英文聽說讀寫專案

後端使用手冊

**目錄**

[**壹、** **系統架構** 3](#_Toc100579458)

[**一、** **網站架構** 3](#_Toc100579459)

[**二、** **資料庫結構** 3](#_Toc100579460)

[貳、 Django REST Framework教學 11](#_Toc100579461)

[一、 快速入門 11](#_Toc100579462)

[1. 建置專案 11](#_Toc100579463)

[2. 資料建模 16](#_Toc100579464)

[3. 建構序列器 18](#_Toc100579465)

[4. 視圖 20](#_Toc100579466)

[二、 Google Api 串接教學 24](#_Toc100579467)

[三、 存取方式 25](#_Toc100579468)

[四、 英翻中、中翻英(API版本) 27](#_Toc100579469)

[**參、** **使用資源及API定價** 28](#_Toc100579470)

[**一、** **函式庫** 28](#_Toc100579471)

[**二、** **各API收費方式** 29](#_Toc100579472)

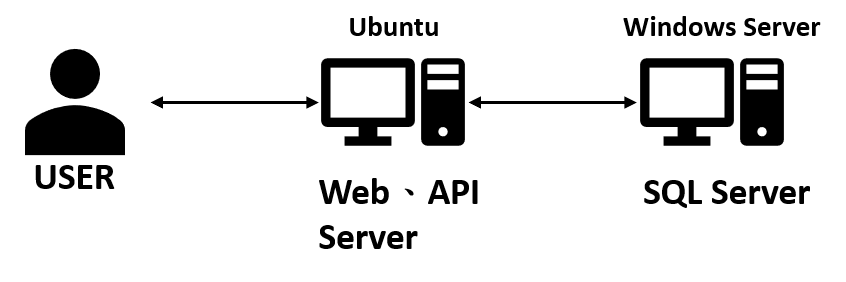
[1. Google translate api 29](#_Toc100579473)

[2. Google dialogflow 29](#_Toc100579474)

[3. IBM 29](#_Toc100579475)

1. **系統架構**
2. **網站架構**

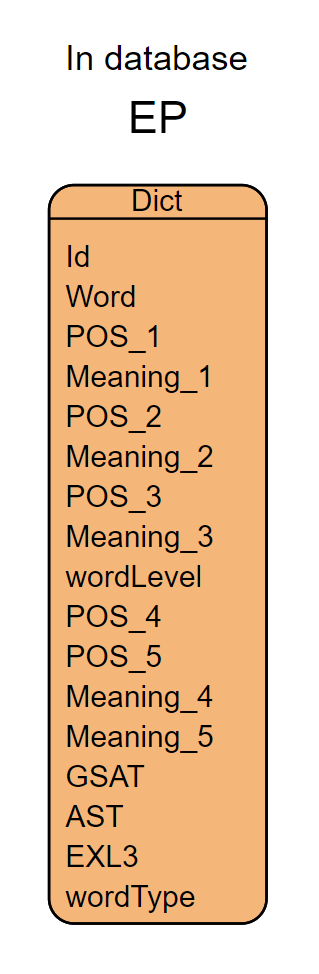
下圖為硬體架構圖

****

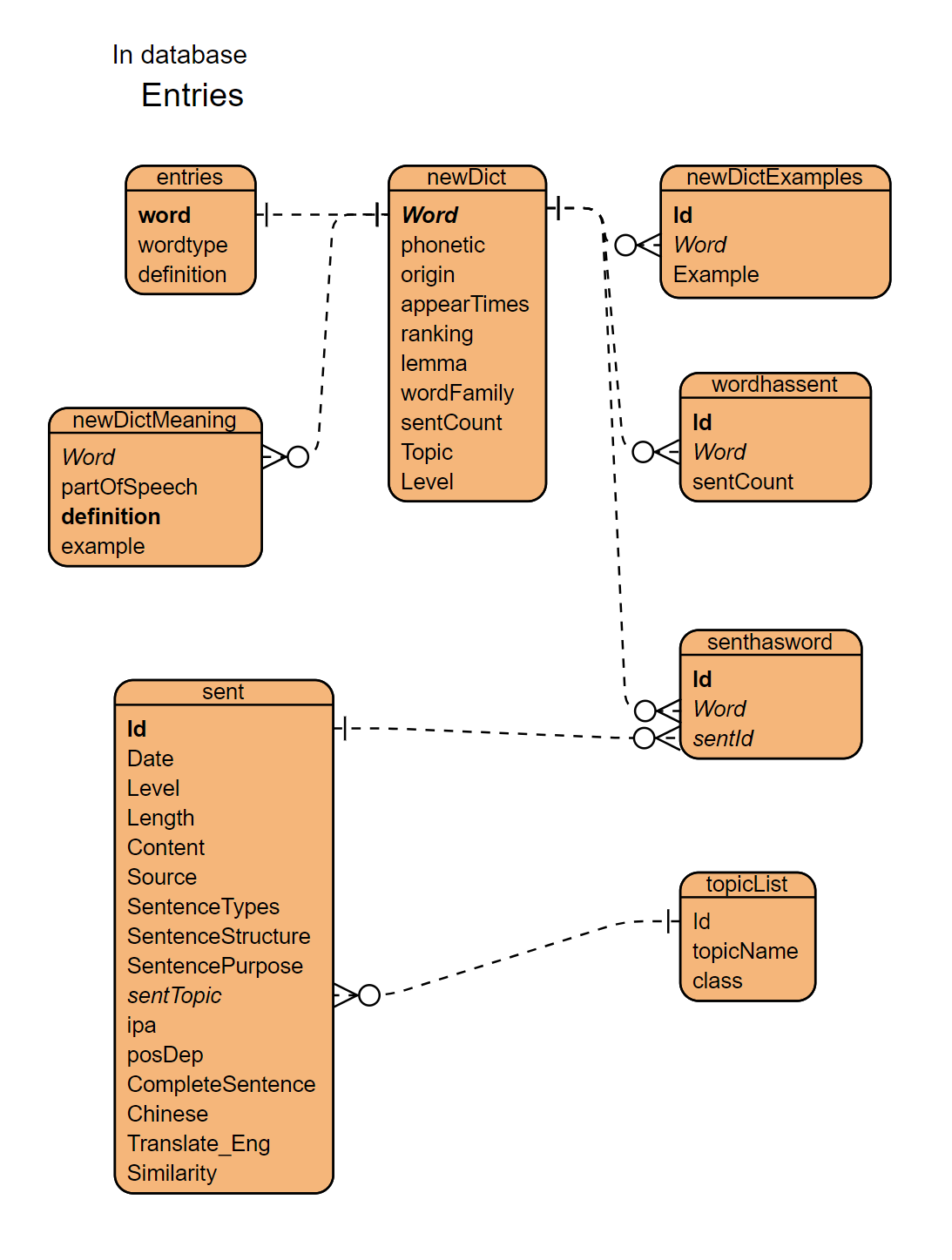
下圖為網頁架構圖

1. **資料庫結構**

在第一個資料庫EP裡有一個表叫Dict，存放有中文定義及Topic分類的單字，主鍵為Id，POS為詞性，Meaning為定義wordLevel為CREF定義之等級，wordType為Topic分類，GSAT、AST、EXL3為各項考試，若有在該考試出現過則該欄位為True。如下表所示。

****

在Entries資料庫中有以下所示之資料表，下圖為Entries DB中表格之ER圖。

****

以下為各表欄位之型態與備註。

**entries資料表**

|  |  |  |
| --- | --- | --- |
| **欄位** | **型態** | **備註** |
| **word** | **Varchar(25)** | **單字** |
| **wordtype** | **Varchar(20)** | **詞性** |
| **definition** | **text** | **定義** |

