

前置安裝 (Anaconda Prompt)

- `pip install django==3.2`—Django(建議使用3.2以免出現xampp相容問題)
- `django-admin --version` —確認Django有無安裝完成
- `pip install mysqlclient` --連接MySQL資料庫

前置安裝(資料庫)

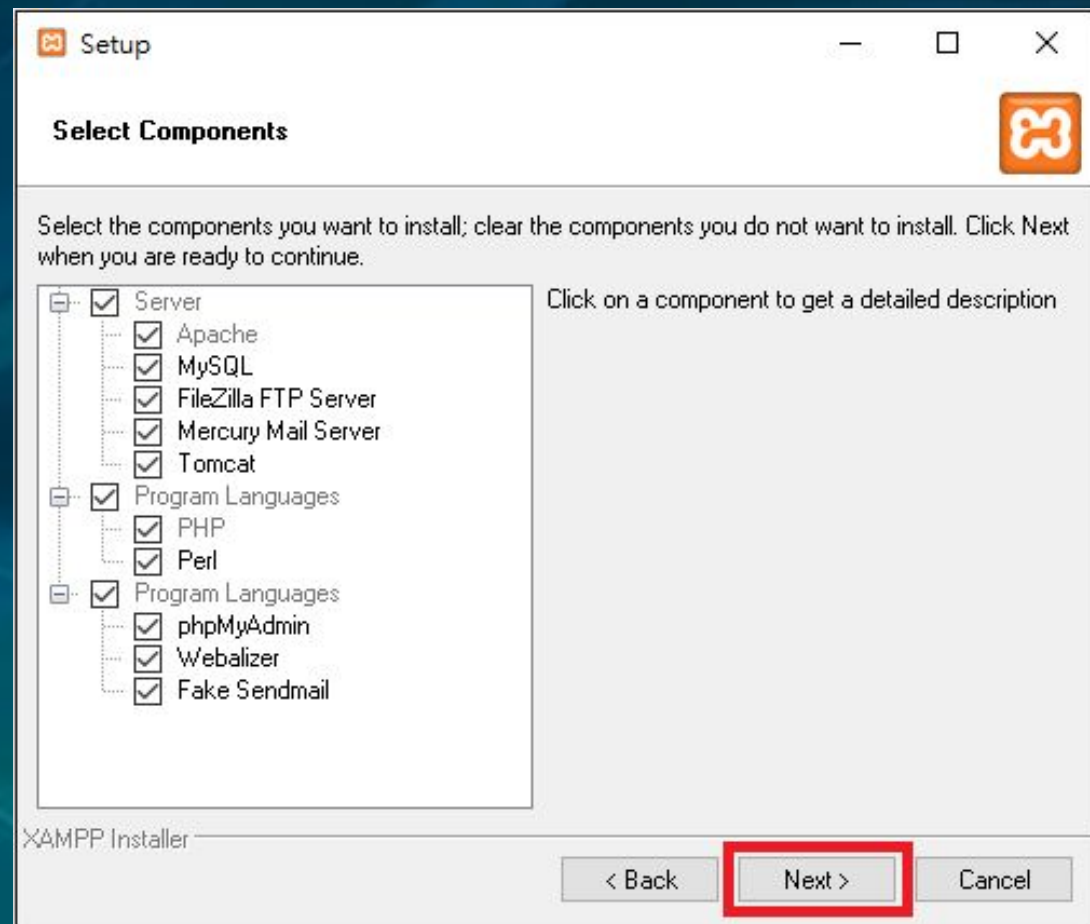
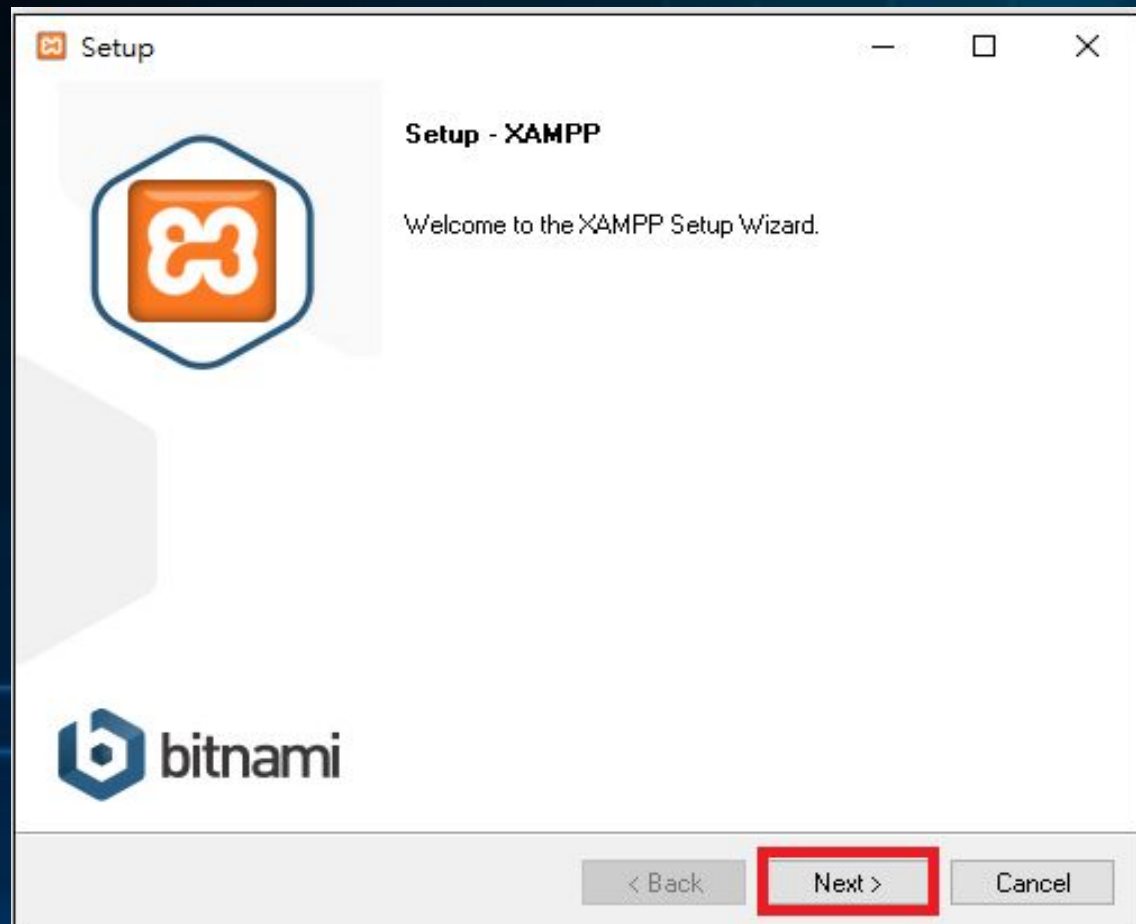
- 到 <https://www.apachefriends.org/>

下載XAMPP(Apache、MariaDB、PHP、Perl)



