

# 入門 Django 並打造具 AI 服務的網站

## 實戰工作坊



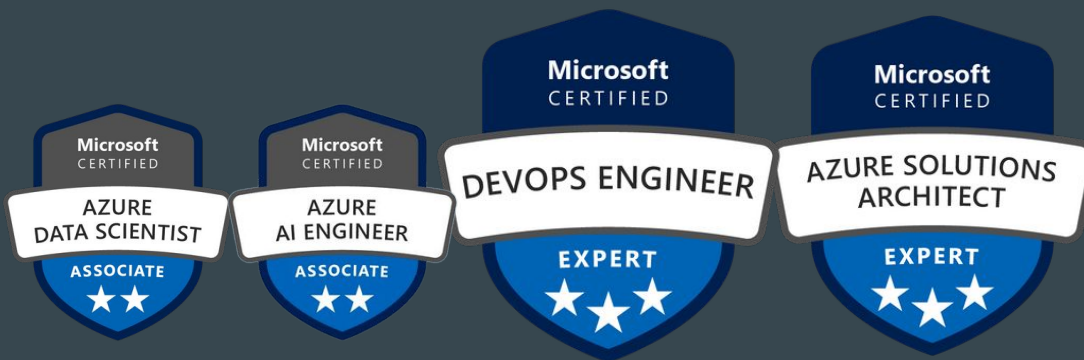
Ko Ko, Microsoft AI MVP

2023/11/08 @Modern Web Conference 2023 @Popop Taipei

# 關於 Ko Ko



- 連續四年當選 Microsoft AI MVP
- 國內外大型年會講者, 包含 COSCUP、DevDays Asia、PyCon APAC、PyCon HK、名古屋開源年會、香港開源年會等等。
- 合著有《駕馭 ChatGPT 4: 探索 Azure OpenAI 與 Cognitive Service for Language 開發實踐 (使用 .NET 與 Node. Js)》。
- 經營粉絲專頁「大魔術熊貓工程師」。
- <https://linktr.ee/qgkerk>



# 實戰工作坊大綱

- Django 基本介紹
- 實作(建一個小熊故事機)
- 虛擬環境架設
- Django 專案建立
- MTV 架構與路由
- ORM 與資料庫
- Admin 後台
- 串接 ChatGPT



# Django 的歷史

2003 年，報社 Lawrence Journal-World 的軟體工程師 Adrian Holovaty 和 Simon Willison 開始著手開發。

以前叫 The CMS，後來 spin off 成 Ellington CMS。

2005 年 7 月改名為 Django，被以 BSD 授權條款釋出。

以知名爵士吉他手 Django Reinhardt 的名字命名。

The Django logo, featuring the word "django" in a white, lowercase, sans-serif font, set against a dark green rectangular background.

