

Network Programming

- 網路程式設計 -

資料庫及後台管理

授課教師：張珀銀 老師

04

資料庫及後台管理

1. 4-1 Django 與資料庫
2. 4-2 admin 後台管理與 ModelAdmin 類別
3. 4-3 資料庫查詢
4. 4-4 網頁基礎模版
5. 課堂練習

4-1 Django 與資料庫

4.1.1 使用Django 資料庫

要在Django 中使用資料庫，有下列幾個重要的步驟：

1. 在<model.py> 檔中定義class 類別，每一個類別相當於一個資料表。
2. 在class 類別中定義變數，每一個變數相當於一個資料表欄位。
3. 以「python manage.py makemigrations」建立資料庫和Django 間的中介檔。
4. 以「python manage.py migrate」同步更新資料庫內容。
5. 在Python 程式中存取資料庫。

Object-relational Mappers (ORMs)

An object-relational mapper (ORM) is a code library that automates the transfer of data stored in relational database tables into objects that are more commonly used in application code.

Relational database (such as PostgreSQL or MySQL)

ID	FIRST_NAME	LAST_NAME	PHONE
1	John	Connor	+16105551234
2	Matt	Makai	+12025555689
3	Sarah	Smith	+19735554512
...

Python objects

```
class Person:  
    first_name = "John"  
    last_name = "Connor"  
    phone_number = "+16105551234"
```

```
class Person:  
    first_name = "Matt"  
    last_name = "Makai"  
    phone_number = "+12025555689"
```

```
class Person:  
    first_name = "Sarah"  
    last_name = "Smith"  
    phone_number = "+19735554512"
```

ORMs provide a bridge between
**relational database tables, relationships
and fields** and **Python objects**

Why are ORMs useful?

- ORMs provide a high-level abstraction upon a relational database that allows a developer to write Python code instead of SQL to create, read, update and delete data and schemas in their database.
- Developers can use the programming language they are comfortable with to work with a database instead of writing SQL statements or stored procedures.

Why are ORMs useful?

For example, without an ORM a developer would write the following SQL statement to retrieve every row in the USERS table where the zip_code column is 94107:

```
SELECT * FROM USERS WHERE zip_code=94107;
```

The equivalent [Django ORM](#) query would instead look like the following Python code:

```
# obtain everyone in the 94107 zip code and assign to users  
variable
```

```
users = Users.objects.filter(zip_code=94107)
```

Why are ORMs useful?





- 用Python代碼代替SQL可以加速WEB應用程式的開發，特別是在專案的開始階段。潛在的開發速度提升來自於不必從Python代碼切換到編寫聲明式範式SQL語句。
- 雖然一些軟體發展人員可能不介意在語言之間來回切換，但僅使用一種程式設計語言建立原型或啟動Web應用程式通常更容易管理。
- ORM使應用程式在各種關聯式資料庫之間切換有理論上的可能。例如，開發人員可以使用SQLite進行本地開發，而在生產環境中使用MySQL。一個已正式上線的應用程式可以從MySQL切換到PostgreSQL，只需修改少量的程式。

Why are ORMs useful?

- 實務上建議正式營運環境與開發環境皆使用相同的資料庫進行開發，避免在正式營運環境中出現意外錯誤。
- 若非緊迫情況，專案極少切換不同規格資料庫。

Do I have to use an ORM for my web application?

- Python 連結關聯式資料庫並非一定要採用Python ORM
- 底層連結通常由另一個稱為資料庫連接器 (Connector)的函式庫提供；
下表展示ORM如何與不同的Web框架、連接器和關聯式資料庫的方案。

web framework	None	Flask	Flask	Django
ORM	SQLAlchemy	SQLAlchemy	SQLAlchemy	Django ORM
database connector	(built into Python stdlib)	MySQL-python	psycopg	psycopg
relational database	 SQLite	 MySQL	 PostgreSQL	 PostgreSQL

What are the downsides of using an ORM?

