```
from flask import Flask, jsonify
import json

app = Flask(__name__)

@app.route("/jsonify")
def use_jsonify():
    return jsonify({"status": "jsonify ok"}), 250

@app.route("/dumps")
def use_dumps():
    # 手動轉成字串
    return json.dumps({"status": "dumps ok"})

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



32

使用api 查詢學生資料

- 加入個資料查詢的方法取得所有學生資料

```
from flask import Flask, jsonify
import json, os

app = Flask(__name__)

# 指定資料檔案
JSON_FILE = "students.json"
# 讀取 JSON 資料的函式
def load_data():
    if not os.path.exists(JSON_FILE):
        # 若檔案不存在，回傳空 list
        return []
    try:
        with open(JSON_FILE, "r", encoding="utf-8") as f:
            return json.load(f)
    except json.JSONDecodeError:
        return [] # 檔案內容壞掉時防呆

# 讀取全部學生資料
@app.route("/students", methods=["GET"])
def get_all_students():
    data = load_data()
    return jsonify(data)


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

33

Result

34

RESTFUL API

- 加入個資料查詢的方法取得特定學生資料
- <> 是 Flask 動態路由（Dynamic Route） 的語法
- <> 是 Flask 當作是路徑參數（path parameter），用來把 URL 裡的一部分，當成函式的輸入參數。

```
# 讀取特定學生（用 ID 查）
@app.route("/students/<sid>", methods=["GET"])
def get_student(sid):
    data = load_data()
    # 用 id 找對應學生
    student = next((s for s in data  if s.get("id") == sid), None)
    print(student)
    if student:
        return jsonify(student)
    else:
        return jsonify({"錯誤": "沒有找到學生資料"}), 404
```

35

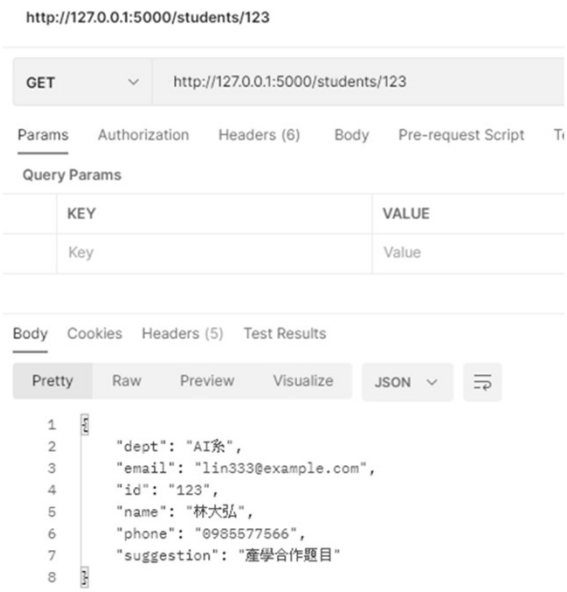
RESTFUL API

- <> 裡除了寫名稱，也可以加上型別轉換器（Converter），例如：

型別轉換器	說明	範例 URL
<string:var>	預設型別（字串）	/students/alice
<int:var>	只接受整數	/students/123
<float:var>	接受浮點數	/grades/87.5
<path:var>	可包含 /	/files/images/pic1.png

36

Result



37

使用API新增學生資料

```
@app.route("/students", methods=["POST"])
def add_student():
    # 使用API新增學生資料
    body = request.get_json(silent=True) or {}
    print(body)
    data = load_data()
    print(data)
    # 必填欄位
    required = ["id", "name", "email", "dept", "phone", "suggestion"]


    missing = [r for r in required if not str(body.get(r, "")).strip()]
    if missing:
        return jsonify({"錯誤": f"Missing fields: {', '.join(missing)}"}), 400
```

```
    # 檢查 ID 是否重複
    if any(s["id"] == body["id"] for s in data):
        return jsonify({"錯誤": f"Student ID ' {body['id']} ' 資料已經存在"}), 400


    # 組合新資料
    new_student = {
        "id": str(body["id"]).strip(),
        "name": str(body["name"]).strip(),
        "email": str(body["email"]).strip(),
        "dept": str(body["dept"]).strip(),
        "phone": str(body["phone"]).strip(),
        "suggestion": str(body["suggestion"]).strip()
    }


    data.append(new_student)
    save_data(data)
    return jsonify(new_student), 201
```

38

Result

{
"id": "B000303",
"name" : "林大弘",
"email": "lin333@example.com",
"dept": "AI系",
"phone": "0911222333",
"suggestion" : "網頁應用程式好簡單"
}