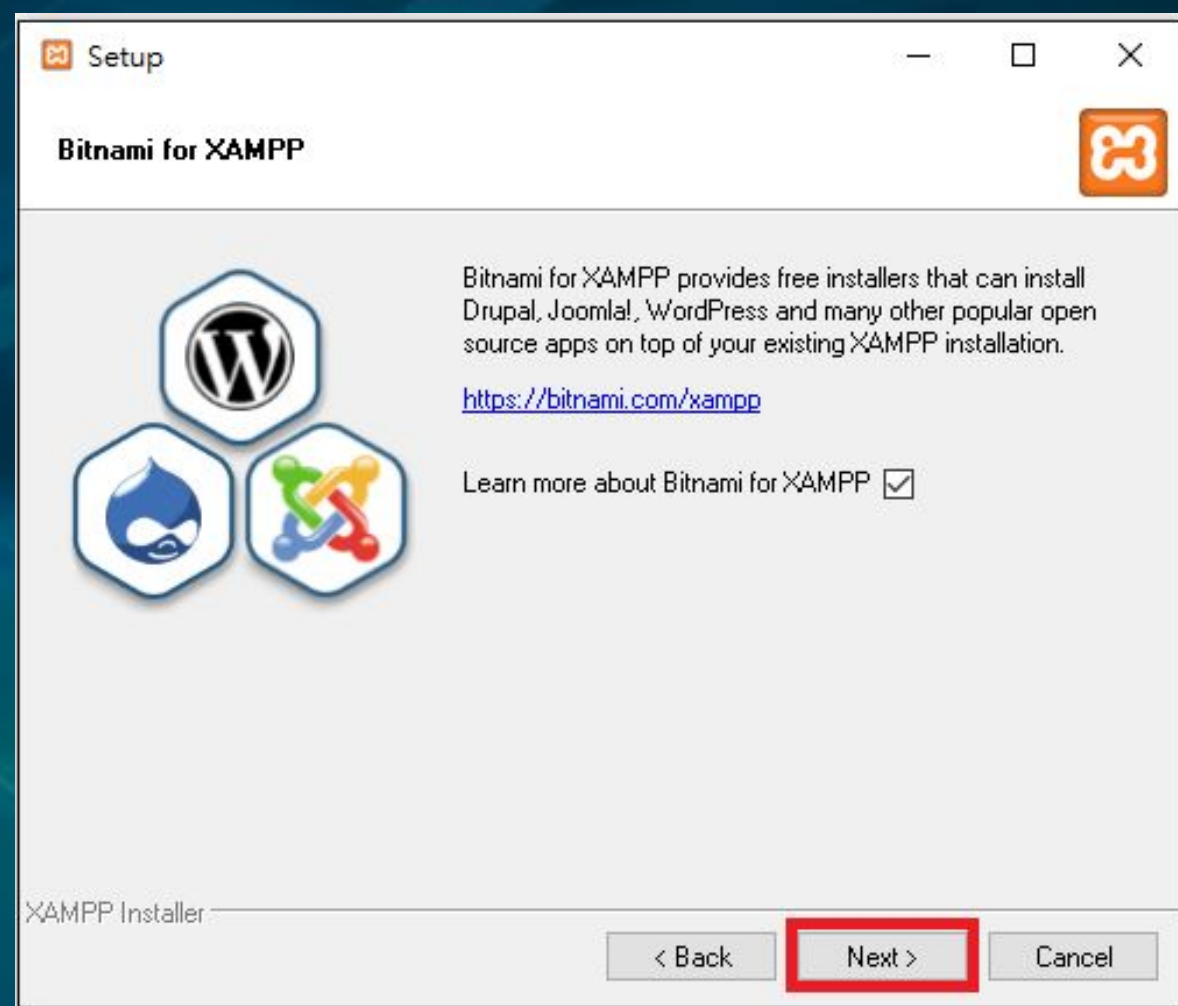
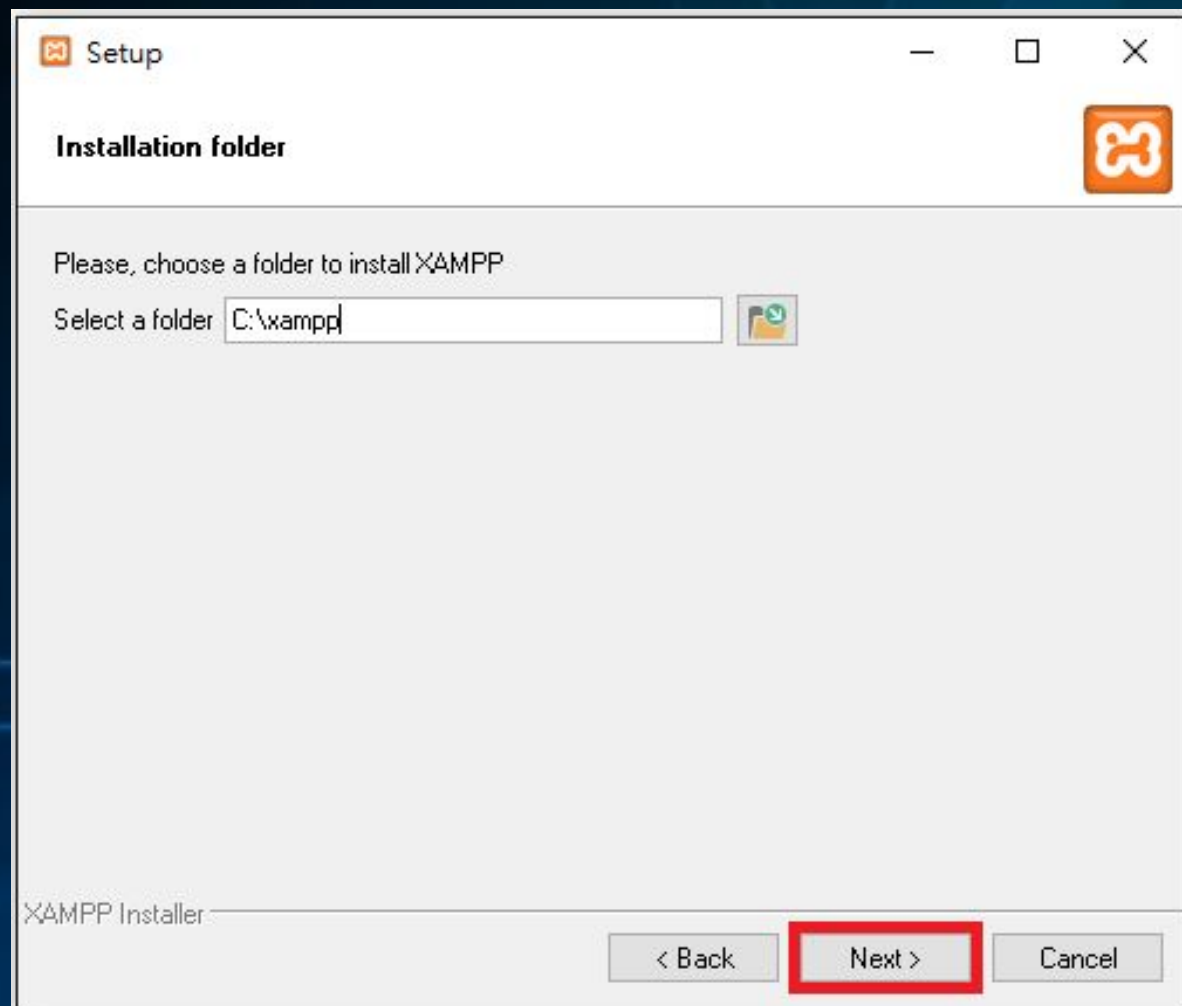
前置安裝(資料庫)

- 安裝



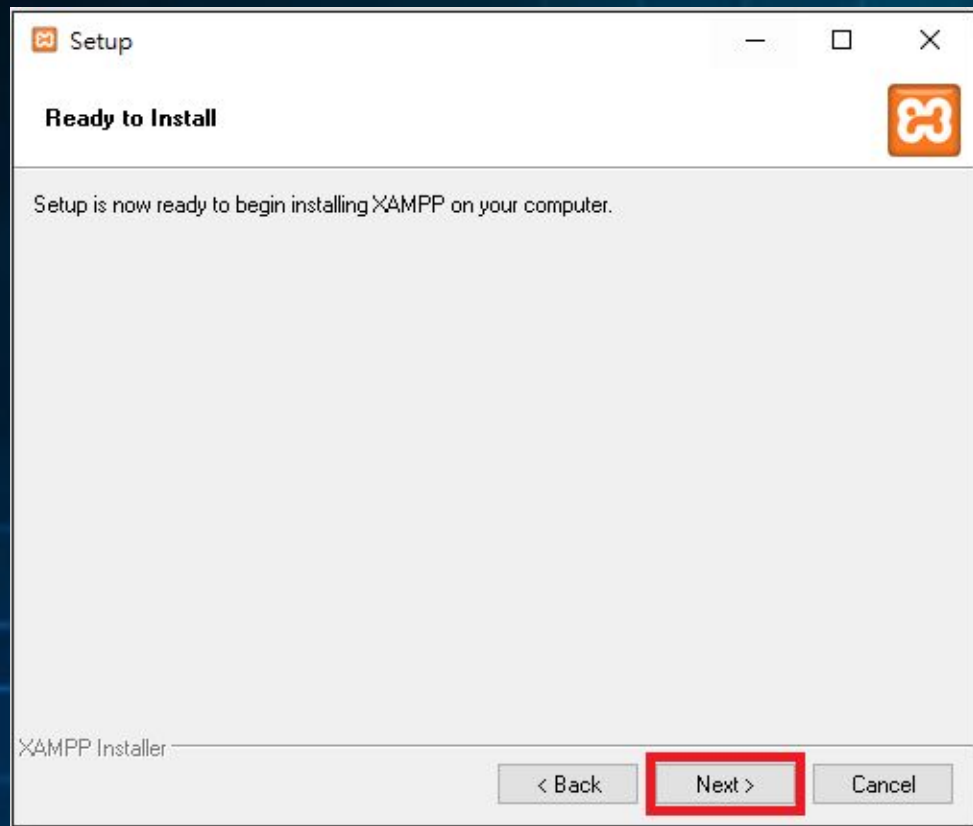
前置安裝(資料庫)

- 安裝









前置安裝(資料庫)