ORM有許多缺點，包括

- 阻抗失配 (Impedance mismatch)
- 性能下降的可能性 (Potential for reduced performance)
- 將複雜性從資料庫轉移到應用程式碼
(Shifting complexity from the database into the application code)

Python ORM Implementations

There are numerous ORM implementations written in Python, including

- SQLAlchemy
- Peewee
- The Django ORM
- PonyORM
- SQLAlchemy
- Tortoise ORM (source code)

Django's ORM

- Django web框架自帶內置的物件關係映射模組，通常被稱為“Django ORM”或“Django的ORM”。
- Django的ORM對於簡單和中等複雜度的資料庫操作來說效果很好。但仍有部分使用者抱怨ORM使複雜的查詢，不如直接編寫SQL或使用SQLAlchemy。
- 從技術上實現整合SQL，但須將查詢綁定到特定的資料庫實現。
- ORM與Django緊密地結合在一起，若用SQLAlchemy替換默認的ORM，則是較為困難的解決方案。
- 將來可能會出現可交換的ORM後端，然而大多數Django項目都綁定默認ORM，故若要改用ORM，最好閱讀一些進階範例及工具，以最有效率的方式的完成工作。

4.1.2 定義資料模型

建立資料模型

首先建立student 類別，建立的類別必須以「class student(models.Model)」繼承models.Model，然後在student 類別建立欄位。

```
studentsapp\models.py

from django.db import models

class student(models.Model):
    cName = models.CharField(max_length=20, null=False)
    cSex = models.CharField(max_length=2, default='M', null=False)
    cBirthday = models.DateField(null=False)
    cEmail = models.EmailField(max_length=100, blank=True, default='')
    cPhone = models.CharField(max_length=50, blank=True, default='')
    cAddr = models.CharField(max_length=255, blank=True, default='')

    def __str__(self):
        return self.cName
```

models.Model 常用的欄位

models.Model 常用的欄位如下表：

欄位格式	參數	說明
BooleanField		True 或 False，用於 checkbox 輸入資料。
CharField	max_length：最大字串長度	用於單行輸入字串資料。
DateField	auto_now：物件儲存時自動取得目前的日期。 auto_now_add：只有在物件建立時才加入目前的日期。	日期格式，即 datetime.date。
DateTimeField	auto_now：物件儲存時自動取得目前的日期時間。 auto_now_add：只有在物件建立時才加入目前的日期時間。	日期時間格式，即 datetime.datetime。
EmailField	max_length：最大字串長度 254 個字元。	電子郵件。
FloatField		浮點數。
IntegerField		整數，值由 -2147483648~2147483647。
PositiveIntegerField		正整數，值由 0 到 2147483647。
TextField		多行輸入字串資料，用於 HTML 表單的 textarea 欄位。

models.Model 欄位的屬性

models.Model 欄位的屬性如下表：

欄位選項	說明
null	欄位是否可為 null 值，預設為 False。
blank	欄位是否可為空白內容，預設為 False。
choices	設定 select 欄位的選項。例如：以 items 元組定義選項。 class student(models.Model): items = (('JUNIOR', 'Junior'),('SENIOR', 'Senior'),) year_in_school = models.CharField(choices=items ,)
default	欄位預設值。
editable	設定此欄位是否可顯示，預設為 True。
primary_key	設定此欄位是否為主鍵，預設為 False。
unique	設定此欄位是否為唯一值，預設為 False。

建立 migration 資料檔

```
python manage.py makemigrations
```

模型與資料庫同步

```
python manage.py migrate
```


4-2 admin 後台管理與 ModelAdmin 類別

4.2.1 admin 後台管理

```
studentsapp\admin.py  
  
from django.contrib import admin  
from studentsapp.models import student  
  
# Register your models here.  
admin.site.register(student)
```