**newDict資料表**

此資料表無中文定義，為用entries資料表單字去WordNet抓取資料而來。

|  |  |  |
| --- | --- | --- |
| **欄位** | **型態** | **備註** |
| **word** | **Varchar(50)** | **單字** |
| **phonetic** | **text** | **讀音** |
| **origin** | **text** | **起源** |
| **appearTimes** | **BIGINT** | **出現次數，來自google統計** |
| **ranking** | **Varchar(50)** | **評等** |
| **lemma** | **Varchar(50)** | **詞條** |
| **wordFamily** | **Varchar(50)** | **字群** |
| **sentCount** | **INT** | **有多少句子** |
| **Topic** | **INT** | **CREF分類** |
| **Level** | **Varchar(50)** | **CREF等級** |

**newDictExamples資料表**

WordNet蒐集而來**。**

|  |  |  |
| --- | --- | --- |
| **欄位** | **型態** | **備註** |
| **Id** | **int** | **編號，自動產生** |
| **word** | **Varchar(41)** | **單字** |
| **Example** | **text** | **範例句** |

**newDictMeaning資料表**

WordNet蒐集而來。

|  |  |  |
| --- | --- | --- |
| **欄位** | **型態** | **備註** |
| **word** | **Varchar(41)** | **單字** |
| **partOfSpeech** | **Varchar(50)** | **詞性** |
| **definition** | **text** | **定義** |
| **example** | **text** | **範例句或字** |

**sent資料表**

|  |  |  |
| --- | --- | --- |
| **欄位** | **型態** | **備註** |
| **Id** | **Int** | **編號，自動產生** |
| **Date** | **Date** | **匯入日期** |
| **Level** | **varChar(10)** | **句子CREF等級** |
| **Length** | **Int** | **句子長度** |
| **Content** | **Text** | **內容** |
| **Source** | **varChar(30)** | **來源** |
| **SentenceTypes** | **varChar(30)** | **句子型態** |
| **SentenceStructure** | **varChar(30)** | **句子結構** |
| **SentencePurpose** | **varChar(30)** |  |
| **sentTopic** | **Int** | **句子CREF種類** |
| **ipa** | **Text** | **整句的音標** |
| **posDep** | **Int** |  |
| **CompleteSentence** | **Boolean** | **是否為完整句子** |
| **Chinese** | **Text** | **中文翻譯** |
| **Translate\_Eng** | **Text** | **將中文翻譯翻成英文** |
| **Similarity** | **Double** | **翻譯完英文與原句的相似度** |

**SentHasWord資料表**

|  |  |  |
| --- | --- | --- |
| **欄位** | **型態** | **備註** |
| **Id** | **int** | **編號，自動生成** |
| **word** | **Varchar(50)** | **單字** |
| **sentId** | **int** | **對應至Sent的Id** |

**WordHasSent資料表**

|  |  |  |
| --- | --- | --- |
| **欄位** | **型態** | **備註** |
| **Id** | **int** | **編號，自動生成** |
| **word** | **Varchar(50)** | **單字** |
| **wordCount** | **int** | **該單字有多少句子** |

**topicList資料表**

topics皆來自CREF定義

https://www.oxfordlearnersdictionaries.com/us/topic/

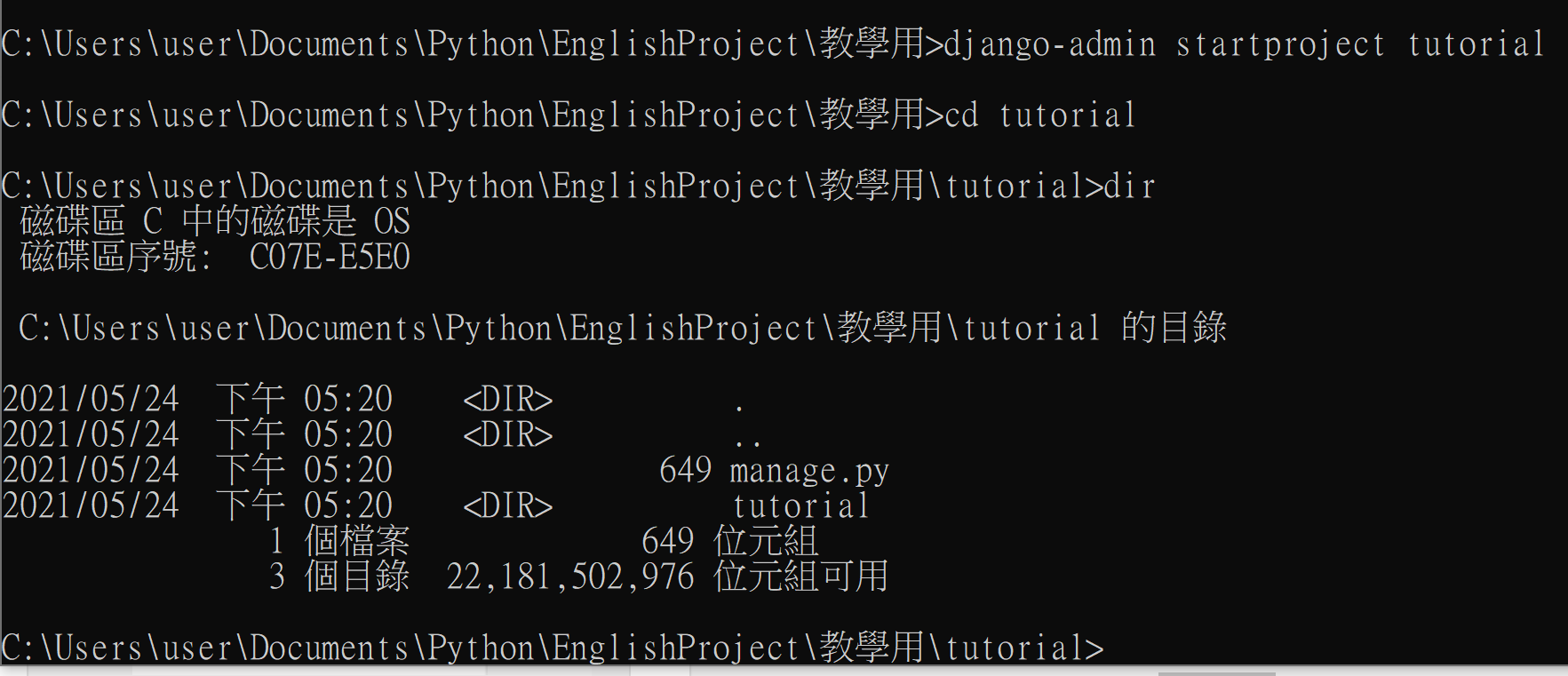
|  |  |  |
| --- | --- | --- |
| **欄位** | **型態** | **備註** |
| **Id** | **int** | **編號，自動生成** |
| **topicName** | **Varchar(50)** | **分類名稱(細項)** |
| **class** | **Varchar(50)** | **類別名稱(大項)**  **Ps.該分類屬於哪項類別** |