- 安裝



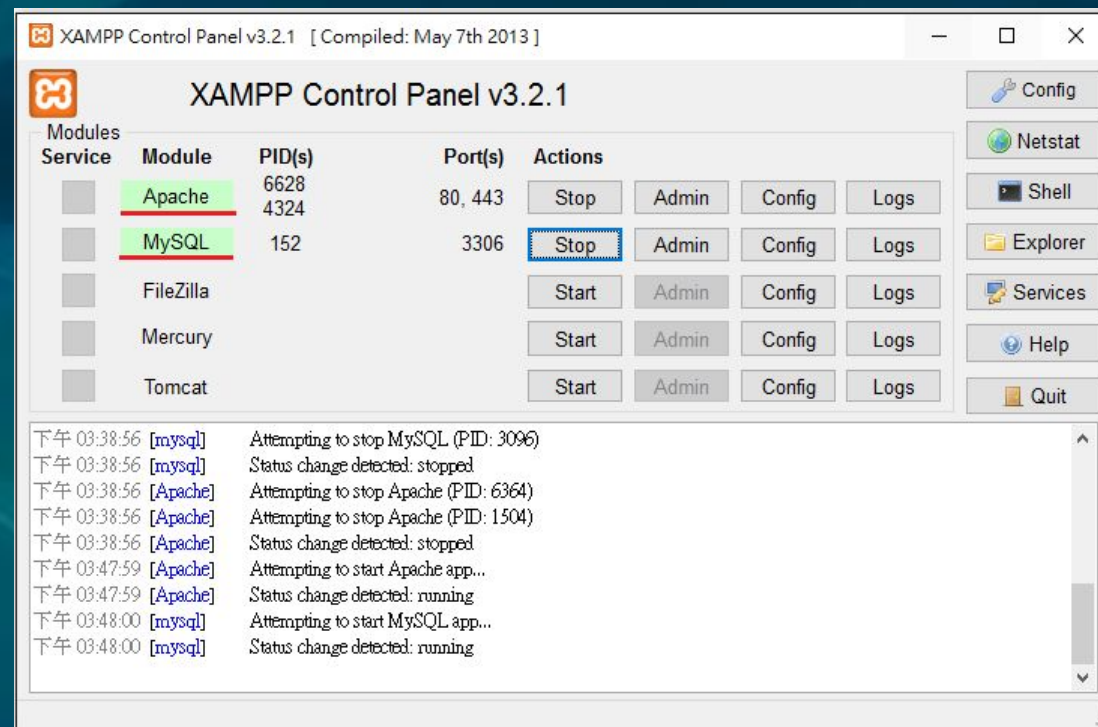
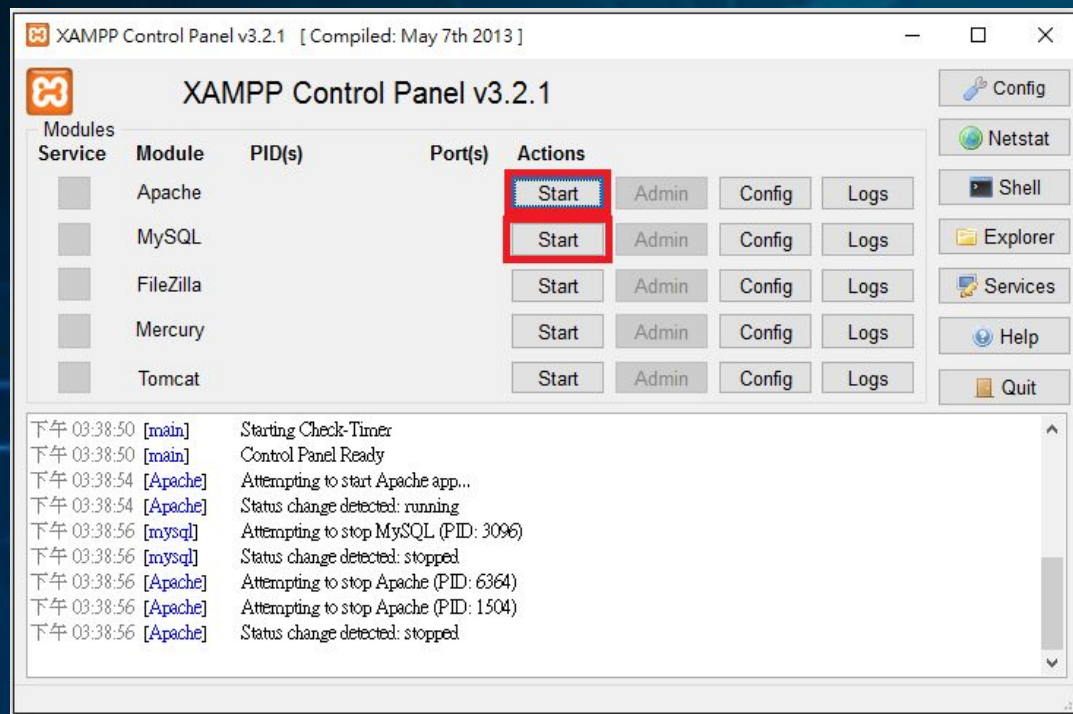
前置安裝(資料庫)

- 開啟C:\xampp (安裝位置) 執行 xampp_control.exe

 uninstall.dat	2015/10/23 下午 ...	DAT 檔案	209 KB
 uninstall.exe	2015/10/23 下午 ...	應用程式	6,844 KB
 xampp_shell.bat	2015/10/23 下午 ...	Windows 批次檔案	2 KB
 xampp_start.exe	2013/3/30 下午 0...	應用程式	116 KB
 xampp_stop.exe	2013/3/30 下午 0...	應用程式	116 KB
 xampp-control.exe	2013/6/17 下午 0...	應用程式	2,509 KB
 xampp-control.ini	2016/1/4 下午 03...	組態設定	2 KB
 xampp-control.log	2016/1/4 下午 03...	文字文件	9 KB

前置安裝(資料庫)

- 將Apache、MySQL開啟 (點擊Start)



前置安裝(資料庫)

- 開啟瀏覽器, 輸入 localhost/dashboard

若看到以下圖示代表環境安裝成功

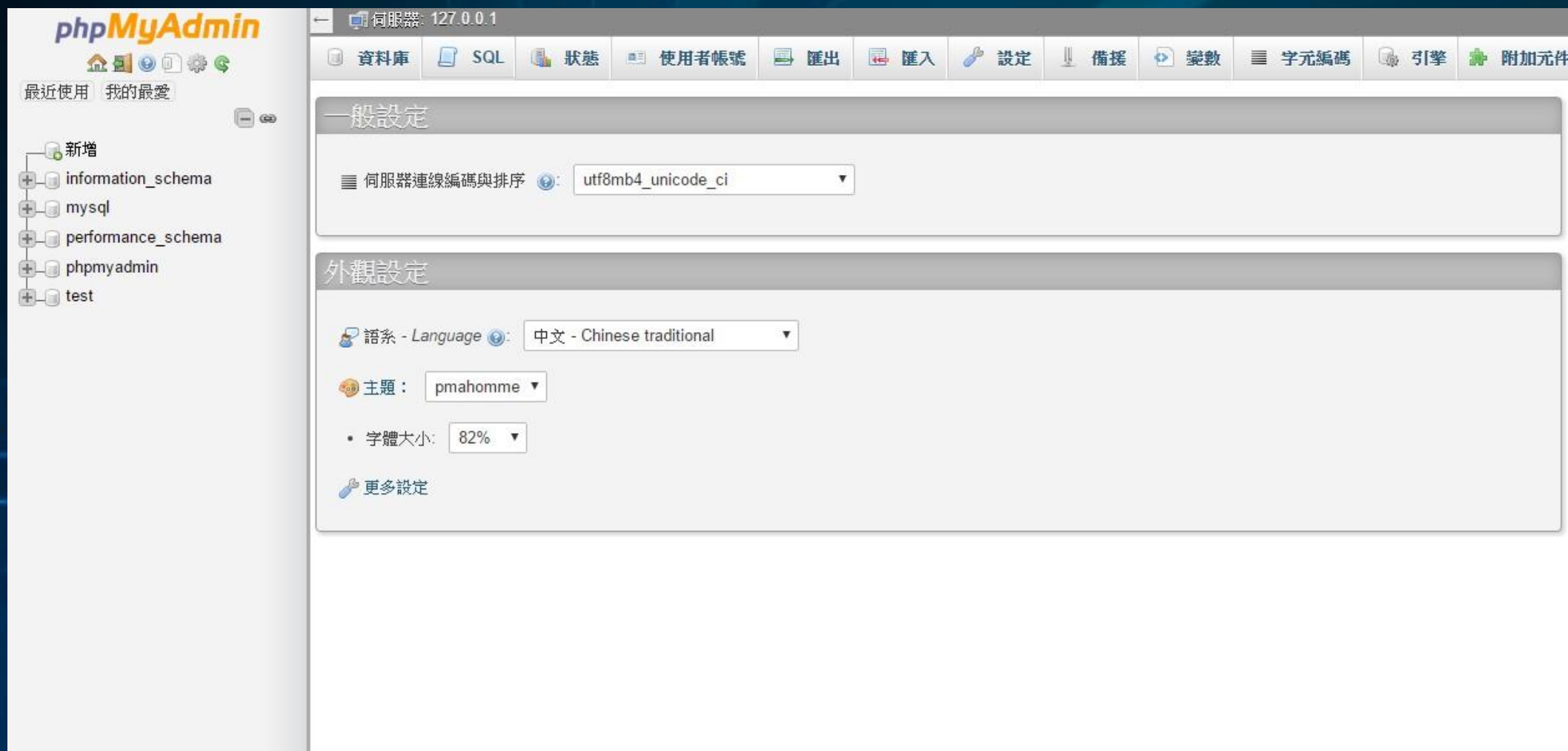


資料庫修正

- 避免日後遇到問題，請先在C:\xampp\mysql內
- 新增資料夾lib
- 進到lib內新增資料夾plugin
- 即存在C:\xampp\mysql\lib\plugin
- 內容為空即可