請記得再啟動伺服器，輸入「127.0.0.1:8000/admin/」這個網址，將會開啟admin的登入畫面。



建立管理者帳號和密碼

```
python manage.py createsuperuser
```

啟動伺服器，在瀏覽器輸入「127.0.0.1:8000/admin/」，輸入帳號和密碼後，將會進入Django 管理頁面，其中的Students 就是我們向admin 註冊的資料表。



新增資料

按 **新增** 鈕，進入資料輸入的頁面，輸入資料後按 **儲存** 鈕。

STUDENTSAPP

Students + 新增

認證與授權

使用者 + 新增

群組 + 新增

新增 student

CName : 李采茜

CSex : F

CBirthday : 1987/04/04 今天 | 📅

CEmail : elvan@superstar.com

CPhone : 0976123456

CAddr : 台北市信義路643號

儲存並新增另一個 儲存並繼續編輯 儲存

點選指定的資料即可編輯該筆資料，進入資料編輯頁面後可以修改或刪除該筆資料，按 **新增 STUDENT +** 鈕可以新增更多資料。

STUDENTSAPP

Students + 新增

認證與授權

使用者 + 新增

群組 + 新增

✓ The student "李采茜" was added successfully.

選擇 student 來變更

動作: [dropdown] 去 1 中 0 個被選

☐ STUDENT

☐ 李采茜

1 student

新增 STUDENT +

新增更多的資料

新增的資料，點選後可編輯該筆資料

4.2.2 定義 ModelAdmin 類別

顯示多個欄位

例如：定義studentAdmin 類別，此類別必須繼承admin.ModelAdmin 類別，並以list_display 定義顯示id、cName、cSex、cBirthday、cEmail、cPhone 和 cAddr 欄位。(註：id 是系統自動產生的欄位，它會由0 開始加1 遞增)

```
studentsapp\admin.py
```

```
略.....
```

```
8 class studentAdmin(admin.ModelAdmin):  
9     list_display=('id','cName','cSex','cBirthday','cEmail', 'cPhone','cAddr')  
10 admin.site.register(student,studentAdmin)
```

資料過濾

list_filter 可以建立過濾欄位。例如：「list_filter=('cName','cSex)」以cName和cSex 欄位建立過濾欄位。注意：過濾欄位必須使用串列或元組，完成後會在右方出現過濾的欄位。

依欄位搜尋

search_fields 可以設定依指定的欄位搜尋。例如：「search_fields=('cName',)」依cName 欄位搜尋，搜尋欄位必須使用串列或元組，完成後會在上方出現塊文字方塊，輸入資料後按 搜尋 鈕，即會依cName 欄位搜尋。

排序

ordering 可以依指定的欄位排序，例如：「ordering=('id',)」依id 欄位遞增排序，排序欄位必須使用串列或元組。如果加上「-」則可設定為遞減排序，例如：「ordering=(-'id',)」。

範例：建立studentAdmin 類別，定義資料顯示欄位、資料過濾欄位、資料搜尋欄位和資料排序欄位。

搜尋欄位

動作: 8 中 0 個被選

<input type="checkbox"/>	ID	CNAME	CSEX	CBIRTHDAY	CEMAIL	CPHONE	CADDR
<input type="checkbox"/>	1	李采茜	F	1987年4月4日	elvan@superstar.com	0976123456	台北市信義路643號
<input type="checkbox"/>	2	林木田	M	2001年8月9日	david@superstar.com	0978125643	台北市仁愛路43號
<input type="checkbox"/>	3	田出火	M	1998年12月30日	lily@superstar.com	0945678912	台北市忠孝東路12號
<input type="checkbox"/>	4			2008年6月24日	beauty@superstar.com	0912563478	台北市信義路98號
<input type="checkbox"/>	5			1980年8月1日	joe@superstar.com	0956784321	台北市和平東路56號
<input type="checkbox"/>	6	黃竹林	M	1994年1月29日	john@superstar.com	0966665555	台北市南昌路65號
<input type="checkbox"/>	7	邱諒全	M	2005年4月23日	goan@superstar.com	0913427777	台北市信義路1008號
<input type="checkbox"/>	8	黃珠文	F	1992年4月25日	mary@superstar.com	0933338888	台北市南海路13號