# 使用 Django 框架來開發

**快速開發：**Django 提供了許多現成的功能和工具，例如 ORM、表單處理、驗證和安全性等，這樣可以大幅度縮短開發時間，從而提高開發效率。

**簡單易用：**Django 框架使用的 Python 是一種簡單易學的程式語言，且 Django 框架的文檔完善，為開發人員提供了良好的學習資源和指導。

**安全性高：**Django 框架對安全性非常重視，預設提供了一些安全機制，例如防止跨站腳本攻擊(XSS)和跨站請求偽造(CSRF)攻擊等，並且可以方便地定制和擴展。

**可擴展性強：**Django 支援插件機制，例如 Django REST framework、Django CMS 等，這些插件可以輕鬆地實現更多功能和特性，從而使應用更具有可擴展性。

**社群活躍：**Django 框架的社群相當活躍，這些開發人員貢獻了大量的程式碼、文檔和工具，同時也提供了豐富的學習資源，可以讓開發人員更好地理解和使用 Django。

# 許多世界級大廠都用 Django



# Django MTV 架構說明

**View:** 視圖是一個請求處理函數，它接收 HTTP request 並返回 HTTP response。視圖通過模型訪問滿足請求所需的數據，並將回應的內容委託給模板來呈現。

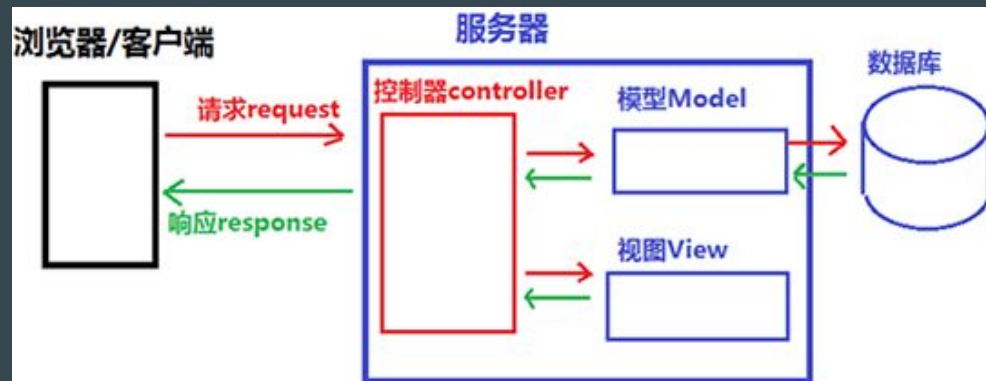
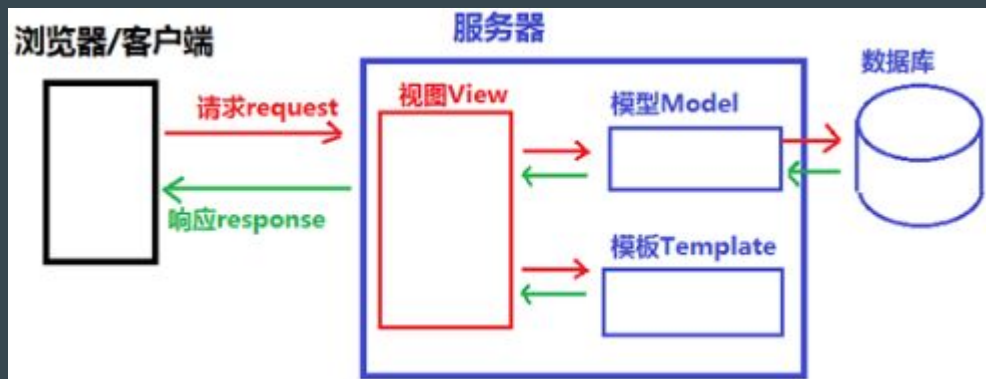
**Models:** 模型是定義應用程式資料結構的 Python 物件，並提供在資料庫中做 CRUD 的機制。

**Templates:** 模板是定義文件（例如 HTML 頁面）的結構或佈局的文本文件。一個視圖可以使用 HTML 模板，從數據填充它動態地創建一個 HTML 頁面模型。可以使用模板來定義任何類型的文件的結構，常見的為 Django Template 和 Jinja2。

# MTV 和 MVC 架構的差異


工作任務	MTV 架構	MVC 架構
資料庫存取	Model	Model
顯示資料給使用者看	Template	View
控制器	View	Controller






實作開始

# 先打星星

 **mwc2023\_django\_with\_aoai** Public

Pin Unwatch 1


main 1 branch 0 tags Go to file Add file <> Code

 **Ko-Ko-Kirk** first commit

e5c9d9e 1 minute ago 1 commit

chatgpt_project	first commit	1 minute ago
first_app	first commit	1 minute ago
.env	first commit	1 minute ago
.gitignore	first commit	1 minute ago
README.md	first commit	1 minute ago
manage.py	first commit	1 minute ago

README.md



## mwc2023\_django\_with\_aoai

This is a workshop for Django with Azure Open AI and Azure Speech in Modern Web Conference 2023.

You can follow the PDF file in this repository to build this project step by step.