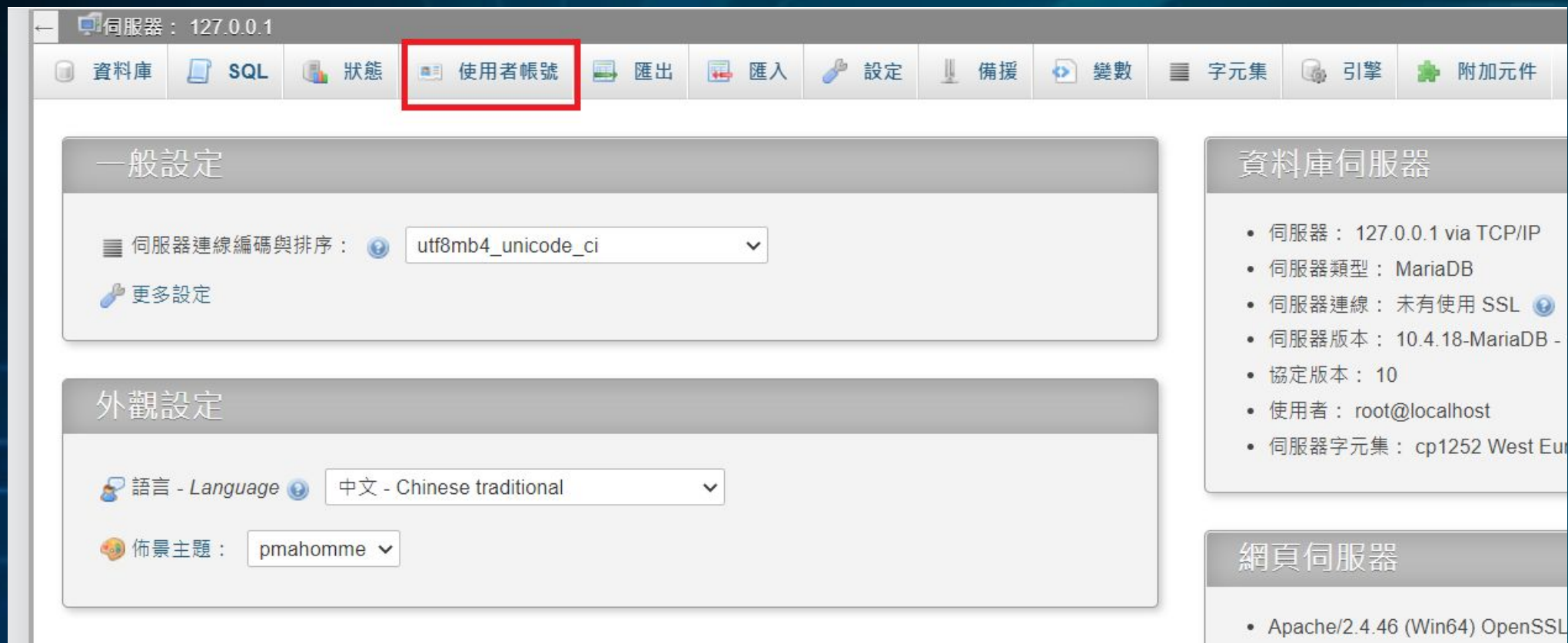
前置安裝(資料庫)

- 打開瀏覽器輸入 localhost/phpmyadmin/ 便可直接連線到資料庫操作頁面。



前置安裝(資料庫)

- 可以新增個人化使用者



前置安裝(資料庫)

- 範例使用 使用者名稱：testt 密碼：test

使用者帳號一覽

	使用者名稱	主機名稱	密碼
<input type="checkbox"/>	任何	%	否
<input type="checkbox"/>	pma	localhost	否
<input type="checkbox"/>	root	127.0.0.1	否
<input type="checkbox"/>	root	::1	否
<input type="checkbox"/>	root	localhost	否
<input type="checkbox"/>	test	%	是

☐ 全選 已選擇項目： 匯出

新增

新增使用者帳號

刪除所選的使用者帳號

新增使用者帳號

登入資訊

使用者名稱：

使用文字方塊 ▼

testt

主機名稱：

任何主機 ▼

%

密碼：

使用文字方塊 ▼

....

重新輸入：

....

認證外掛程式

原生 MySQL 認證 ▼

產生密碼：

產生

建立專案資料夾

- 創建一新資料夾，並進入資料夾

API(Python)

2025/4/7 下午 11:41

檔案資料夾

- 在Anaconda Prompt 底下輸入 Django-admin.py startproject { 專案名稱 }

```
(embedded) C:\Users\user\Desktop\CH06\API(Python)>django-admin startproject api
```

```
(embedded) C:\Users\user\Desktop\CH06\API(Python)>cd api
```

- CD 進入專案資料夾後輸入 manage.py startapp { 應用名稱 }

```
(embedded) C:\Users\user\Desktop\CH06\API(Python)\api>python manage.py startapp app
```


專案資料夾完成預覽

The diagram illustrates the directory structure of a Django project. It starts with a root folder named 'API(Python)'. Inside this folder is a sub-folder named 'api'. The 'api' folder contains two items: another folder named 'app' and a file named 'manage.py'. A red arrow points from the 'api' folder to a detailed view of its contents, and a green arrow points from the 'app' folder to a detailed view of its contents.

API(Python) Directory Structure:

名稱	修改日期	類型	大小
api	2025/4/7 下午 11:42	檔案資料夾	
app	2025/4/7 下午 11:42	檔案資料夾	
manage.py	2025/4/7 下午 11:41	PY 檔案	1 KB

api Folder Contents:

名稱	修改日期	類型	大小
pycache	2025/4/7 下午 11:42	檔案資料夾	
init.py	2025/4/7 下午 11:41	PY 檔案	0 KB
asgi.py	2025/4/7 下午 11:41	PY 檔案	1 KB
settings.py	2025/4/7 下午 11:41	PY 檔案	4 KB
urls.py	2025/4/7 下午 11:41	PY 檔案	1 KB
wsgi.py	2025/4/7 下午 11:41	PY 檔案	1 KB

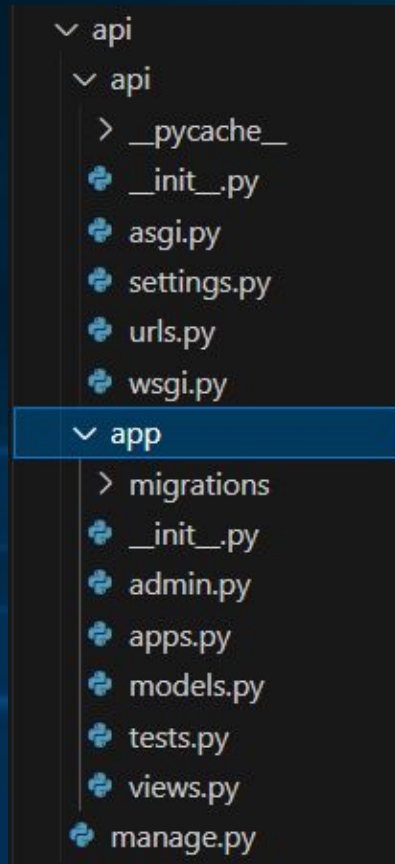
app Folder Contents:

名稱	修改日期	類型	大小
migrations	2025/4/7 下午 11:42	檔案資料夾	
init.py	2025/4/7 下午 11:42	PY 檔案	0 KB
admin.py	2025/4/7 下午 11:42	PY 檔案	1 KB
apps.py	2025/4/7 下午 11:42	PY 檔案	1 KB
models.py	2025/4/7 下午 11:42	PY 檔案	1 KB
tests.py	2025/4/7 下午 11:42	PY 檔案	1 KB
views.py	2025/4/7 下午 11:42	PY 檔案	1 KB

views.py Details:

- 類型: PY 檔案
- 大小: 60 個位元組
- 修改日期: 2025/4/7 下午 11:42

各檔案功能

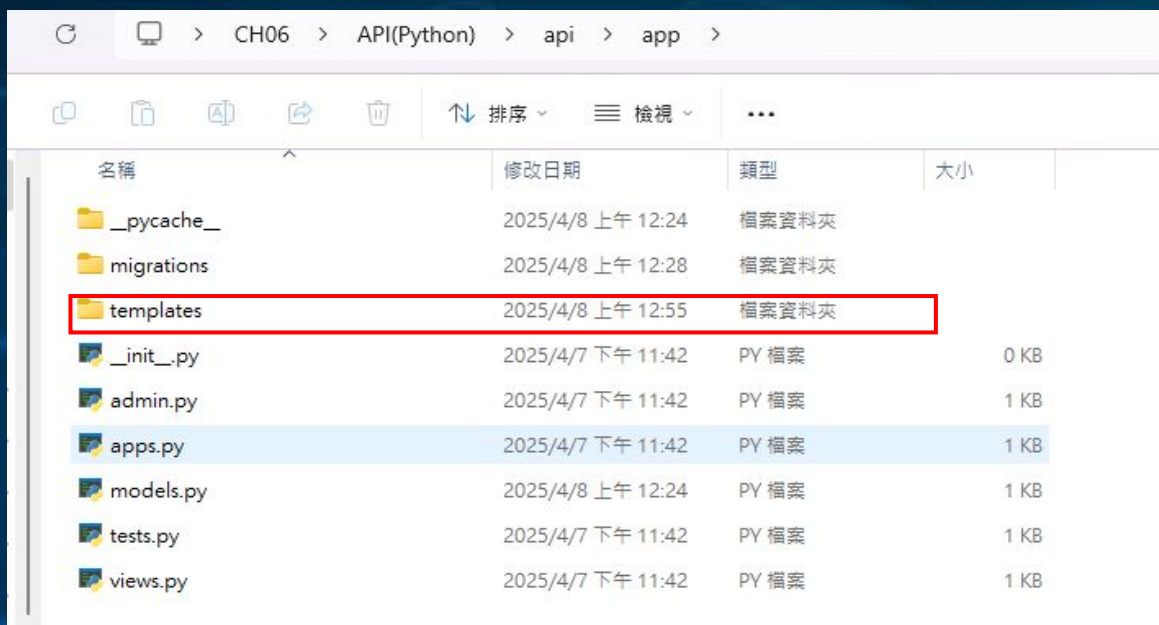


settings.py—Django設定部分
url.py--負責處理路由
model.py—負責資料庫對接
views.py—負責api功能編寫
其餘—用不到

Django編寫邏輯:
進行初始設定(setting.py)->api功能編寫(views.py)->為
該功能添加路由(urls.py)