排序欄位

過濾器

以 cName

- 全部
- 張美雲
- 李采茜
- 林木田
- 田出火
- 邱諒全
- 鄭麗珠
- 黃珠文
- 黃竹林

過濾欄位

以 cSex

- 全部
- F
- M

4-3 資料庫查詢

4.3.1 get() 方法

get() 方法可以取得一筆資料，語法：

```
資料表.objects.get(查詢條件)
```

範例：以Template 樣版顯示資料

執行「<http://127.0.0.1:8000/listone/>」以get() 方法取得student 資料表中「cName="李采茜"」的資料，並以<listone.html> 樣版顯示這筆資料。



4.3.2 objects.all() 方法

objects.all() 可以讀取資料表中所有資料，傳回是QuerySet 型別的資料，它類似串列，語法：

```
資料表.objects.all()
```

例如：讀取student 資料表所有資料。

```
student.objects.all()
```

也可以將資料以「order_by("欄位")」依指定的欄位排序，預設是遞增排序，若欄位前加上「-」則是遞減排序。例如：依id 欄位遞減排序。(id 欄位是系統自動產生的欄位)

```
students = student.objects.all().order_by('-id')
```


範例：以Template 樣版顯示全部資料

執行「<http://127.0.0.1:8000/listall/>」以all() 方法取得student 所有資料表並以<listall.html> 樣版顯示。



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/listall/'. The page title is '顯示所有資料'. Below the title, the text '顯示 student 資料表所有資料' is displayed. A table with 6 columns (編號, 姓名, 性別, 生日, 郵件帳號, 電話, 地址) and 8 rows of student data is shown.

編號	姓名	性別	生日	郵件帳號	電話	地址
1	李采茜	F	1987年4月4日	elvan@superstar.com	0976123456	台北市信義路643號
2	林木田	M	2001年8月9日	david@superstar.com	0978125643	台北市仁愛路43號
3	田出火	M	1998年12月30日	lily@superstar.com	0945678912	台北市忠孝東路12號
4	張美麗	F	2008年6月24日	beauty@superstar.com	0912563478	台北市信義路98號
5	鄭麗珠	F	1980年8月1日	joe@superstar.com	0956784321	台北市和平東路56號
6	黃竹林	M	1994年1月29日	john@superstar.com	0966665555	台北市南昌路65號
7	邱諒金	M	2005年4月23日	goan@superstar.com	0913427777	台北市信義路1008號
8	黃珠文	F	1992年4月25日	mary@superstar.com	0933338888	台北市南海路13號

4-4 網頁基礎模版

網站中各網頁通常會有相同的部分，以維持網站一致性風格，例如同樣的頁首及頁尾。在實務上通常會將相同的部分抽離出來，另外建立為基礎模版，然後在各網頁中繼承基礎模版，未來若需要修改這些共同的部分，只要修改基礎模版即可。

在基礎模版中首先將共同的部分分離出來，也可以「`{% block 名稱 %}{% endblock %}`」定義區塊，而在呼叫模版必須以「`{% extends 基礎模版 %}`」繼承基礎模版，並設定block 區塊內容。

為了方便對照，以<listall.html> 樣版為例，將它拆開為<base.html> 基礎模版和<index.html> 呼叫模版。執行「http://127.0.0.1:8000/」或「http://127.0.0.1:8000/index/」的畫面。