# Django REST Framework教學

## 快速入門

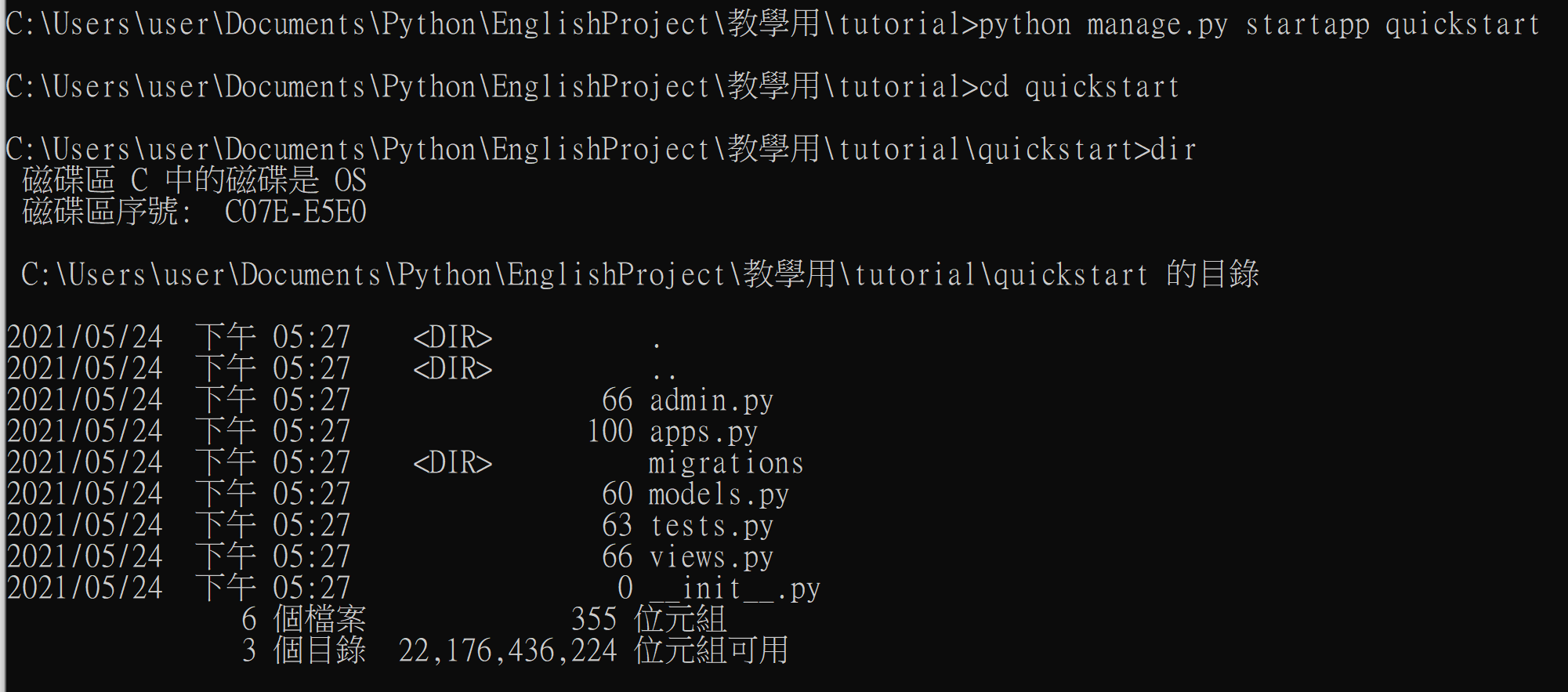
### 建置專案

先切換至想建置專案的目錄下，使用django-admin startproject your\_project\_name 建設出專案，建置好後切換至專案目錄底下，成功的話如下圖所示。



接著我們開啟一個APP，APP可以想像成是一個功能或是資料表或是應用集區，使用python manage.py startapp yourappname 進行建設。

成功的話會如下圖所示。



絕大多數文件的命名和它們的目的有關（比如視圖函數就是views.py，模型就是models.py，測試是tests.py，網站管理設置是admin.py，註冊應用是apps.py）

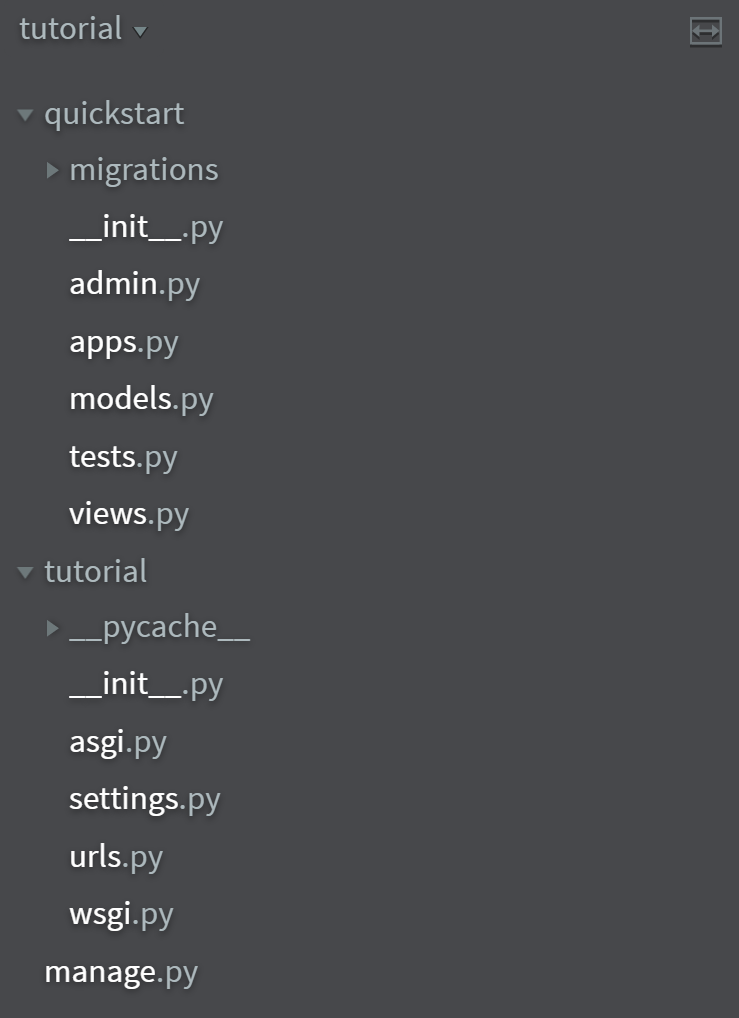
除上面所說的文件外，我們還有：

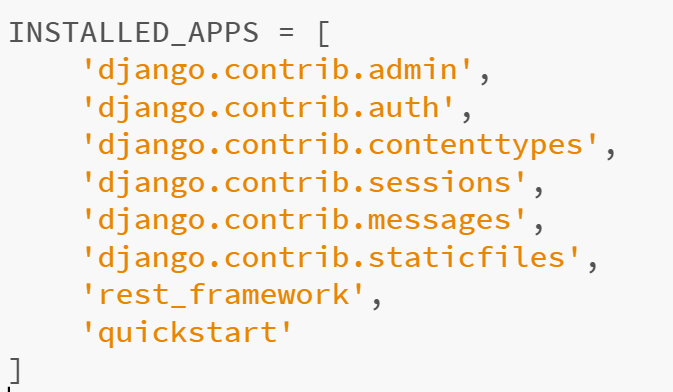
一個migration文件夾，用來存放“migrations”:當你修改你的數據模型時，這個文件會自動升級你的資料庫。

\_\_init\_\_.py :一個空文件，Django/Python會將這個文件作為Python套件包並允許你在項目的其他部分使用它。

建立完成APP後我們要去設定的地方向Django註冊這個APP。

設定的位置在專案名稱資料夾的settings.py



在其中INSTALLED\_APPS裡加入APP的名字，因為我們使用的是REST的框架，所以要在註冊rest\_framework。

順便可以修改Database設定，若是要連接MySQL請參考以下網址。

[[Python] Django連接現有MySQL資料庫](https://chentsungyu.github.io/2020/02/13/Python/%5bPython%5d%20Django%E9%80%A3%E6%8E%A5%E7%8F%BE%E6%9C%89MySQL%E8%B3%87%E6%96%99%E5%BA%AB/)

完成後我們來連結URL映射器(urls.py)

在項目文件夾裡，創建網站時同時生成了URL映射器（urls.py）。

儘管你可以用它來管理所有的URL映射，但是更常用的做法是把URL映射留到它們相關的應用中。



可以看到原本的註解中就有教你怎麼設定連結，我們要實做的事Including another URLconf的部分。

在urlpatterns下方插入新的程式碼，並import include的函式庫。



這個新項目包括一個 path() ，它使用模式 quickstart/ 轉發請求到模塊 quickstart.urls（具有相對 URL : / quickstart /urls.py 的文件）