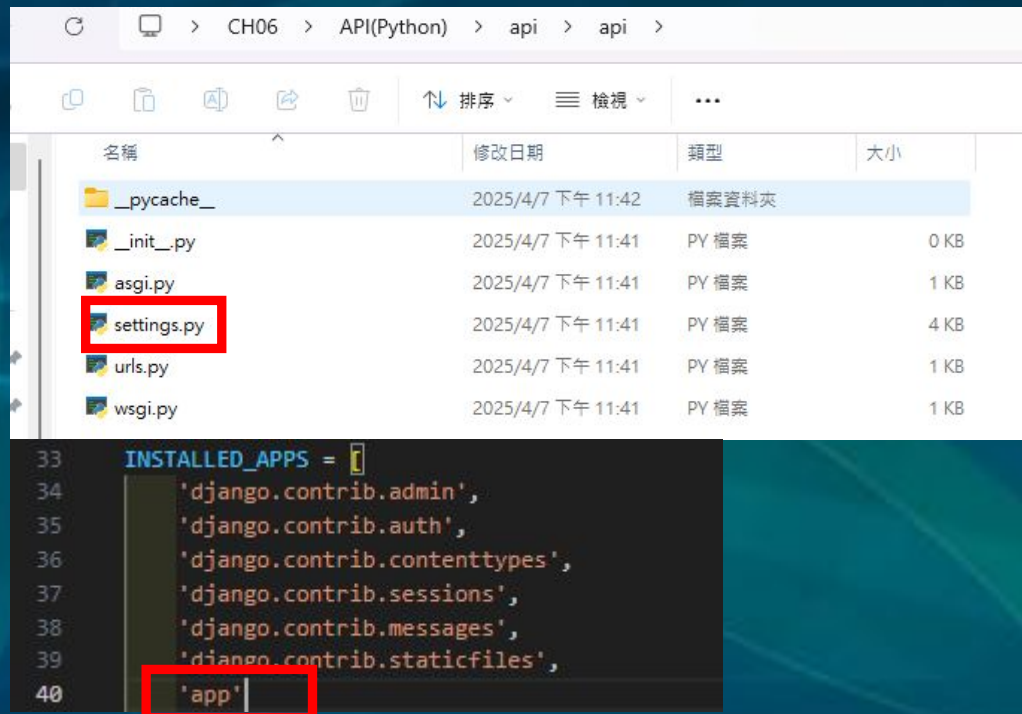
Templates模板

- Templates為Django內建的一種設計 HTML 頁面的系統，能將後端資料動態渲染到前端頁面中，本次api的前端將使用Templates模板進行編寫(可更換成自己熟悉的前端語言or架構)
- 在{應用名稱}資料夾裡創建Templates資料夾



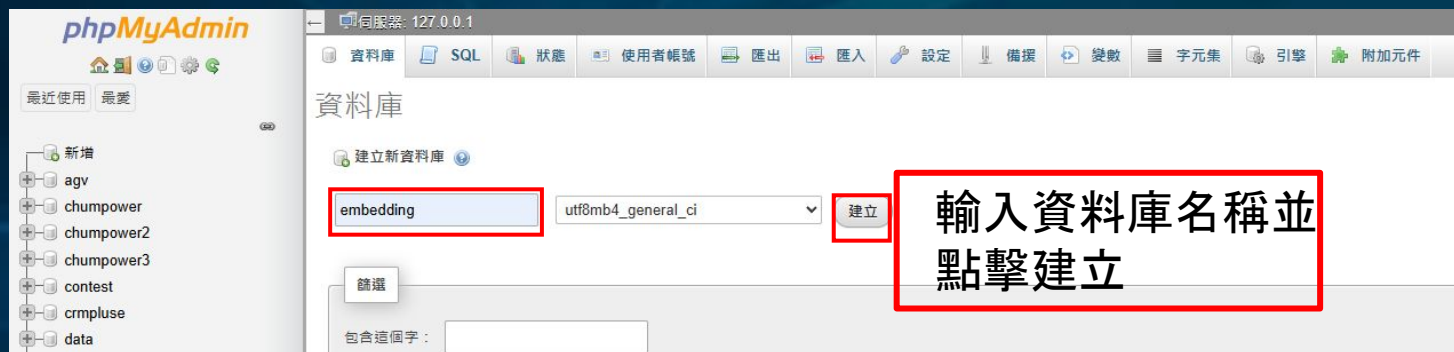
將新增的 Django app 加入設定檔

- 將新建立的{ 應用名稱 }加入功能設定檔
- 打開 { 專案名稱 } / { 專案名稱 } /setting.py 找到 INSTALLED_APPS
- 新增 '{ 應用名稱 }'



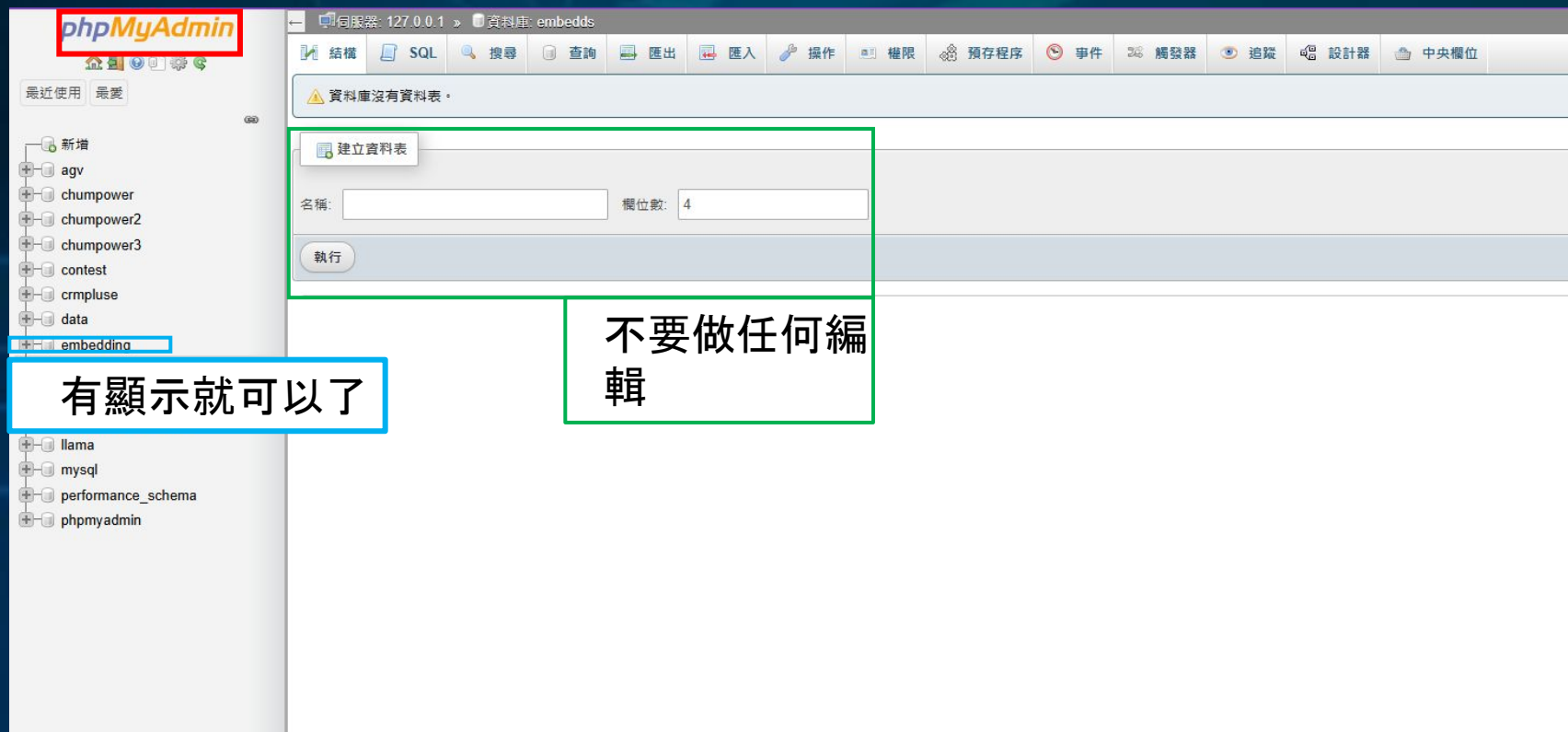
新建資料庫 1

- 啟動xampp
- 瀏覽器進入127.0.0.1或localhost



新建資料庫 2

- 無須編輯,點左上角Logo回主頁即可



Django連接MySQL設定

- 打開setting.py找到下方圖示部分

```
76 DATABASES = {  
77     'default': {  
78         'ENGINE': 'django.db.backends.sqlite3',  
79         'NAME': BASE_DIR / 'db.sqlite3',  
80     }  
81 }  
82
```