顯示所有資料

標題

內容

顯示 student 資料表所有資料

編號	姓名	性別	生日	郵件帳號	電話	地址
1	李采茜	F	1987年4月4日	elvan@superstar.com	0976123456	台北市信義路643號
2	林木田	M	2001年8月9日	david@superstar.com	0978125643	台北市仁愛路43號
3	田出火	M	1998年12月30日	lily@superstar.com	0945678912	台北市忠孝東路12號
4	張美麗	F	2008年6月24日	beauty@superstar.com	0912563478	台北市信義路98號
5	鄭麗珠	F	1980年8月1日	joe@superstar.com	0956784321	台北市和平東路56號
6	黃竹林	M	1994年1月29日	john@superstar.com	0966665555	台北市南昌路65號
7	邱諒金	M	2005年4月23日	goan@superstar.com	0913427777	台北市信義路1008號
8	黃珠文	F	1992年4月25日	mary@superstar.com	0933338888	台北市南海路13號

<base.html> 基礎模版建立<html> 標籤包含 <head> 、<body> 架構，並以 block 定義 title 、content 區塊。

templates\base.html

```
1 <!-- base.html -->
2 <!DOCTYPE html>
3 <html>
4 <head>
5     {% block title %}{% endblock %}
6 </head>
7 <body>
8     {% block content %} {% endblock %}
9 </body>
10 </html>
```

<index.html> 繼承<base.html> 基礎模版，並建立 title、content 區塊。

templates\index.html

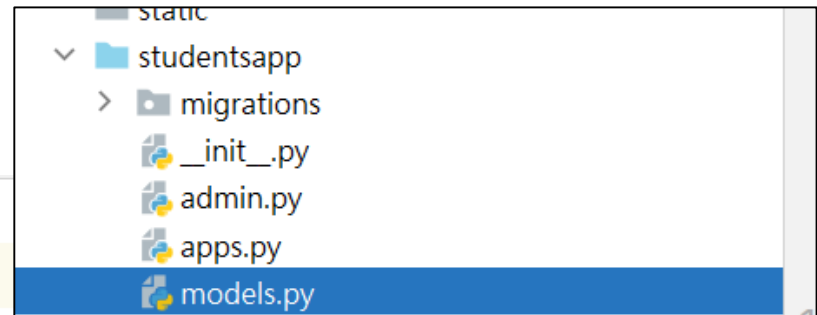
```
1  {% extends 'base.html' %}
2  {% block title %}<title> 顯示所有資料 </title>{% endblock %}
3  {% block content %}
4  <h2 align="left"> 顯示 student 資料表所有資料 </h2>
5  <table border="1" cellpadding="0" cellspacing="0">
6      <th> 編號 </th><th> 姓名 </th><th> 性別 </th><th> 生日 </th>
7      <th> 郵件帳號 </th><th> 電話 </th><th> 地址 </th>
8      {% for student in students %}
9          <tr>
10             <td>{{ student.id }} </td>
11             <td>{{ student.cName }} </td>
12             <td>{{ student.cSex }} </td>
13             <td>{{ student.cBirthday }} </td>
14             <td>{{ student.cEmail }} </td>
15             <td>{{ student.cPhone }} </td>
16             <td>{{ student.cAddr }} </td>
17         </tr>
18     {% endfor %}
19 </table>
20 {% endblock %}
```



Exercise 1: ListOne

models.py

```
from django.db import models
```



```
class Student(models.Model):
```

```
    cName = models.CharField(max_length=20, null=False)
```

```
    cSex = models.CharField(max_length=2, default='M', null=False)
```

```
    cBirthday = models.DateField(null=False)
```

```
    cEmail = models.EmailField(max_length=100, blank=True, default='')
```

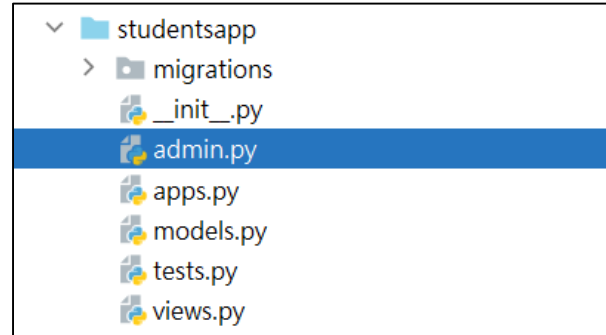
```
    cPhone = models.CharField(max_length=50, blank=True, default='')
```

```
    cAddr = models.CharField(max_length=255, blank=True, default='')
```

```
    def __str__(self):
```

```
        return self.cName
```

