(It’s django C8)

<https://ithelp.ithome.com.tw/articles/10212427>

**建置project的指令**

**Django-admin.py startproject myrest**

**建置app的指令**

**Python manage.py startapp myrestapp**

**安裝app**

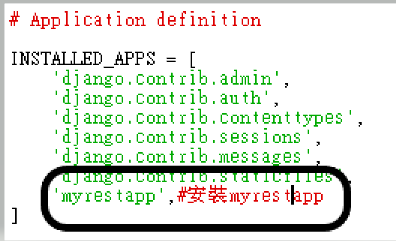
Step1:把要用的myrestapp資料夾整個移動到myrest下

(裡面必須對應的view和templates )

Step2:在myrest資料夾下的templates新增myrestapp的模板(html)

Step3:修改myrest-myrest-資料夾下setting.py檔案

🡪在INSTALLED\_APPS裡加入app的名稱



Step4:修改myrest-myrest-資料夾下的urls檔案

🡪修改或增加app裡所用到函數的名稱

urls:[C:\Users\小青\myrest\myrest\urls.py](file:///C:\Users\小青\myrest\myrest\urls.py)

view:[C:\Users\小青\myrest\myrestapp\views.py](file:///C:\Users\小青\myrest\myrestapp\views.py)

Html:[C:\Users\小青\myrest\myrestapp\templates\menu.html](file:///C:\Users\小青\myrest\myrestapp\templates\menu.html)

Setting:[C:\Users\小青\myrest\myrest\settings.py](file:///C:\Users\小青\myrest\myrest\settings.py)

**Django模型🡪ORM機制**

**(object-relational mapping物件關聯映射)**

1. 在app裡建立一個模型：

[C:\Users\小青\myrest\myrestapp\models.py](file:///C:\Users\小青\myrest\myrestapp\models.py)

1. 所有的模型都是繼承自django.db.models.Model類別

|  |  |
| --- | --- |
| Name=models.CharField(max\_length=15) | Name是資料表中的一個field object |
|  | CharField是文字資料型態的欄位物件 max\_length參數可以設定文字最大長度 blank參數可以允許欄位為空 DecimalField為浮點數的資料 max\_digits參數設定最大位數 decimal\_places參數設定小數點位數 BooleanField為布林真假值資料 |
| restaurant=models.Foreignkey(Restaurant) | 包含了重要的資料表關聯性的概念 一家餐廳有多種食物 或是一本書有許多編者或作者 這屬於一種Many-to-one的關係 這時候就要用ForeignKey(外鍵)來描述 |
| my\_id=module.AutoField(primary\_key=True) | 1. 左方的方式可自訂主鍵 2. 不然，Django預設遞增1的方式 |

**檢查資料指令**

**Python manage.py check**

**建立migration 資料檔**

1. 在建立資料表前，必須先建立migration資料檔，以確保資料表的架構和版本有被記錄下來
2. 建立migration資料檔指令：

**python manage.py makemigrations myrestapp**

1. 這會對指定的應用做migration檢查，如果模型有異動，會產生新的migration檔
2. 如果makemigrations後面不指定app名稱，則django會對所有安裝好的app做migration的檢查。當模型有更動時，makemigrations，會生成新的migration檔。
3. 可從檔名以及裡頭的dependencies和operations串列，知道之前做了哪些變動

以及是依據哪個migration做變動的

6. 使用下列指令可以知道模型是如何被翻譯成SQL語言的

**Python manage.py sqlmigrate myrestapp 0001**

**Appname 版本號**

**模型與資料庫之同步---利用migrate**

1. 將模型同步到資料庫，真正產生資料表：

**python manage.py migrate myrestapp 0001**

1. django1.7以前版本用syncdb(有用到再新增資料)

**資料操作(建立模型物件與寫入)**

1. 進入django shell

**Python manage.py shell**

1. 建立一個模型物件(生成一筆尚未被寫入資料庫中資料)

**From myrestapp.models import Myrestapp,Food**

**r1=Myrestapp(name=’QQ’,pthone\_number=’0911676878’,address=’abcde’**

1. 將模型物件(資料)寫入資料庫

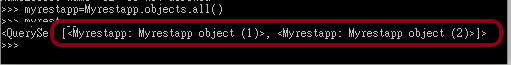
**r1.save()**

一次完成模型資料的建立與寫入資料庫

**r2=Myrestapp.objects.create(name=’QAQ’,phone\_number=’091112345’,address=’abcde’)**

1. **r1** ( 還沒退出shell可直接打r1 )