[https://github.com/Ko-Ko-Kirk/mwc2023\\_django\\_with\\_aoai](https://github.com/Ko-Ko-Kirk/mwc2023_django_with_aoai)



# 用 VS CODE 來開發 Python

免費且開源：Visual Studio Code (VS Code) 是一個免費、開源的編輯器，由 Microsoft 開發並支持，具有強大的社群支援和持續更新。

跨平台：VS Code 可以在 Windows、macOS 和 Linux 上運行，使得跨平台開發變得無縫。

豐富的插件生態系統：VS Code 擁有龐大的插件市場，你可以輕鬆地安裝 Python 開發所需的擴展，如 Python 擴展包、Linter、Debugger 等。

內置終端機：VS Code 內建終端機可以讓你直接在編輯器中執行 Python。



# https://code.visualstudio.com

https://code.visualstudio.com

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn

Search Docs

Download

Version 1.74 is now available! Read about the new features and fixes from November.

## Code editing. Redefined.

Free. Built on open source. Runs everywhere.

Download for Windows  
Stable Build

Web, Insiders edition, or other platforms

By using VS Code, you agree to its  
license and privacy statement.

The screenshot shows the Visual Studio Code interface. On the left is the 'EXTENSIONS MARKETPLACE' sidebar with a list of extensions including Python, GitLens, C/C++, ESLint, Debugger for Chrome, Language Support for Java, VS Code Icons, Vetur, and C#. The main editor area shows a JavaScript file named 'serviceWorker.js' with code for a service worker. The bottom panel shows the 'TERMINAL' with the command 'i.node' and the output 'You can now view create-react-app in the browser.' and 'Local: http://localhost:3000/'.

IntelliSense

Run and Debug

Built-in Git

Extensions

**License Agreement**

Please read the following important information before continuing.



Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

*This license applies to the Visual Studio Code product. Source Code for Visual Studio Code is available at <https://github.com/Microsoft/vscode> under the MIT license agreement at <https://github.com/microsoft/vscode/blob/main/LICENSE.txt>. Additional license information can be found in our FAQ at <https://code.visualstudio.com/docs/supporting/faq>.*

**MICROSOFT SOFTWARE LICENSE TERMS****MICROSOFT VISUAL STUDIO CODE**

- ☒ I accept the agreement  
☐ I do not accept the agreement

**Next >**

Cancel



### Select Additional Tasks

Which additional tasks should be performed?



Select the additional tasks you would like Setup to perform while installing Visual Studio Code, then click Next.

Additional icons:

☒ Create a desktop icon

Other:

☒ Add "Open with Code" action to Windows Explorer file context menu

☒ Add "Open with Code" action to Windows Explorer directory context menu

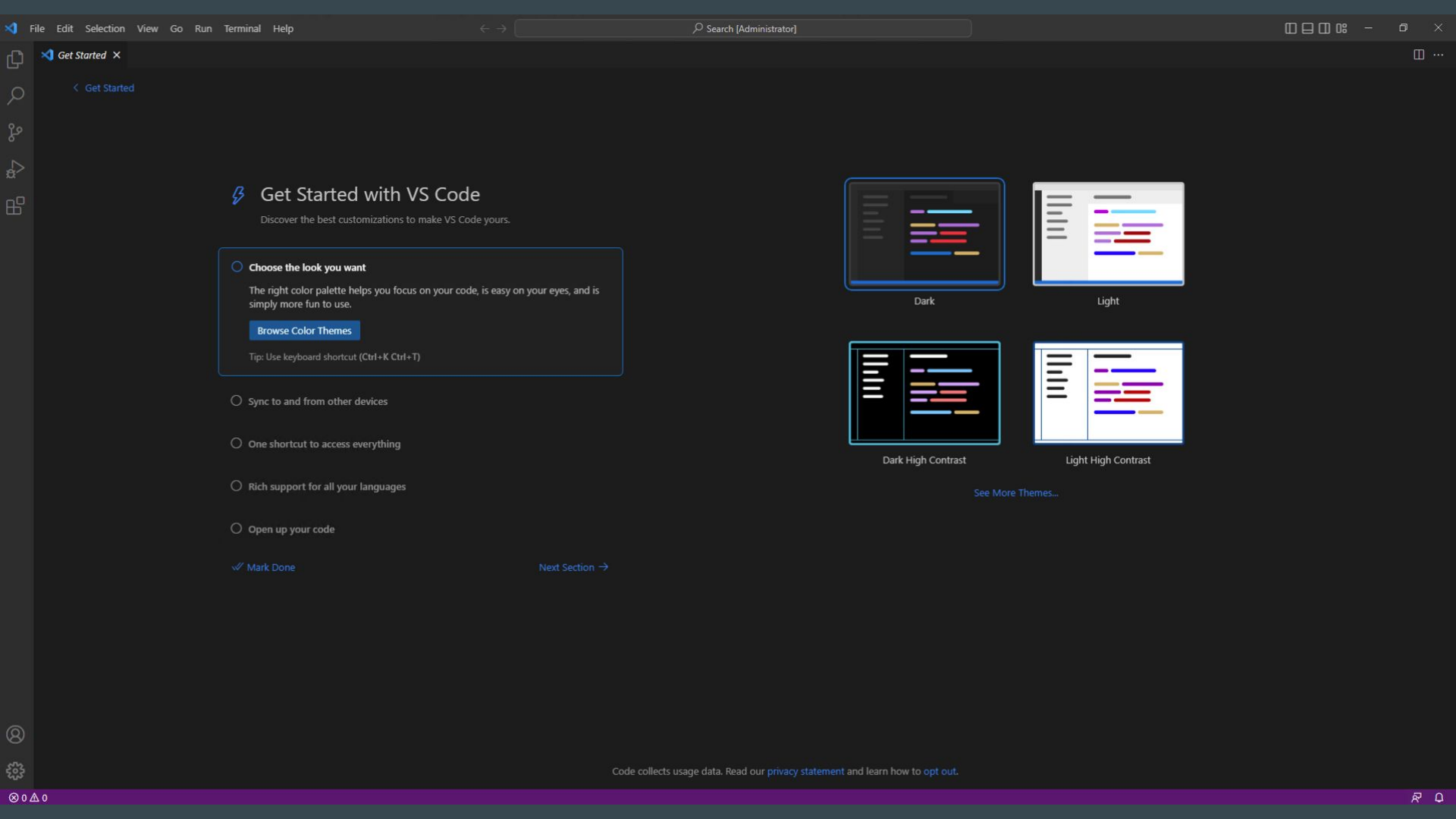
☒ Register Code as an editor for supported file types

☒ Add to PATH (requires shell restart)