- 更改為

```
80 DATABASES = {  
81     'default': {  
82         'ENGINE': 'django.db.backends.mysql',  
83         'NAME': 'embedding',          # 資料庫名稱  
84         'USER': 'user',               # 使用者名稱 (根據你的設定)  
85         'PASSWORD': 'user',          # 密碼  
86         'HOST': 'localhost',          # 或 127.0.0.1  
87         'PORT': '3306',               # MySQL 預設埠號  
88     }  
89 }  
90
```

Django API — 建立資料表 (以溫溼度為例)

- 開啟 { 應用名稱 }/models.py 新增一段【資料表結構】

```
api > app > models.py > ...
1  from django.db import models
2
3  class SensorData(models.Model):
4      temperature = models.FloatField()      # 溫度 (°C)
5      humidity = models.FloatField()         # 濕度 (%)
6      recorded_at = models.DateTimeField(auto_now_add=True) # 自動紀錄時間
7
8      Tabnine | Edit | Test | Explain | Document
9      def __str__(self):
10         return f"{self.recorded_at} | 溫度: {self.temperature}°C | 濕度: {self.humidity}%"
```

- 完成後執行

```
python manage.py makemigrations
```

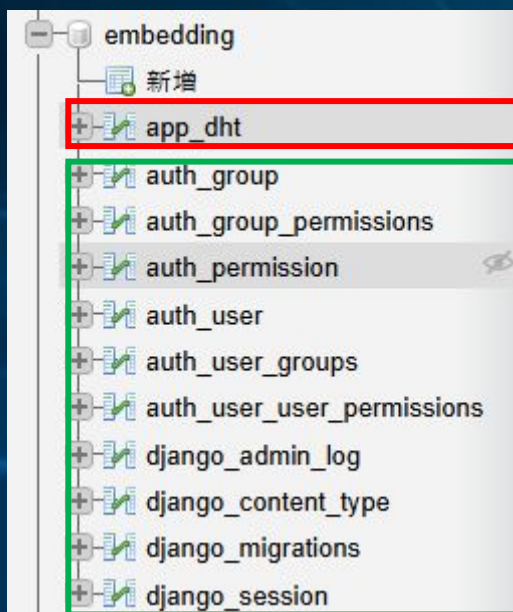
檢查資料庫有無變動

```
python manage.py migrate
```

應用所有變動

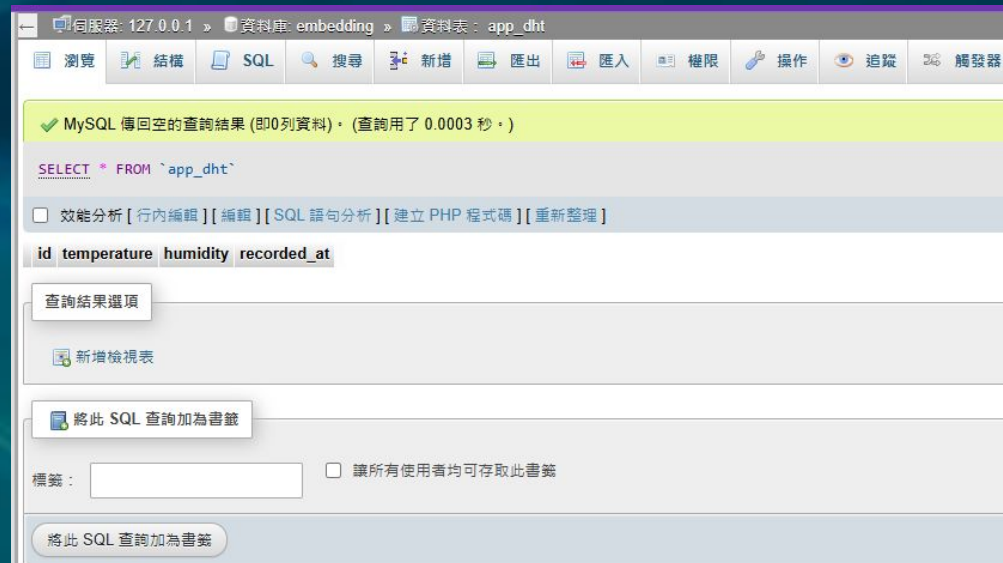
確認資料表是否建立

- 啟動xampp
- 瀏覽器進入127.0.0.1或localhost



目標資料表

Restful api
相關資料表
(不重要)



Django API 功能 — 資料上傳

- 開啟 { 應用名稱 }/views.py 新增一段【動作/副函式/API】
- 利用POST接收參數

```
api > app > views.py > ...
1  from django.shortcuts import render, redirect
2  from .models import dht #從models.py載入需要的資料表
3
4  # 上傳資料
   Tabnine | Edit | Test | Explain | Document
5  def sensor_upload(request):
6      if request.method == 'POST':
7          temp = request.POST.get('temperature')
8          hum = request.POST.get('humidity')
9          dht.objects.create(temperature=temp, humidity=hum)
10         return redirect('sensor_list')
11     return render(request, 'sensor_upload.html')
12
```


Django API 功能 一路由設定

- 開啟{ 專案名稱 }/urls.py 新增API功能路由
- 每個api功能都要有一個路由來讓前端進行使用

```
17 from django.contrib import admin
18 from django.urls import path
19 from app import views
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('sensor/upload/', views.sensor_upload, name='sensor_upload'),
24     path('sensor/list/', views.sensor_data, name='sensor_list'),
25 ]
26
```

從views.py載入所有api功能

Django API 功能 — 資料查詢(下載)

- 開啟{ 應用名稱 }/views.py 新增一段【動作/副函式/API】
- 利用POST帶入參數, 查詢特定值或範圍資料

```
13  # 查詢資料
    Tabnine | Edit | Test | Explain | Document
14  def sensor_data(request):
15      data = dht.objects.all().order_by('-recorded_at')
16      return render(request, 'sensor_data.html', {'data': data})
17  |
```

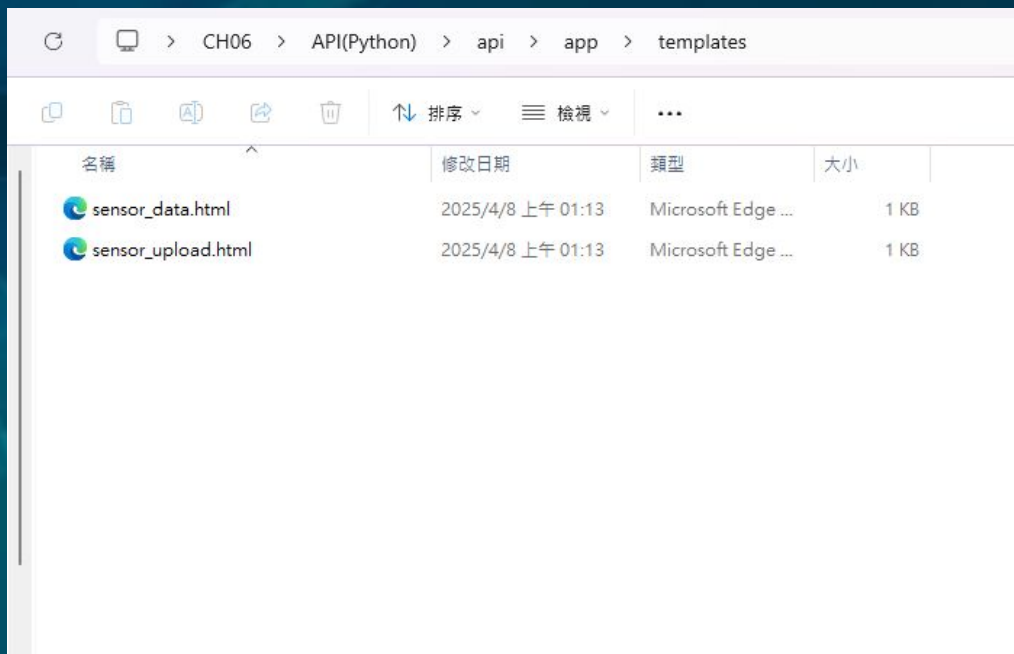
Django API 功能 一路由設定

- 開啟{ 專案名稱 }/urls.py 新增API功能路由

```
17 from django.contrib import admin
18 from django.urls import path
19 from app import views
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('sensor/upload/', views.sensor_upload, name='sensor_upload'),
24     path('sensor/data/', views.sensor_data, name='sensor_data'),
25 ]
26
```

前端網頁

- 在 {應用名稱}/templates/ 資料夾裡, 建立兩個 HTML 檔案
 - sensor_upload.html
 - sensor_data.html



sensor_upload.html程式碼

api > app > templates > sensor_data.html > html > body > table > tr

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>感測資料紀錄</title>
5  </head>
6  <body>
7  <h2>感測資料列表</h2>
8  <table border="1">
9  <tr>
10 <th>時間</th>
11 <th>溫度 (°C)</th>
12 <th>濕度 (%)</th>
13 </tr>
14 {% for item in data %}
15 <tr>
16 <td>{{ item.recorded_at }}</td>
17 <td>{{ item.temperature }}</td>
18 <td>{{ item.humidity }}</td>
19 </tr>
20 {% endfor %}
21 </table>
22 <a href="/sensor/upload/">返回上傳表單</a>
23 </body>
24 </html>
25
```


sensor_data.html程式碼