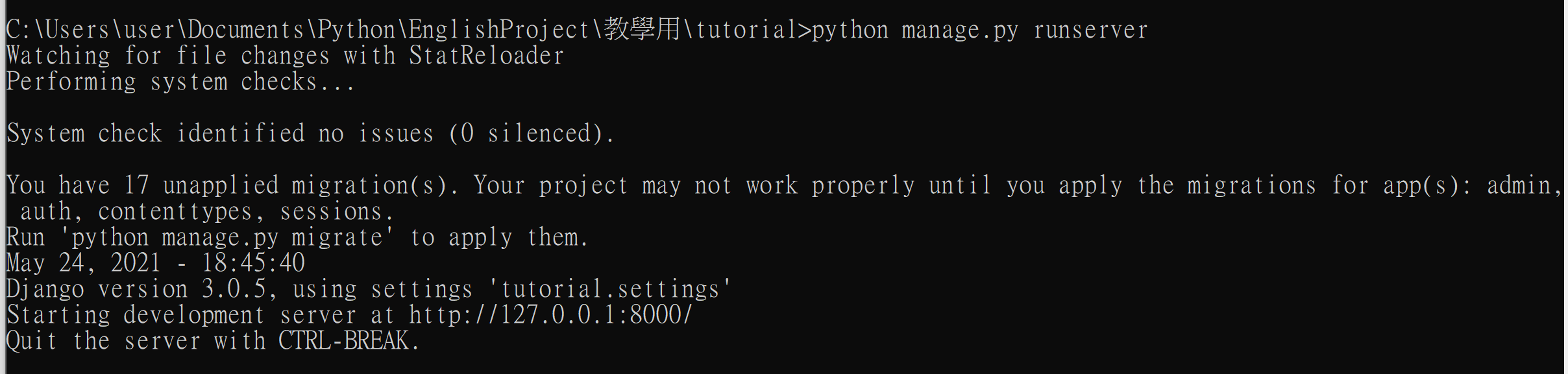
接著在應用程式中建立urls.py的文件，並輸入程式碼。



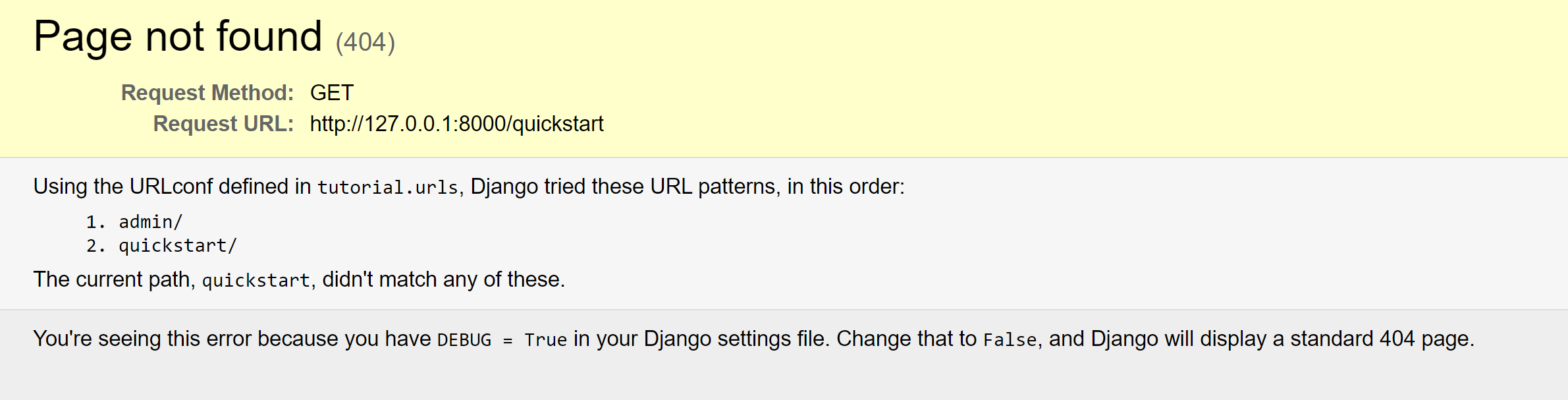
設定完成後，可以啟動整個專案試試看。

使用python manage.py runserver

試試看，成功會如下圖所示。



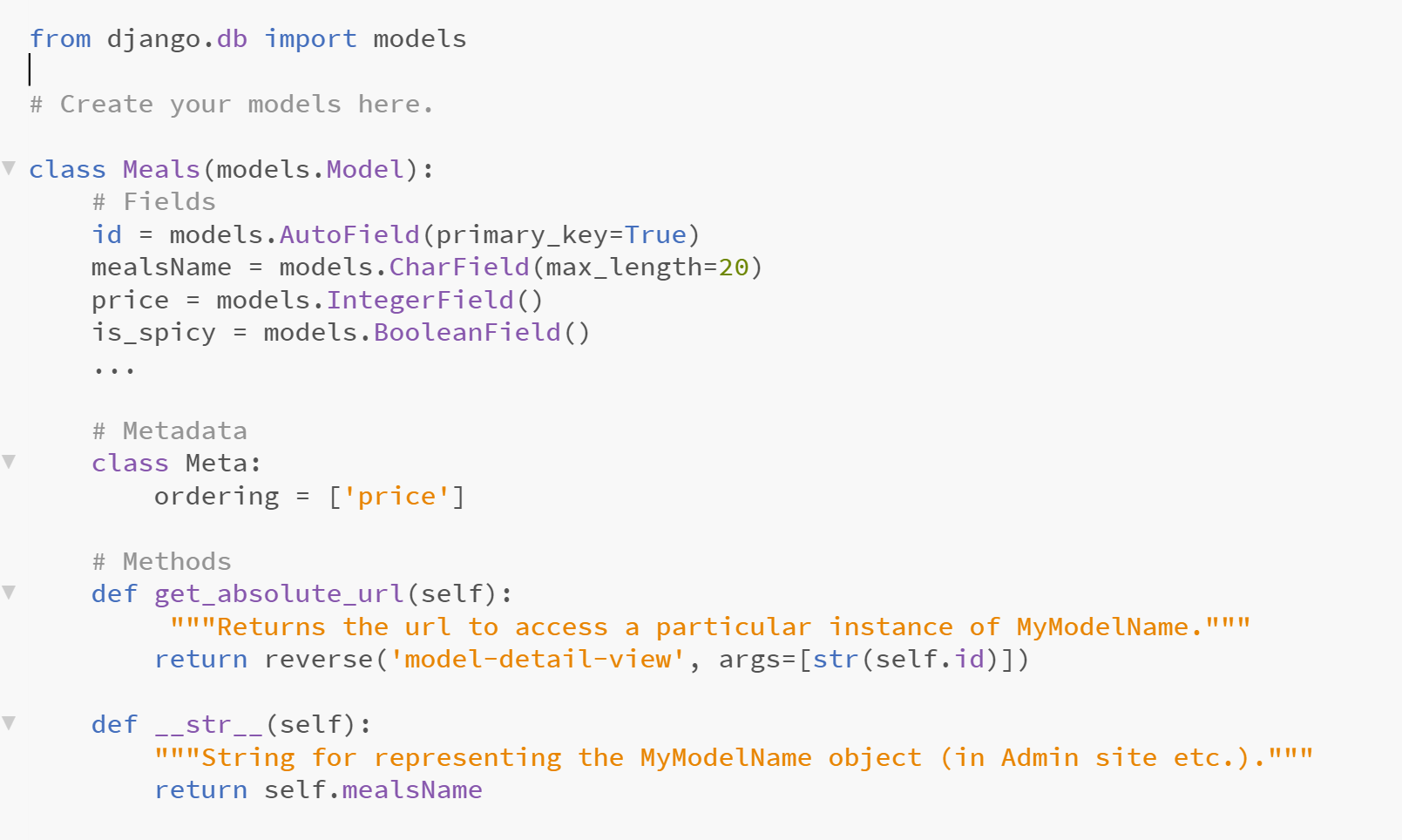
連接上127.0.0.1:8000/quickstart如下圖頁面所示即為正常，因為還沒將URL指定到VIEW