admin.py



```
from django.contrib import admin

from studentsapp.models import Student

admin.site.register(Student)
```


url.py

```
from django.contrib import admin
from django.urls import path
from studentsapp.views import listone, list_all, list_index

urlpatterns = [
    path('admin/', admin.site.urls),
    path('listone/', listone),
    path('list_all/', list_all),
    path('index/', list_index),
]
```

views.py

```
from studentsapp.models import Student

# Create your views here.
def listone(request):
    try:
        unit = Student.objects.get(cName='靜香')
    except:
        error_message = 'data reading errors'
    return render(request, 'listone.html', locals())
```

listone.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Display one record</title>
</head>
<body>
  <H2>Display student table's one record</H2>
  ID: {{ unit.id }}<br>
  Name: {{ unit.cName }}<br>
  Sex: {{ unit.cSex }}<br>
  Birthday: {{ unit.cBirthday }}<br>
  Email: {{ unit.cEmail }}<br>
  Phone: {{ unit.cPhone }}<br>
  Addr: {{ unit.cAddr }}<br>
</body>
</html>
```

Display student table's one record

ID: 1

Name: 靜香

Sex: F

Birthday : March 1, 2022

Email: june@gmail.com

Phone: 0932

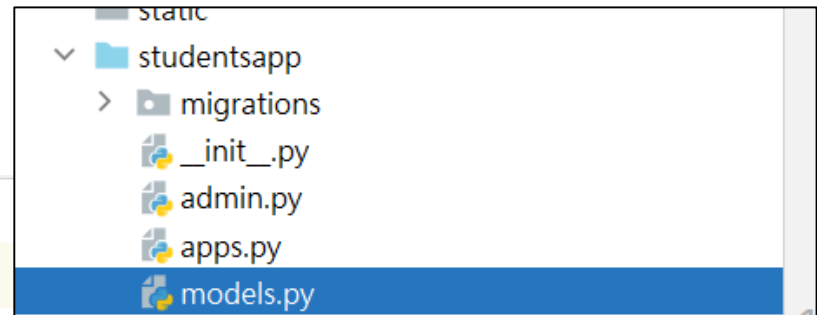
Addr: address01



Exercise 2: list_all

models.py

```
from django.db import models
```



```
class Student(models.Model):
```

```
    cName = models.CharField(max_length=20, null=False)
```

```
    cSex = models.CharField(max_length=2, default='M', null=False)
```

```
    cBirthday = models.DateField(null=False)
```

```
    cEmail = models.EmailField(max_length=100, blank=True, default='')
```

```
    cPhone = models.CharField(max_length=50, blank=True, default='')
```

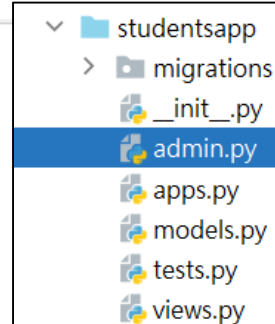
```
    cAddr = models.CharField(max_length=255, blank=True, default='')
```

```
    def __str__(self):
```

```
        return self.cName
```

admin.py

```
from django.contrib import admin  
  
from studentsapp.models import Student
```



```
#  
# admin.site.register(Student)
```

```
class StudentAdmin(admin.ModelAdmin):  
    list_display = ('id', 'cName', 'cSex', 'cBirthday', 'cEmail', 'cPhone', 'cAddr')  
    list_filter = ('cName', 'cSex', 'cBirthday')  
    search_fields = ('cName',)  
    ordering = ('id',)
```

試看看不同的display

試看看不同的filter