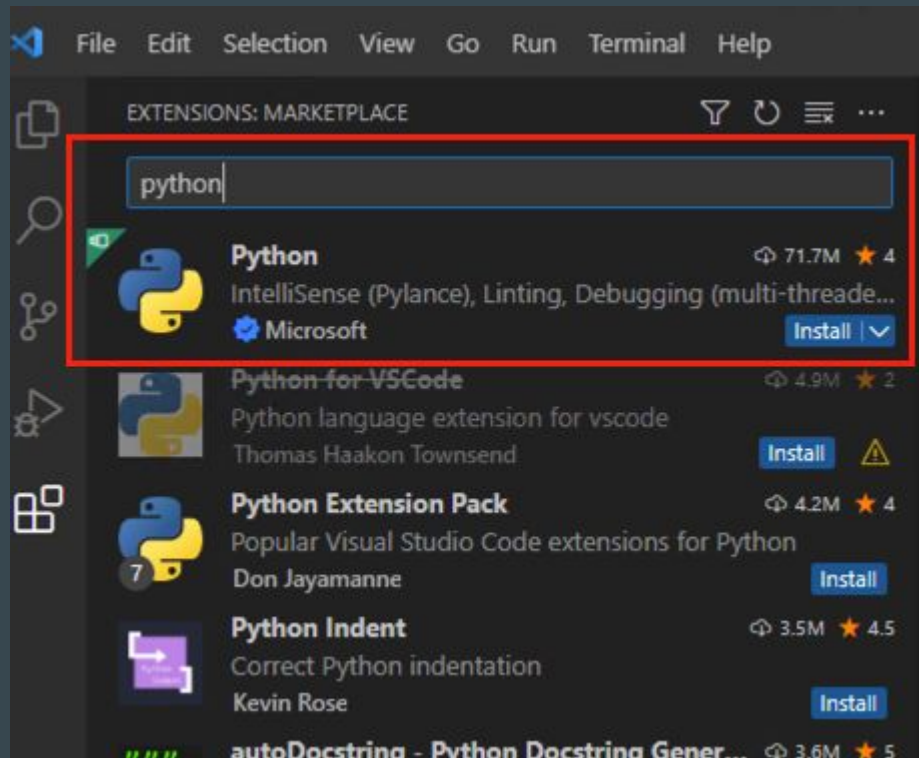
< Back

Next >

Cancel





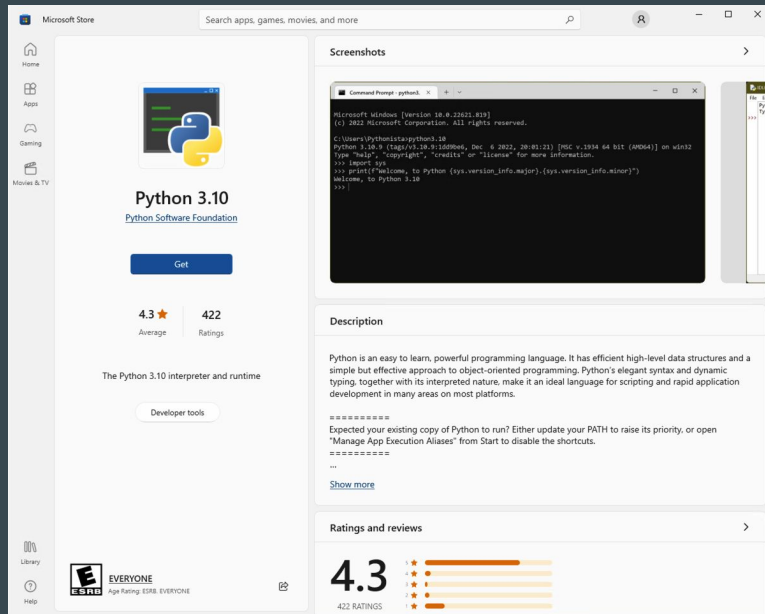


# 安裝 Python

Windows 可以在 Microsoft Store 裡安裝

Ubuntu 20.04 及以上版本自帶了 Python 3

Mac 可以用 homebrew 安裝: `brew install python`



# 創建虛擬環境，並且會切換

```
$ python -m venv my-django-venv
```

```
$ pip install -r requirements.txt
```

```
$ .\my-django-venv\Scripts\activate
```

```
$ pip freeze > requirements.txt
```

```
(Mac) source ./my-django-venv/bin/activate
```

```
$ deactivate
```

# 用 pyenv 來做 Python 多版本間的管理

Windows PS : Invoke-WebRequest -UseBasicParsing -Uri  
"https://raw.githubusercontent.com/pyenv-win/pyenv-win/master/pyenv-win/install-pyenv-win.ps1" -OutFile "./install-pyenv-win.ps1"; &"./install-pyenv-win.ps1"

Mac : brew install pyenv

Ubuntu : curl -L  
<https://github.com/pyenv/pyenv-installer/raw/master/bin/pyenv-installer> | bash

安裝 python 3.12 指令 : \$ pyenv install 3.12



Select a Python installation to create the virtual environment

+ Enter interpreter path...

⚙ Use Python from `python.defaultInterpreterPath` setting ~/opt/anaconda3/bin/python

Python (3.12.0) ~/.pyenv/versions/3.12.0/bin/python

Pyenv

Python (3.11.5) /usr/local/bin/python3

Global

Python (3.10.13) /usr/local/bin/python3.10

Python (3.9.18) /usr/local/bin/python3.9

Python (3.9.6) /usr/bin/python3

# 使用 Python Environment Manager 更容易操作



GLOBAL ENVIRONMENTS

- > Conda 4.13.0
- > PipEnv
- > Pyenv 2.3.31
- > Global
- > Venv

> .venv (3.11.5) .venv

**Name:** .venv  
**Version:** 3.11.5 (v3.11.5:cce6ba91b3, Aug 24 2023, 1300.0.29.30)]  
**Architecture:** 64-bit  
**Executable:** .venv/bin/python  
**SysPrefix:** .venv  
**Folder:** ~/Desktop/programming/2023modernweb  
**Environment Type:** Venv