### 資料建模

接著我們要來建置我們的資料模型，以菜單來舉例，我們需要有菜名、價錢、是不是辣的，三種元素來構成一行餐點的資料。

菜名我們需要是字元的格式，價錢為整數的格式，是不是辣的為布林格式，依此我們可以來建置我們的資料模型，打開APP的models.py，輸入以下的程式碼建置出Meals資料表。

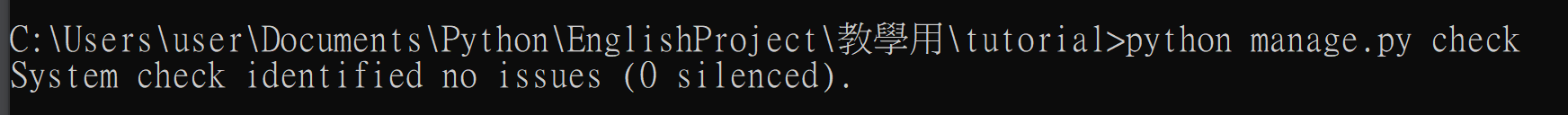


一個模型通常是繼承自 django.db.models.Model的子類， 可以包括屬性，方法和描述性資料(metadata)。

詳細內容細節請至以下連結查詢:

[Django Tutorial Part 3: Using models](https://developer.mozilla.org/zh-TW/docs/Learn/Server-side/Django/Models)

建置好模型後我們先使用cmd指令確認模型是否正確，使用 python manage.py check，如果正確會如下圖所示。



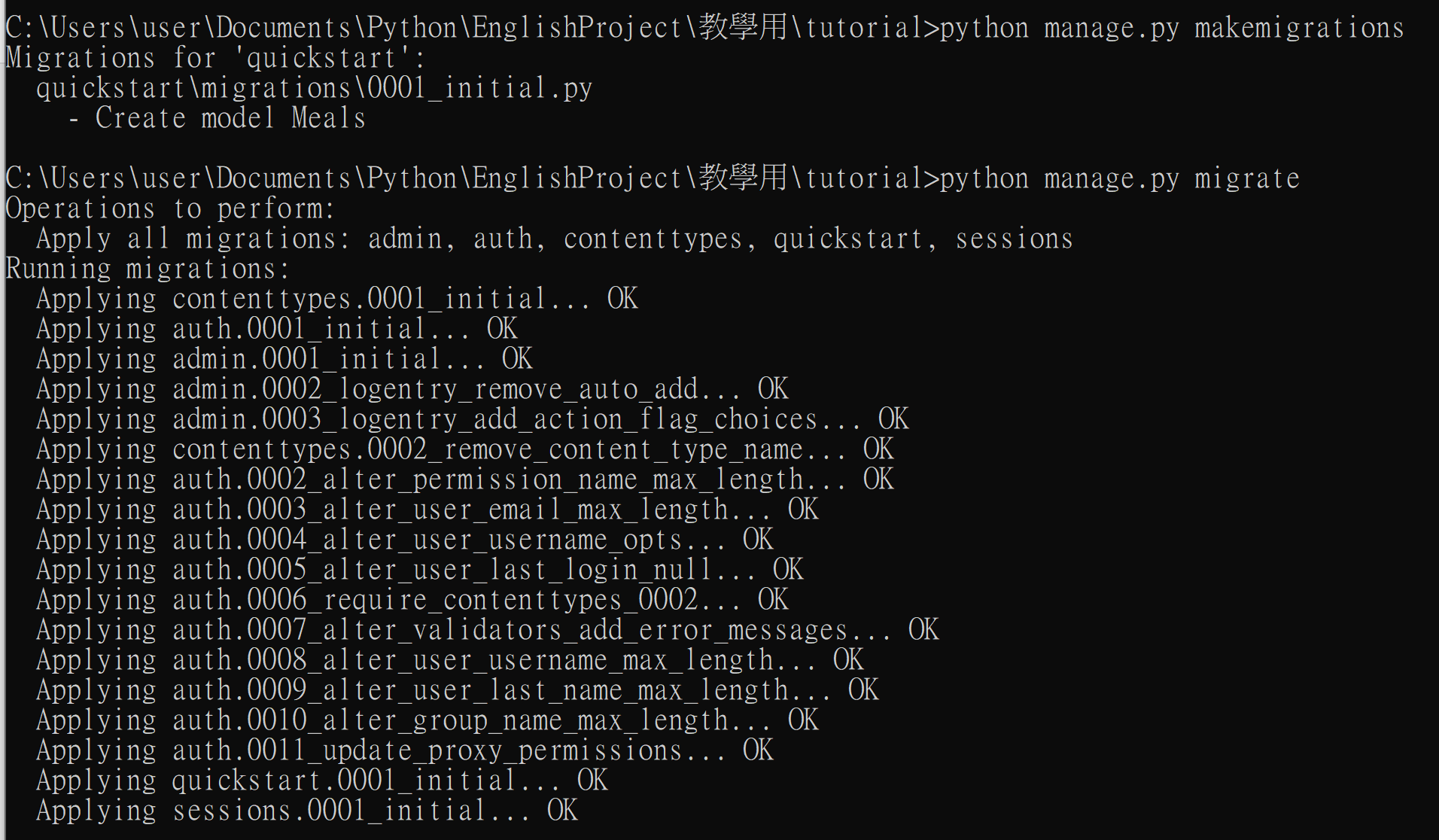
確認無誤後我們使用

python manage.py makemigrations

python manage.py migrate

將其同步至資料庫，資料建模的部分就完成了。

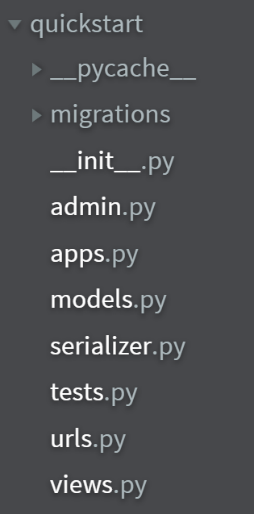
成功會如下圖。



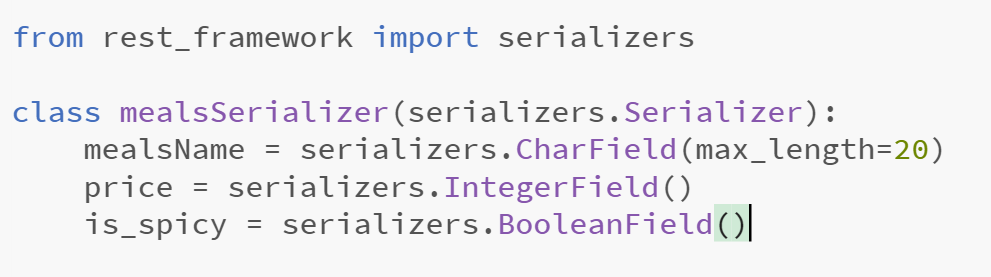
### 建構序列器

序列器可以把像queryset和model實體這種複雜格時的資料輕鬆渲染成JSON，XML或其他內容類別的原生Python類別。序列器還提供反序列化，在驗證完傳入的資料後允許解析資料轉換回複雜類別。