```
admin.site.register(Student, StudentAdmin)
```

url.py

```
from django.contrib import admin
from django.urls import path
from studentsapp.views import listone, list_all, list_index

urlpatterns = [
    path('admin/', admin.site.urls),
    path('listone/', listone),
    path('list_all/', list_all),
    path('index/', list_index),
]
```


views.py

```
from django.shortcuts import render
from studentsapp.models import Student

# Create your views here.
def listone(request):
    try:
        unit = Student.objects.get(cName='靜香')
    except:
        error_message = 'data reading errors'
    return render(request, 'listone.html', locals())

def list_all(request):
    students = Student.objects.all().order_by('id')
    return render(request, 'list_all.html', locals())
```

list_all.html

```
</head>
<body>
<table border="1">
  <th>id</th>
  <th>name</th>
  <th>cSex</th>
  <th>cBirthday</th>
  <th>cEmail</th>
  <th>cPhone</th>
  <th>cAddr</th>
  {% for s in students %}
    <tr>
      <td>{{ s.id }}</td>
      <td>{{ s.cName }}</td>
      <td>{{ s.cSex }}</td>
      <td>{{ s.cBirthday }}</td>
      <td>{{ s.cEmail }}</td>
      <td>{{ s.cPhone }}</td>
      <td>{{ s.cAddr }}</td>
    </tr>
  {% endfor %}
</table>
</body>
</html>
```

Results

id	name	cSex	cBirthday	cEmail	cPhone	cAddr
1	靜香	F	March 1, 2022	june@gmail.com	0932	address01
2	大雄	M	March 9, 2022	bear@gmail.com	0955	address02



Exercise 3: index.html

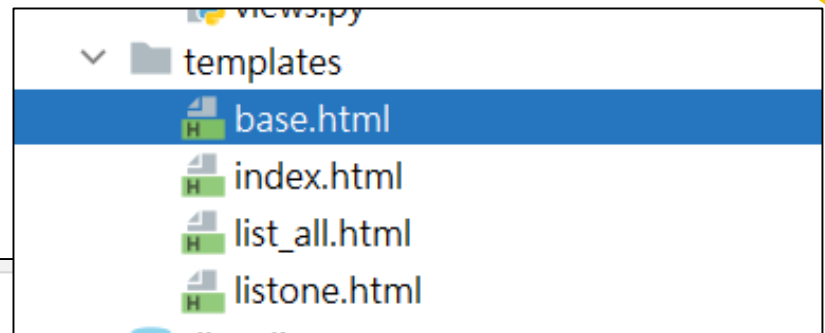
url.py

```
from django.contrib import admin
from django.urls import path
from studentsapp.views import listone, list_all, list_index

urlpatterns = [
    path('admin/', admin.site.urls),
    path('listone/', listone),
    path('list_all/', list_all),
    path('index/', list_index),
]
```

```
def list_index(request):  
    students = Student.objects.all().order_by('id')  
    return render(request, 'index.html', locals())
```

base.html



```
<!DOCTYPE html>
<html lang="en">
<head>
    {% block title %}{% endblock %}
</head>
<body>
    {% block content %}{% endblock %}
</body>
</html>
```

index.html

```
{% extends 'base.html' %}
{% block title %}<title> Display all data</title>{% endblock %}
{% block content %}
    <h2 align="left">Display all student data via templates index.html</h2>
    <table border="1">
        <th>id</th>
        <th>name</th>
        <th>cSex</th>
        <th>cBirthday</th>
        <th>cEmail</th>
        <th>cPhone</th>
        <th>cAddr</th>
        {% for s in students %}
            <tr>
                <td>{{ s.id }}</td>
                <td>{{ s.cName }}</td>
                <td>{{ s.cSex }}</td>
                <td>{{ s.cBirthday }}</td>
                <td>{{ s.cEmail }}</td>
                <td>{{ s.cPhone }}</td>
                <td>{{ s.cAddr }}</td>
            </tr>
        {% endfor %}
    </table>
{% endblock %}
```