2023MODERNWEB

- > .venv
  - > bin
  - > include
  - > lib
  - > .gitignore
  - > pyvenv.cfg

# 安裝 Django

要安裝 Long Term Support (LTS) 版本

<https://www.djangoproject.com/download/>

```
$ pip install Django==4.2.7
```



# 開啟一個 Django 專案

```
$ django-admin startproject chatgpt_project .
```

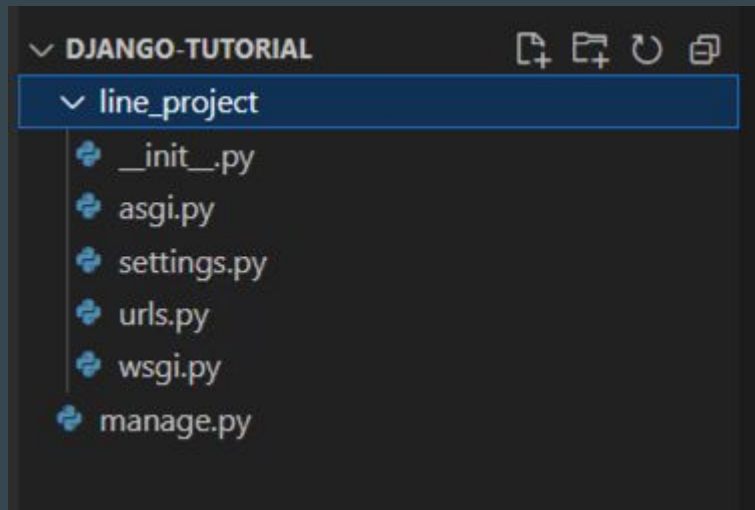
`__init__.py`: 一個空文件, 告訴 Python 這個文件夾是一個 Python package。