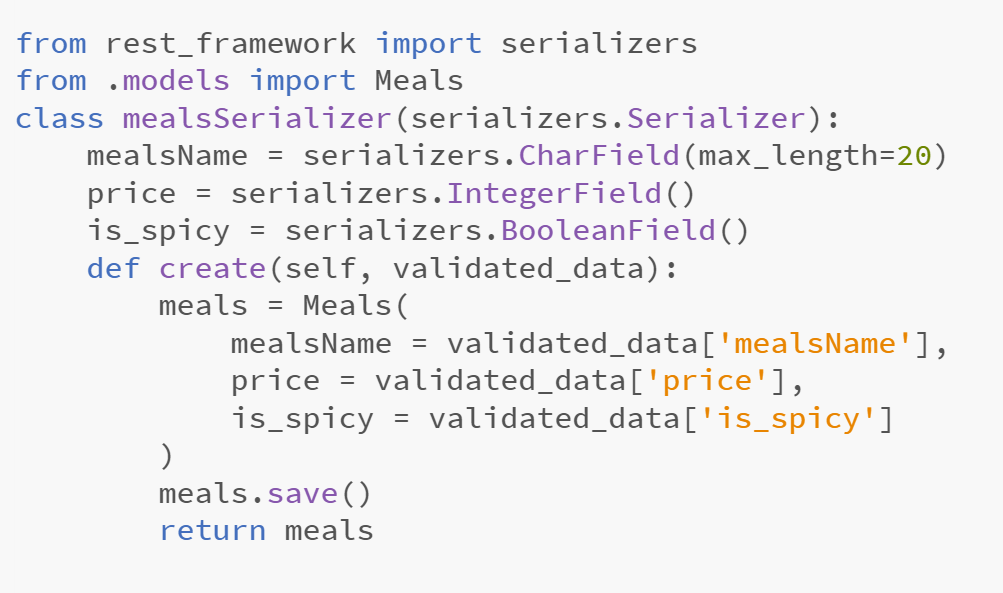
我們在APP資料夾中新增一個serializer.py檔案用來建構我們的序列器。



然後根據我們的模型定義，建構出我們要驗證的資料格式。



我們從serializers中繼承Serializer類別並定義出我們要驗證的格式。接著我們要保存驗證過的資料並實作API中CRUD的Create。



我們在序列器下方新增create方法，並在其中使用驗證過後的資料實體化一個模型並將其儲存至Database，序列器的部分就完成了。

其他更細節的操作參考以下網址:

[序列化器](https://q1mi.github.io/Django-REST-framework-documentation/api-guide/serializers_zh/)

[序列化字段](https://q1mi.github.io/Django-REST-framework-documentation/api-guide/fields_zh/)

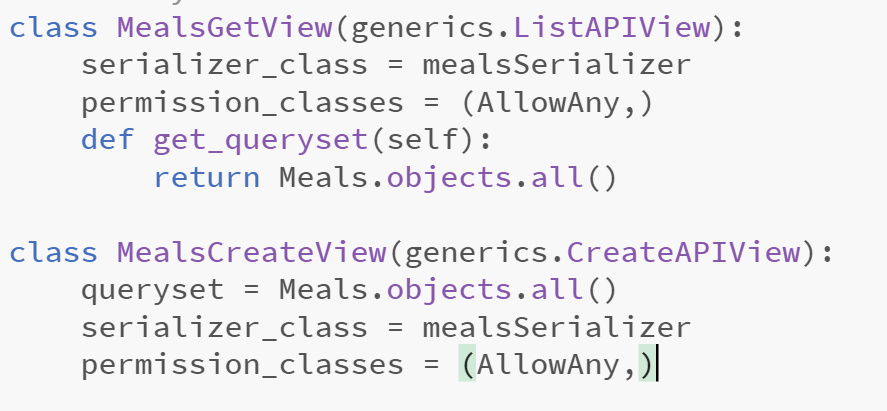
### 視圖

現在我們做好了資料模型及序列器，我們就缺一個視圖來將API接口與這些東西全部連接在一起。

現在我們打開APP資料夾中的views.py，我們等等要實作出CRUD中的Create跟Read。



首先我們先import 資料模型跟序列器還有generics，我們將使用他繼承APIView達到快速實作視圖的效果。



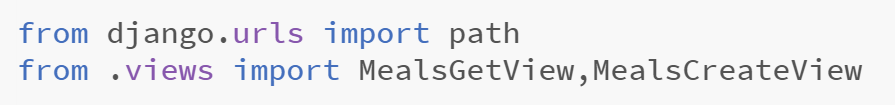
接著我們建構兩個類別分別繼承ListAPIView跟CreateAPIView，一個是可以幫你拿取所有資料，一個是新增資料的視圖，並將序列器指定為我們自己的序列器，permission\_classes代表存取權限開發時先使用AllowAny後面如果有實作JWT或其他使用者登入存取再來進行變更。

更詳細的資料請參考以下連結:

[基於類的實圖](https://q1mi.github.io/Django-REST-framework-documentation/api-guide/views_zh/)

[ViewSets](https://q1mi.github.io/Django-REST-framework-documentation/api-guide/viewsets_zh/)