```
api > app > templates > <> sensor_upload.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>上傳感測資料</title>
5  </head>
6  <body>
7  <h2>輸入感測資料</h2>
8  <form method="POST">
9  <div>{% csrf_token %} 溫度 (°C):<input
10 <div>type="number"
11 <div>name="temperature"
12 <div>step="0.1"
13 </div></div><br />
14 濕度 (%):<input type="number" name="humidity" step="0.1" /><br />
15 <button type="submit">上傳</button>
16 </form>
17 <a href="/sensor/list/">查看所有資料</a>
18 </body>
19 </html>
20
```

Django

- 找到manage.py 輸入python manage.py runserver 127.0.0.1:8000
- 執行server查看結果

```
(embedded) C:\Users\user\Desktop\CH06\API(Python)\api>python manage.py runserver 127.0.0.1:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 08, 2025 - 01:30:28
Django version 3.2, using settings 'api.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
|
```

完成效果預覽

- 瀏覽器輸入127.0.0.1:8000會得到以下頁面
- 輸入http://127.0.0.1:8000/sensor/upload/進入資料上傳頁面
- 輸入http://127.0.0.1:8000/sensor/data/進入資料查詢頁面

Page not found (404)

Request Method: GET

Request URL: http://127.0.0.1:8000/

Using the URLconf defined in api.urls, Django tried these URL patterns in this order:

1. admin/
2. sensor/upload/ [name='sensor_upload']
3. sensor/data/ [name='sensor_data']

The empty path didn't match any of these.

這邊為目前所有可用路由

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

完成效果預覽

- 此為sensor_upload的測試與結果

輸入感測資料

溫度 (°C): 25

濕度 (%): 60

上傳

[查看所有資料](#)

- Prompt出現

```
[08/Apr/2025 01:40:50] "POST /sensor/upload/ HTTP/1.1" 302 0
```

伺服器: 127.0.0.1 > 資料庫: embedding > 資料庫: app_dht

瀏覽 結構 SQL 搜尋 新增 匯出 匯入 權限 操作 追蹤 觸發器

✓ 顯示第 0 - 0 列 (總計 1 筆, 查詢用了 0.0004 秒)

SELECT * FROM `app_dht`

☐ 效能分析 [行內編輯] [編輯] [SQL 語句分析] [建立 PHP 程式碼] [重新整理]

☐ 全部顯示 | 資料列數: 25 | 篩選資料列: 搜尋此資料表

+ 選項

	id	temperature	humidity	recorded_at
<input type="checkbox"/> 編輯 <input type="checkbox"/> 複製 <input type="checkbox"/> 刪除	1	25	60	2025-04-07 17:40:50.908053

↑ ☐ 全選 已選擇項目: ☐ 編輯 ☐ 複製 ☐ 刪除 ☐ 匯出

☐ 全部顯示 | 資料列數: 25 | 篩選資料列: 搜尋此資料表

查詢結果選項

☐ 列印 ☐ 複製到剪貼簿 ☐ 匯出 ☐ 顯示圖表 ☐ 新增檢視表

☐ 將此 SQL 查詢加為書籤

標籤: ☐ 讓所有使用者均可存取此書籤

完成效果預覽

- 此為sensor_data的測試與結果

感測資料列表

時間	溫度 (°C)	濕度 (%)
April 7, 2025, 5:40 p.m.	25.0	60.0

[返回上傳表單](#)

Django API 功能2 — 事前設定 1

- 開啟setting.py, 設定路徑變數

```
import os

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

- 程式碼預覽

```
13     from pathlib import Path
14     import os
15
16
17     # Build paths inside the project like this: BASE_DIR / 'subdir'.
18     BASE_DIR = Path(__file__).resolve().parent.parent
19
20     MEDIA_URL = '/media/'
21     MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
22
```

Django API 功能2 — 事前設定2

- 開啟url.py 加入

```
from django.conf import settings
from django.conf.urls.static import static
```

```
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

加在程式碼最下方

- 程式碼預覽

```
17 from django.contrib import admin
18 from django.urls import path
19 from app import views
20 from django.conf import settings
21 from django.conf.urls.static import static
22
23 urlpatterns = [
24     path('admin/', admin.site.urls),
25     path('sensor/upload/', views.sensor_upload, name='sensor_upload'),
26     path('sensor/data/', views.sensor_data, name='sensor_data'),
27 ]
28
29 urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
30
```

Django API 功能2 — 圖片上傳

- 開啟{ 應用名稱 }/views.py 新增一段【動作/副函式/API】

```
1 from django.shortcuts import render, redirect
2 from .models import dht #從models.py載入需要的資料表
3 from django.conf import settings
4 from django.http import HttpResponseRedirect, Http404
5 import os
```

```
21 # 圖片上傳
22 def image_upload(request):
23     if not os.path.exists(settings.MEDIA_ROOT):
24         os.makedirs(settings.MEDIA_ROOT)
25     message = ''
26     if request.method == 'POST' and request.FILES.get('image'):
27         image = request.FILES['image']
28         file_path = os.path.join(settings.MEDIA_ROOT, image.name)
29         with open(file_path, 'wb+') as destination:
30             for chunk in image.chunks():
31                 destination.write(chunk)
32         message = '圖片已成功上傳！'
33
34     # 列出所有圖片檔案
35     image_files = os.listdir(settings.MEDIA_ROOT) if os.path.exists(settings.MEDIA_ROOT) else []
36     return render(request, 'image_upload.html', {'files': image_files, 'message': message})
37
```

若找不到media資料夾則自動創建

Django API 功能2 –路由設定

```
17 from django.contrib import admin
18 from django.urls import path
19 from app import views
20 from django.conf import settings
21 from django.conf.urls.static import static
22
23 urlpatterns = [
24     path('admin/', admin.site.urls),
25     path('sensor/upload/', views.sensor_upload, name='sensor_upload'),
26     path('sensor/data/', views.sensor_data, name='sensor_data'),
27     path('image/upload/', views.image_upload, name='image_upload'),
28     path('image/download/<str:filename>/', views.image_download, name='image_download'),
29 ]
30
31 urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```


Django API 功能2 — 圖片下載

- 開啟{ 應用名稱 }/views.py 新增一段【動作/副函式/API】

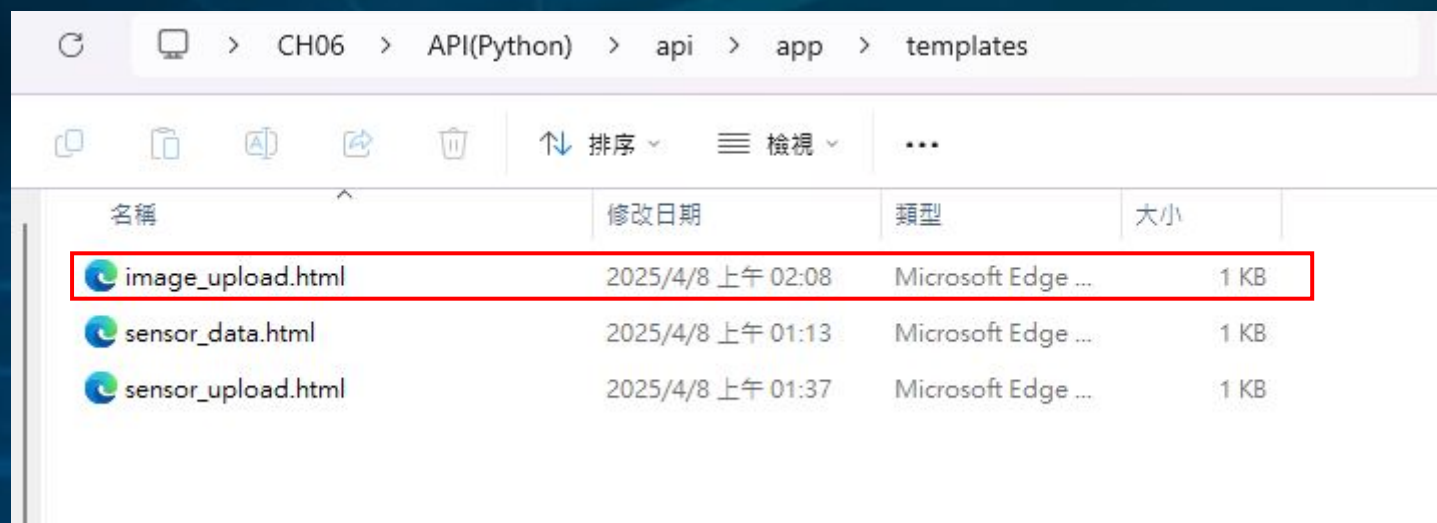
```
37 # 圖片下載
38 def image_download(request, filename):
39     file_path = os.path.join(settings.MEDIA_ROOT, filename)
40     if os.path.exists(file_path):
41         return FileResponse(open(file_path, 'rb'), as_attachment=True)
42     raise Http404("圖片不存在")
43
```