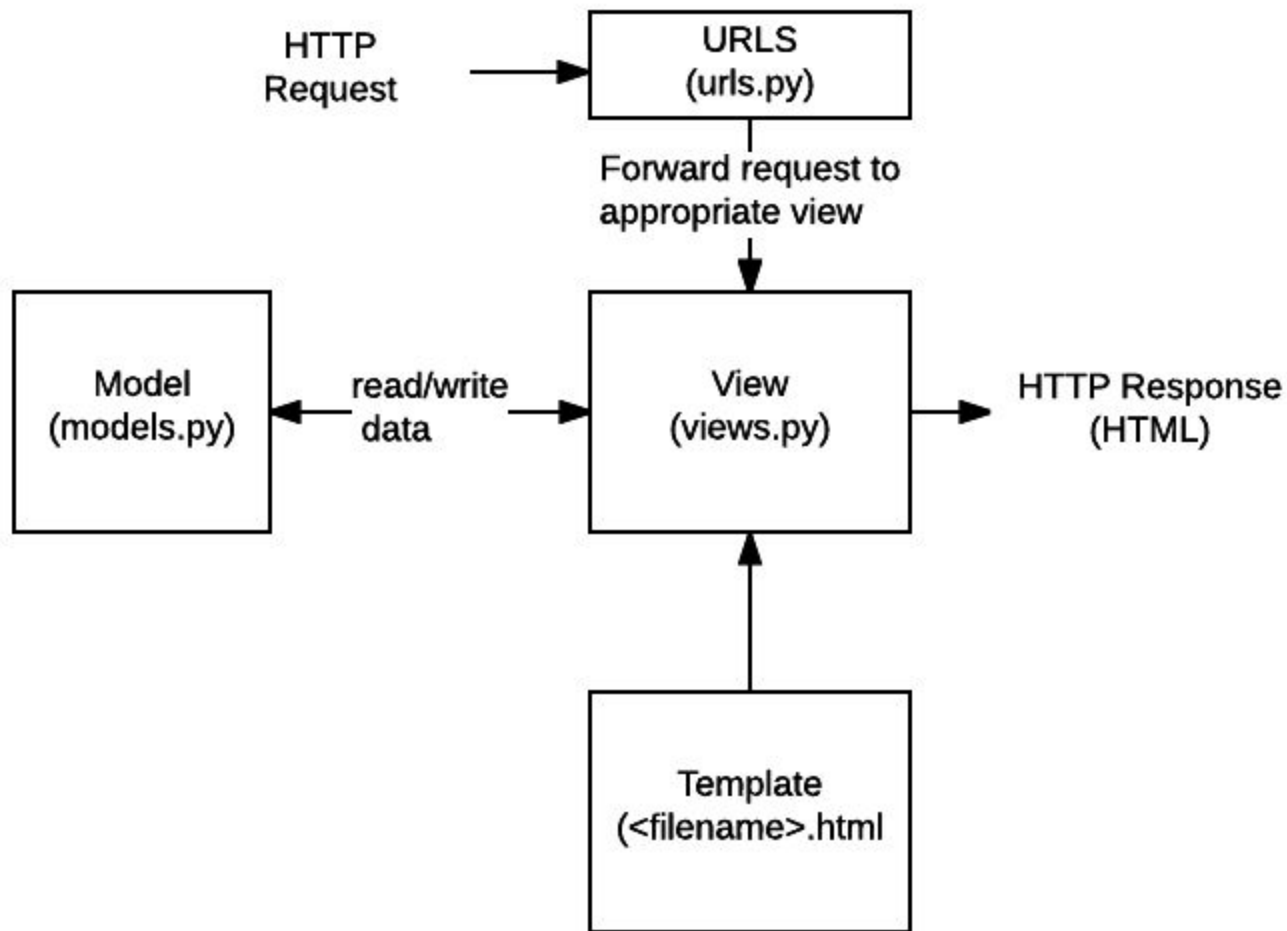
`asgi.py`: 專案的 ASGI 兼容 Web 服務器的入口點。

`settings.py`: 包含 Django 專案的設置, 可以在開發 Web 應用的過程中對其進行修改, 很常用到。

`urls.py`: 包含 Django 專案的目錄和路由, 您也可以可以在開發過程中對其進行修改。

`wsgi.py`: 專案的 WSGI 兼容 Web 服務器的入口點。





# 建立本地資料庫

```
$ python manage.py migrate
```

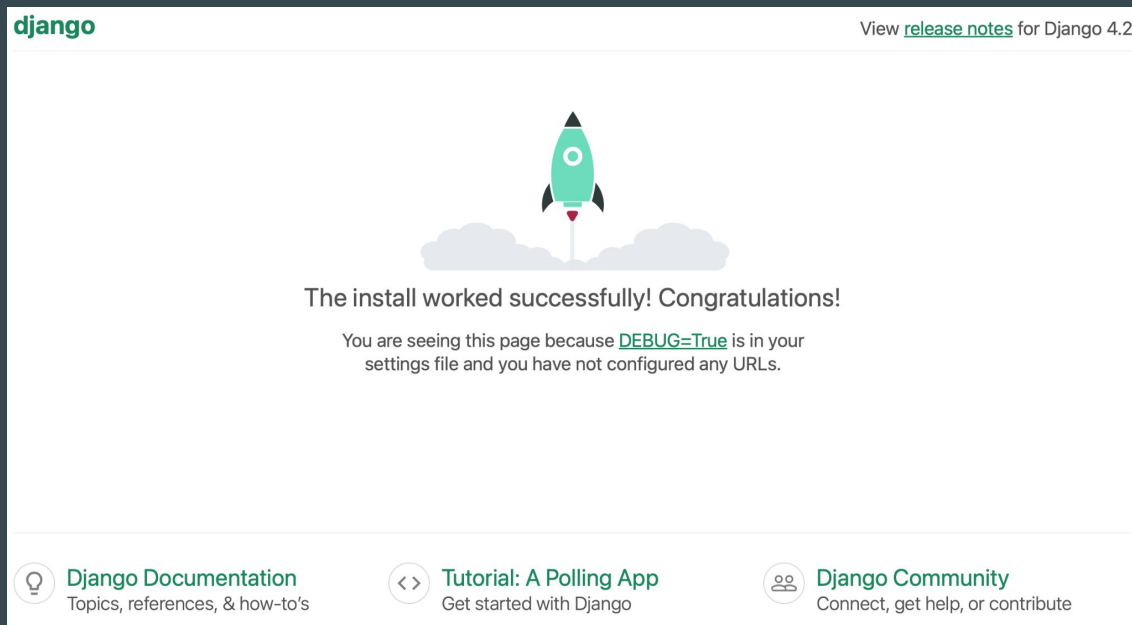
manage.py: 專案的 Django 指令管理程式。可以使用管理指令 `python manage.py <command> [options]`。