Views.py的部分就完成了接著一樣開啟APP資料夾中的urls.py，我們要將視圖與接口串在一起。

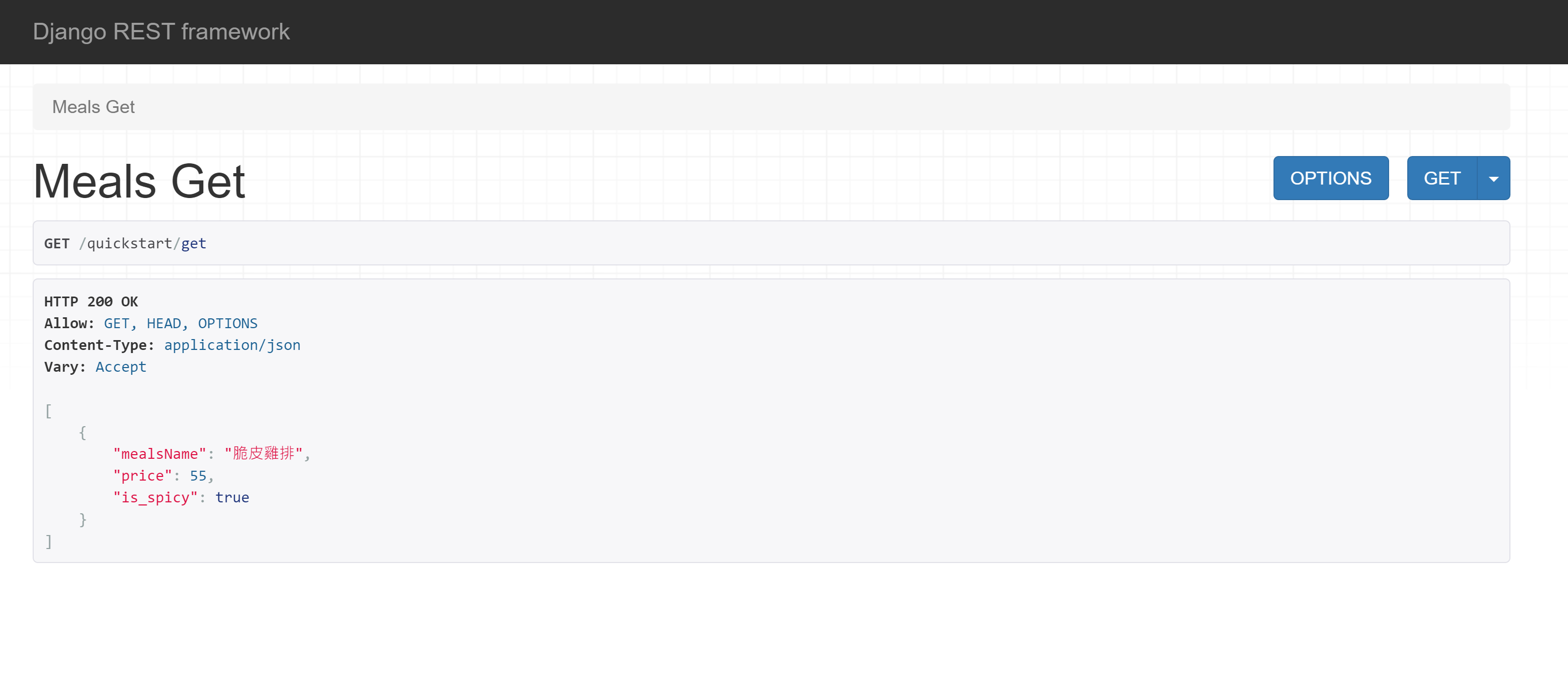


首先將views.py剛剛建立好的兩個視圖引用進來。

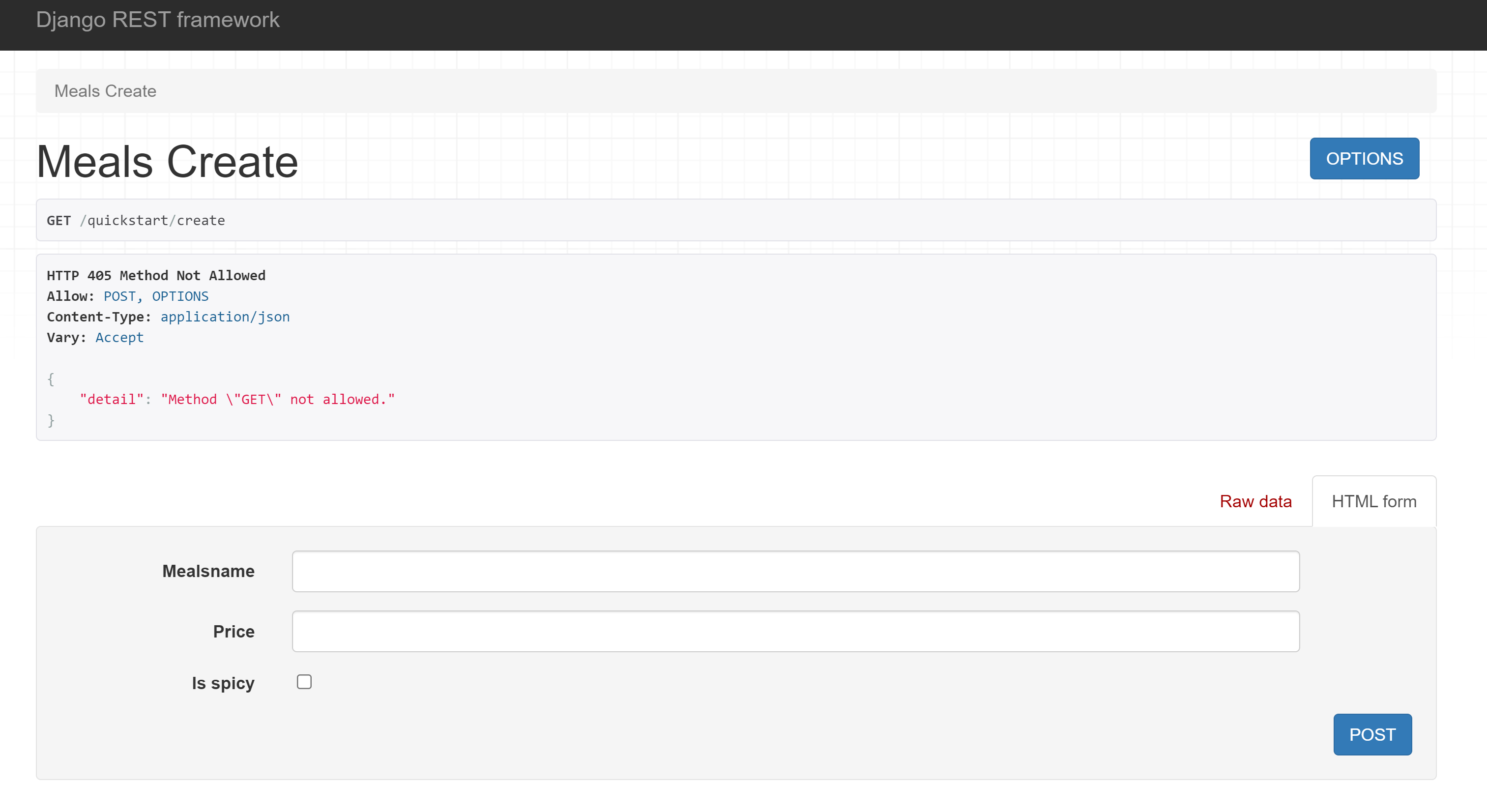


接著定義url名稱，以及指定到各個viewset，記得使用viewset的話要用as\_view()方法才會被path允許。

設定好後runserver如果正常在你的get頁面應該會顯示如下圖。



Create頁面會如下圖

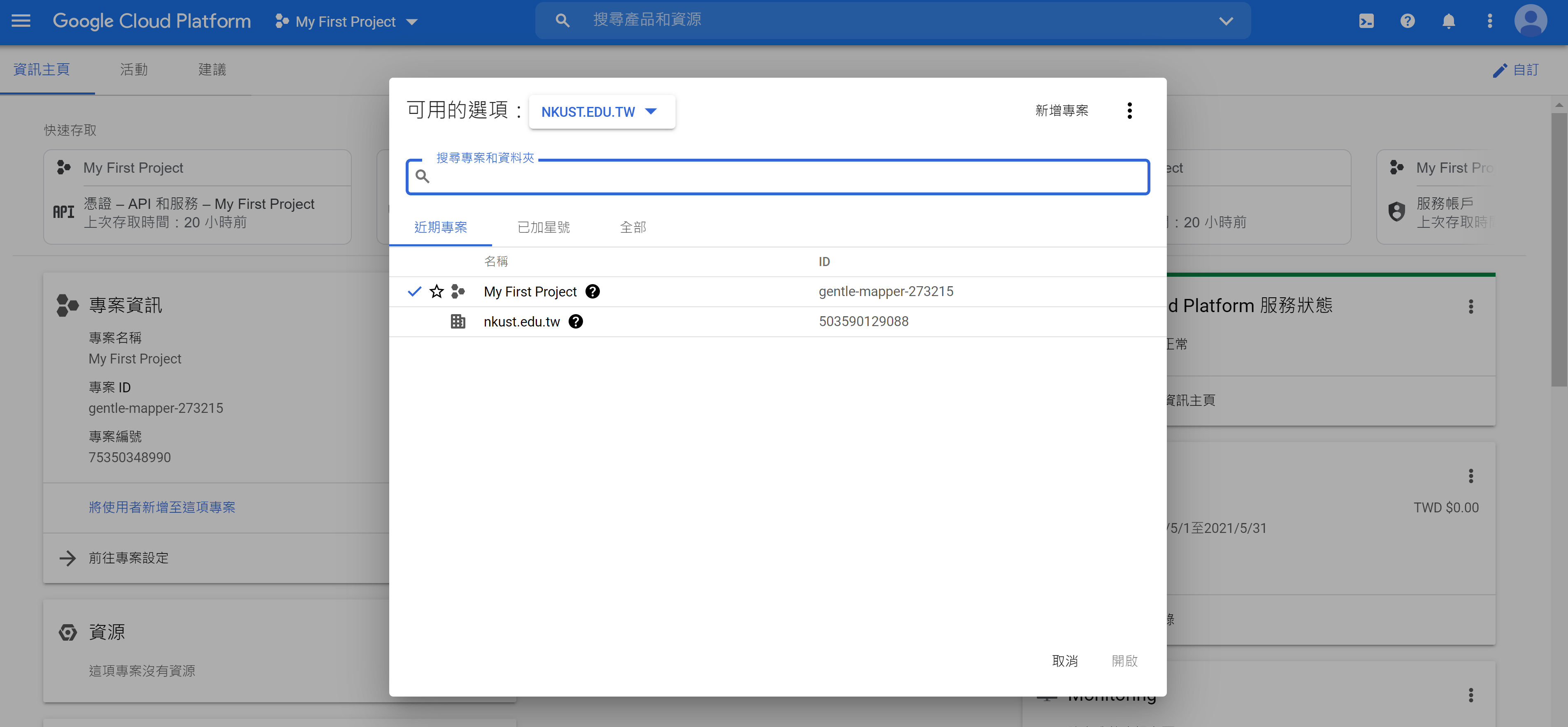


皆可在各個viewset中進行測試，也可用PostMan之類的工具進行串接測試。

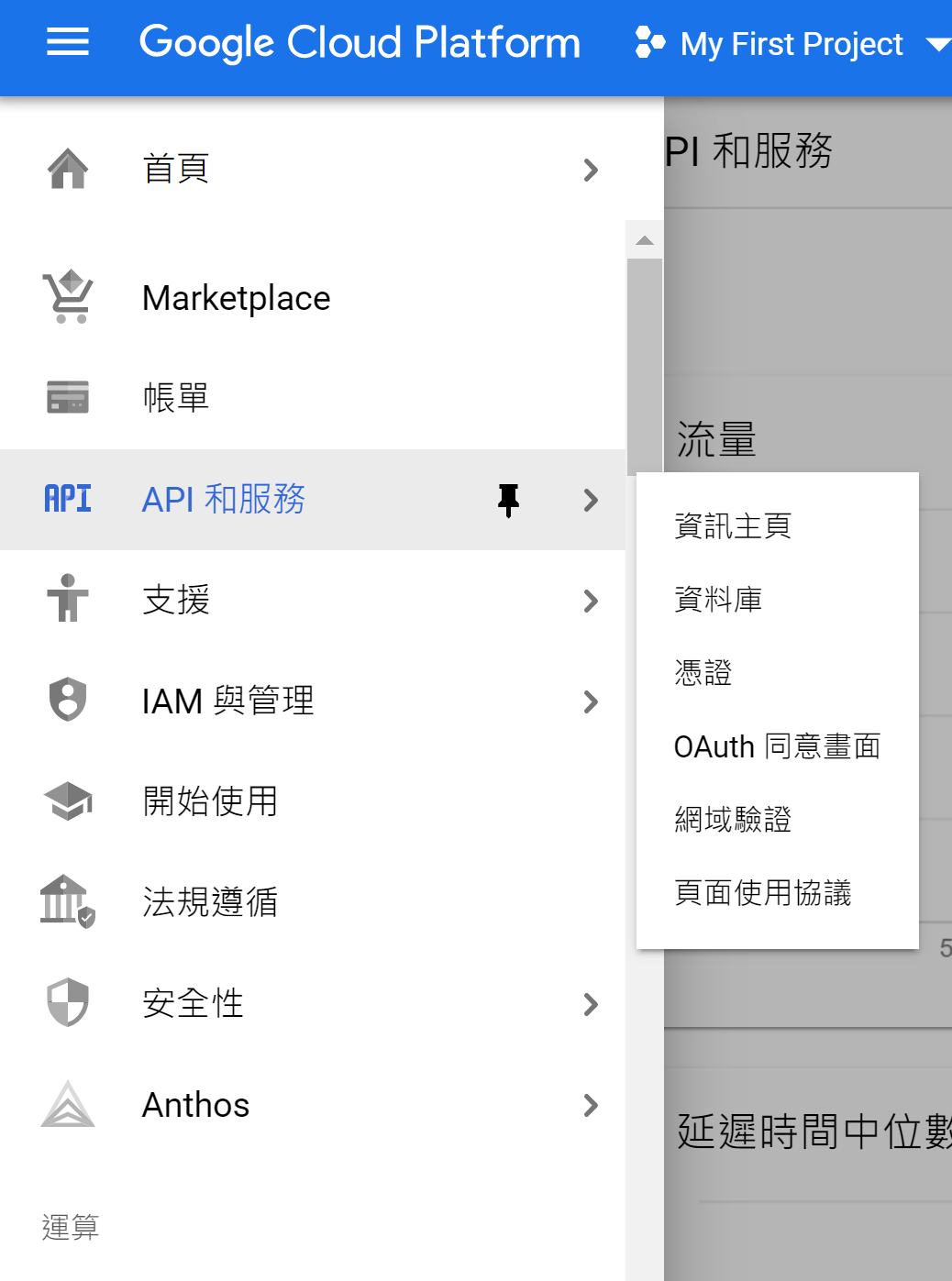
## Google Api 串接教學

前往[google cloud](https://cloud.google.com/) 平台啟動服務

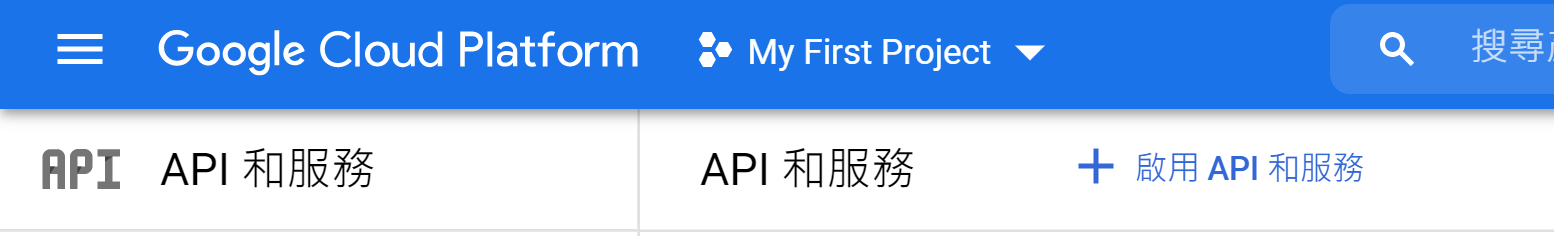
若有免費試用可點選免費試用開通帳戶。



進入控制台後先新增一個專案。



點選API和服務進入資訊主頁



點選啟用API進入選取並啟用想存取的服務。



之後到憑證頁面設定憑證並儲存到本機，即可進行API存取。

## 存取方式

首先我們先import以下google相關函式庫。



接著使用以下程式碼





將憑證json檔放在跟程式碼同目錄之下，才可存取到。

## 英翻中、中翻英(API版本)



首先在main函式中，使用SQL語法取出尚未翻譯的句子，並且限制數量。翻譯前可使用程式抓取一遍並且計算字元數，以字元數\*2不超過90000為主，以免每月免費額度過量。

之後便是對抓取出來的句子進行遞迴放入write\_data函式中進行英翻中、中翻英、並比較相似度且更新至資料庫中。

1. **使用資源及API定價**
2. **函式庫**

|  |  |  |  |
| --- | --- | --- | --- |
| 函式庫 | 憑證 | copyleft | 備註 |
| googletrans | MIT License (MIT) | no |  |
| ibm\_watson | Apache Software License (Apache 2.0) | no |  |
| ibm\_cloud\_sdk\_core | Apache Software License (Apache 2.0) | no |  |
| google | BSD License | no |  |
| google\_trans\_new | MIT License (MIT) | no | 禁止商業用 |
| whatlies | No license | yes |  |
| eng\_to\_ipa | No license | yes |  |
| wiktionaryparser | No license | yes |  |
| language\_tool\_python | GNU GPL | yes |  |
| spacy | MIT License (MIT) | no |  |
| stanza | Apache Software License (Apache 2.0) | no |  |

1. **各API收費方式**
2. Google translate api



1. [Google dialogflow](https://cloud.google.com/dialogflow/pricing#es-agent)
2. [IBM](https://www.ibm.com/tw-zh/cloud/watson-language-translator/pricing)