Django API 功能2 –路由設定

```
17 from django.contrib import admin
18 from django.urls import path
19 from app import views
20 from django.conf import settings
21 from django.conf.urls.static import static
22
23 urlpatterns = [
24     path('admin/', admin.site.urls),
25     path('sensor/upload/', views.sensor_upload, name='sensor_upload'),
26     path('sensor/data/', views.sensor_data, name='sensor_data'),
27     path('image/upload/', views.image_upload, name='image_upload'),
28     path('image/download/<str:filename>/', views.image_download, name='image_download'),
29 ]
30
31 urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

前端網頁

- 在 {應用名稱}/templates/ 資料夾裡, 建立 HTML 檔案
 - image_upload.html



image_upload.html程式碼

api > app > templates > image_upload.html > html > body > ul > li > a

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>圖片上傳與下載</title>
5    </head>
6    <body>
7      <h2>上傳圖片</h2>
8      <form method="POST" enctype="multipart/form-data">
9        {% csrf_token %}
10       <input type="file" name="image" accept="image/*" />
11       <button type="submit">上傳</button>
12     </form>
13
14     <p style="color: green">{{ message }}</p>
15
16     <h3>已上傳圖片列表</h3>
17     <ul>
18       {% for file in files %}
19       <li>
20         {{ file }} -
21         
22         <a href="/media/{{ file }}" target="_blank">查看</a>
23         <a href="{% url 'image_download' file %}">下載</a>
24       </li>
25       {% empty %}
26       <li>沒有上傳的圖片。</li>
27     {% endfor %}
28   </ul>
29 </body>
30 </html>
31
```

Django

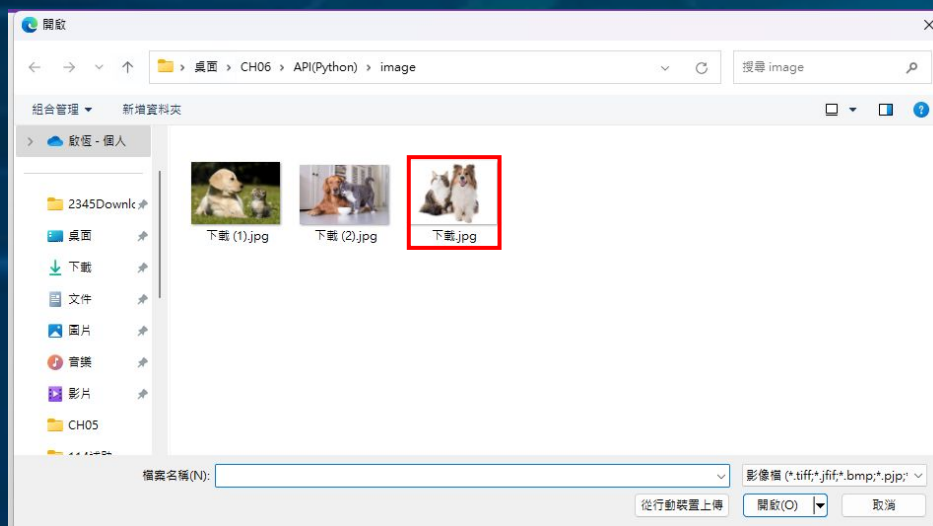
- 找到manage.py 輸入python manage.py runserver 127.0.0.1:8000
- 執行server查看結果

```
(embedded) C:\Users\user\Desktop\CH06\API(Python)\api>python manage.py runserver 127.0.0.1:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 08, 2025 - 01:30:28
Django version 3.2, using settings 'api.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
|
```

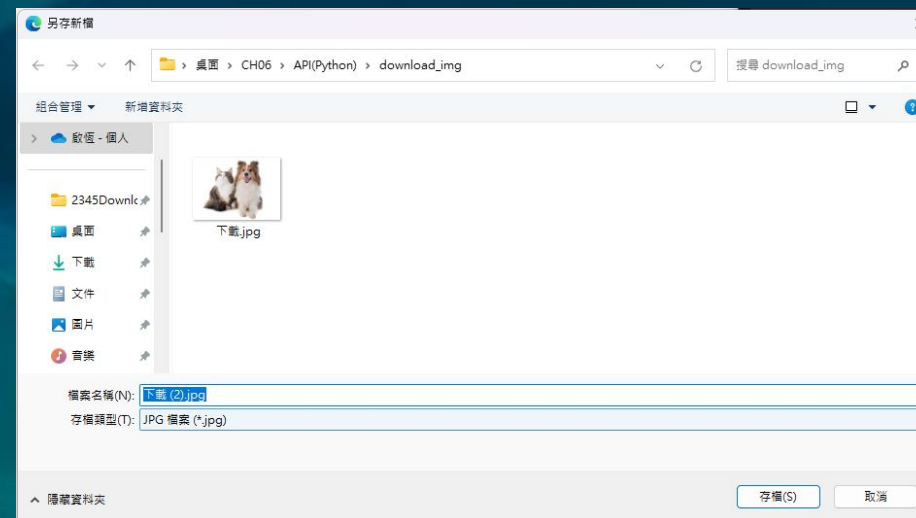
完成效果預覽

- 此為image_upload的測試與結果

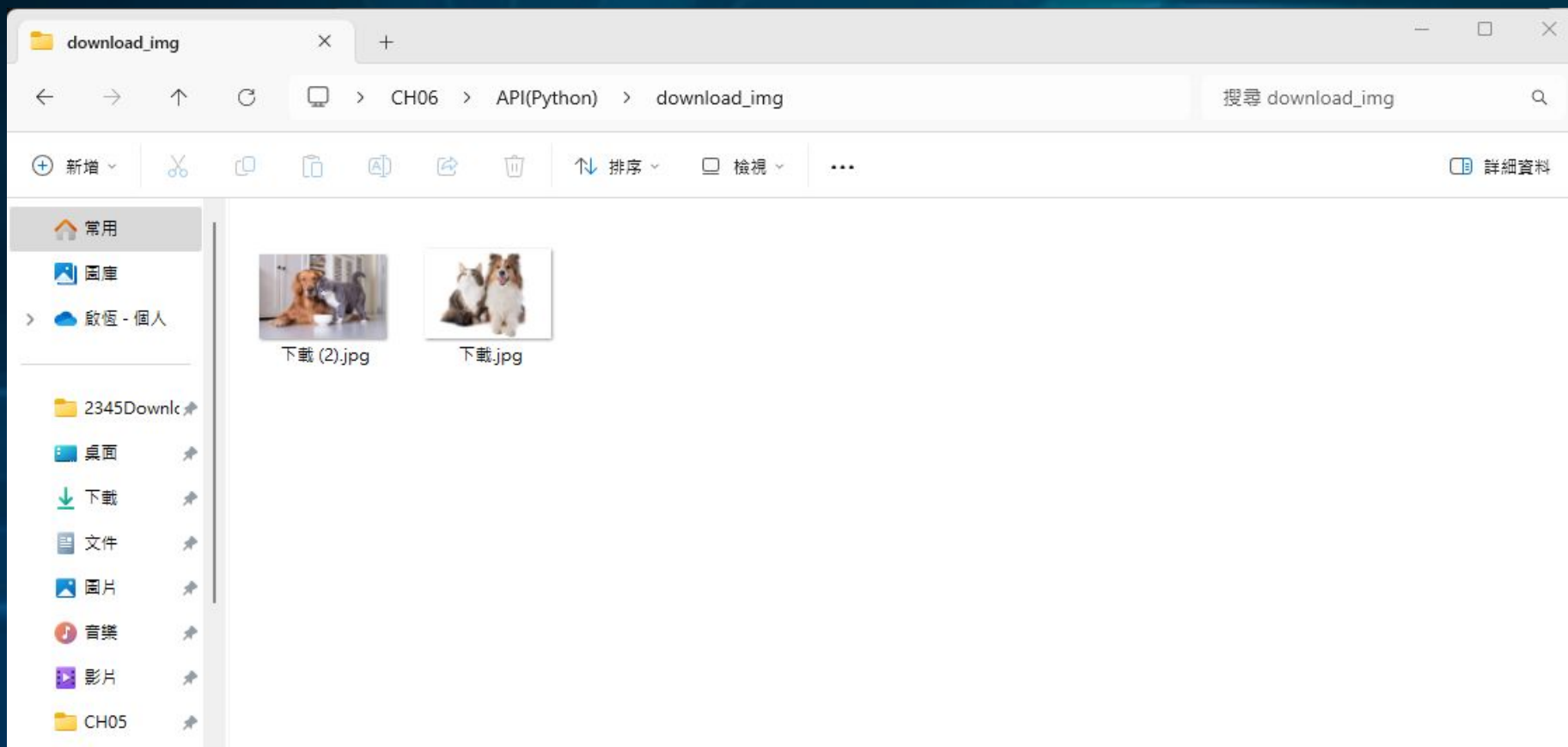


完成效果預覽

- 此為image_download的測試與結果
- 下載圖片至指定路徑or資料夾



完成效果預覽



Django

- 恭喜, 到此步驟所有API已經撰寫完成

The background is a deep blue gradient. It features several bright, horizontal cyan lines that appear to be glowing or laser-like. Overlaid on these are faint, translucent, light blue shapes that resemble stylized petals or leaves, arranged in a circular pattern around the center. The overall effect is futuristic and high-tech.

THANK YOU