39

練習二

- 使用API新增兩筆學生資料至students.json的檔案中，一筆為您的資料，另外一筆學號為您的學號少一號，內容隨便輸入。

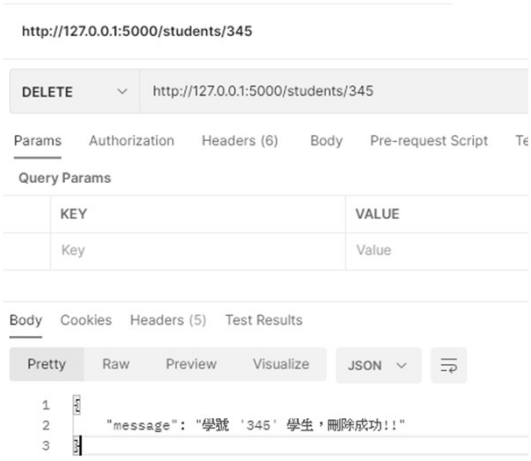
40

使用API刪除學生資料

```
@app.route("/students/<sid>", methods=["DELETE"])
def delete_student(sid):
    data = load_data()

    # 找出要刪除的學生
    student = next((s for s in data if s.get("id") == sid), None)
    if not student:
        return jsonify({"錯誤": f"學號 '{sid}' 查無該筆資料"}), 404


    # 移除該筆資料
    data.remove(student)
    save_data(data)


    return jsonify({
        "message": f"學號 '{sid}' 學生，刪除成功!!"
    }), 200
```

41

Result



42

使用API修改學生資料

- PUT 和 PATCH 都是「更新（Update）資源」的 HTTP 方法，但它們的語意不同。

- 核心差異:

比較項目	PUT	PATCH
更新方式	用新資料取代舊資料（全部欄位）	只修改指定欄位（不動其他欄位）
必須提供所有欄位	是	否
用途	資料結構覆蓋更新	局部更新
常見錯誤	欄位會被覆蓋為空值	只改指定欄位
適用場景	使用者資料重填	修改單一欄位（如電話或建議）
特性	要提供所有欄位	可以只提供部分欄位

43

使用API修改學生資料

```
@app.route("/students/<sid>", methods=["PUT", "PATCH"])
def update_student(sid):
    data = load_data()

    student = next((s for s in data if s["id"] == sid), None)
    if not student:
        return jsonify({"錯誤": "沒有找到您指定的學生"}), 404

    body = request.get_json(silent=True) or {}

    if request.method == "PUT":
        #整筆更新
        required = ["id","name", "email", "dept", "phone", "suggestion"]
        missing = [r for r in required if r not in body]
        if missing:
            return jsonify({"錯誤": f"欄位錯誤: {', '.join(missing)}"}), 400
        student.update(body)

    elif request.method == "PATCH":
        #局部更新
        for key in ["name", "email", "dept", "phone", "suggestion"]:
            if key in body:
                student[key] = body[key]

    save_data(data)
    return jsonify(student), 200
```

44

使用API修改學生資料

PUT
修改學號資料567

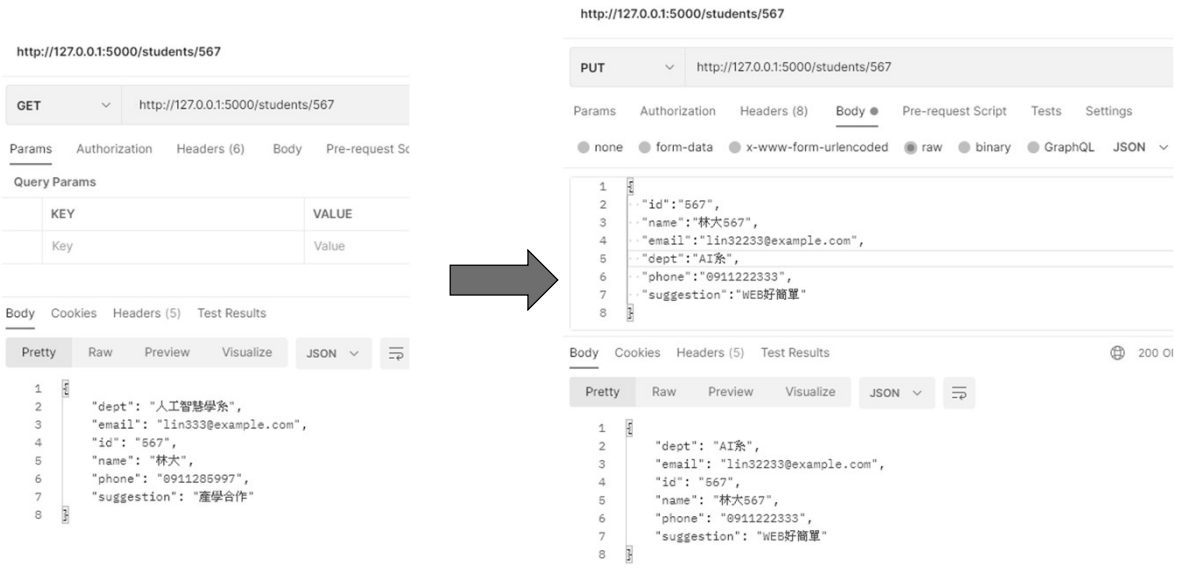
```
{
 "id": "567",
  "name": "林大567",
  "email": "lin32233@example.com",
  "dept": "AI系22",
  "phone": "0911222333",
  "suggestion": "WEB好簡單"
}
```

PATCH
修改學號資料123

```
{
 "name":"林大弘",
  "email":"lin333@example.com",
  "dept":"AI系"
}
```

45

Result



46

Patch 修改資料

47

練習三

- 延續上面練習二，修改students.json的檔案中，您學號的下一位同學基本資料，當中三個欄位，內容如下。

```
{
 "email":"next@google.com",
 "phone":"0921168168",
  "suggestion" :"完成，本學期課程"
}
```

48

Q&A

49