顯示出來的資料為



部會直接顯示人類看得懂的東西

1. 在models.py裡使用 \_\_unicode\_\_的方法，讓顯示出來的東西，人類可以看懂。[C:\Users\小青\myrest\myrestapp\models.py](file:///C:\\Users\\小青\\myrest\\myrestapp\\models.py)

翻譯這篇文章<https://docs.djangoproject.com/en/3.0/ref/unicode/>

**資料操作(指令整理)**

|  |  |
| --- | --- |
| 進入django shell | **python manage.py shell** |
| 建立一個模型物件 | **from myrestapp.models import Myrestapp,Food**  **r1=Myrestapp(name=’QQ’,phone\_number=’0911676878’,address=’abcde’)** |
| 將模型物件寫入資料庫 | **r1.save()** |
| 取出資料表中的資料 | **From myrestapp.models import Myrestapp,Food**  **myrestapp=Myrestapp.objects.all()**  **myrestapp** |
| 資料欄位的顯示 | **myrestapp[0].phone\_number**  **myrestapp[0].id** |
| 資料查詢與查詢集(QuerySet) | **r=Myrestapp.object.get(name=’QQ store’**  **r**  objects.get方法會回傳一個模型物件，也就是一筆資料，如果回傳的是多筆或是查詢結果失敗(空)，都會引發例外，這可以利用try/expect來捕獲並處理 |
| 全查詢 | **myrestapp=Myrestapp.objects.all()**  **myrestapp** |
| 過濾查詢  (利用object 的filter) | **myrestapp=Myrestapp.objects.filter(name=’QQ store’)**  **myrestapp** |
| 多重查詢 | **mrestapp=Myrestapp.objects.filter(name=’QQ store’,phone\_number=’0911676878’** |
| 包含查詢 | **myrestapp=Myrestapp.objects.filter(name\_\_contains=’QQ’)**  **myrestapp**  (利用 屬性名+\_\_contains=搜尋數 可以搜尋該屬性包含值得資料 |
| 排序查詢 | **Food.objects.order\_by(‘price’)**  **Food.objects.order\_by(‘-price’)**  反向排序  (使用order\_by的方法可以自行指定哪個資料進行排序)  在models.py裡加入meta🡪讓查詢每次都照著設定的規則排列 |
| 查詢級的操作與連鎖查詢 | **Food.objects.order\_by(‘price’).filter(is\_spicy=False).get(name\_\_contains=’french’)** |
| 外鍵與跨模型查詢 | **fd=Food.objects.get(name=’french fries’)🡨先把要查的東西指定給fd**  **fd.myrestapp🡨再藉由 . 這個符號往下查詢(myrestapp -- 這外鍵叫啥 -- fd)**  **v=Myrestapp.objects.get(name=’QQ store’)**  **v.food\_set 🡨用 \_set做一個類似objects的關係管理器 (Food 🡪food)**  **v.food\_set.all() 🡨 v裡面的food的所有資料** |
| 更新資料庫數據 | **fd.price 🡨查詢fd下price的值**  **fd.price=200 🡨把price的值改成200**  **fd.save()** **🡨把更改過的值儲存起來**  **Food.objects.filter(name=’french fries’).update(price=50) 🡨因為update是QuerySet的方法，無法作用在單獨物件上。所以要藉著filter出來只有french fries這個物件，去update()**  **Myrestapp.objects.get(name=’QQ store’).food\_set.update(price=100) 🡨把所有的price都update成100** |
| 刪除資料庫數據 | **fd.delete()**  **fd.objects.all().delete()** |

* Objects:

1. 管理器
2. 每個模型都有一個管理器，他包含了關於查詢該模型資料的種種方法

* QuerySet：

1. 查詢集
2. 由objects回傳出來的這個資料清單(嚴格來說不是我們所熟知的那個清單)
3. 他是一個類似於清單的物件，把它想像成是儲存著藉由管理器查詢回來的資料集合