db.sqlite3 為內建的資料庫

# 本地 Server 跑起來

```
$ python manage.py runserver 5566
```

Ctrl+C 離開



# 建立 Django APP

```
$ python manage.py startapp first_app
```

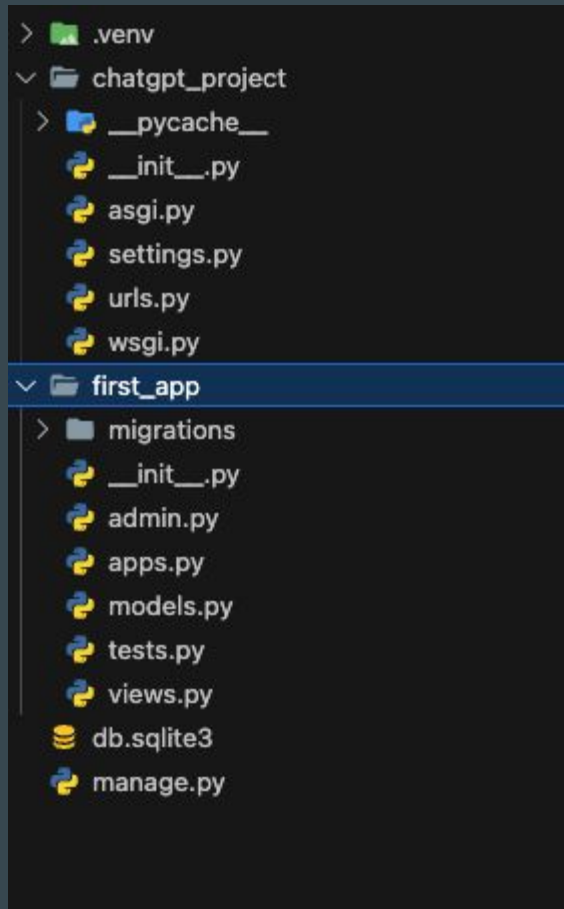
views.py: 控制器, 包含定義 Web 應用頁面的函數。

models.py: 資料庫存取

apps.py: 應用程式配置

admin.py: 用於創建管理界面

tests.py: 用於創建測試



# 來改寫 View

```
from django.http import HttpResponse
```

```
def home(request):
```

```
    return HttpResponse("Hello, Django!")
```

# 在 first\_app 裡建立一個 urls.py 檔

```
from django.urls import path
```

```
from first_app import views
```

```
urlpatterns = [  
    path("", views.home, name="home"),  
]
```

# 在 Project level 的 urls.py include 進去

```
from django.contrib import admin
```

```
from django.urls import include, path
```

```
urlpatterns = [
```

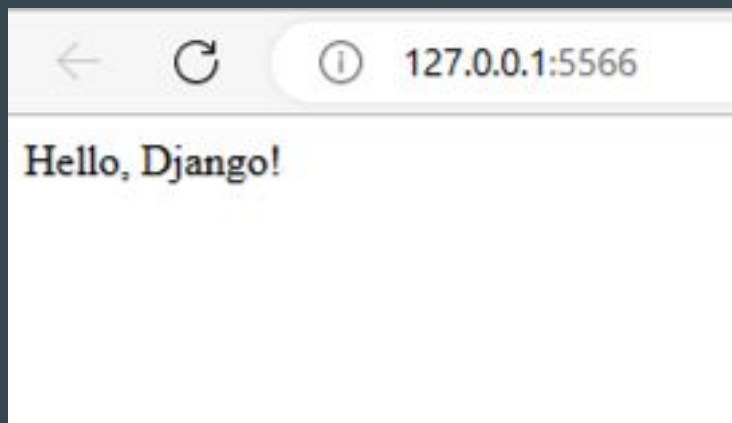
```
    path("", include("first_app.urls")),
```

```
    path('admin/', admin.site.urls)
```

```
]
```

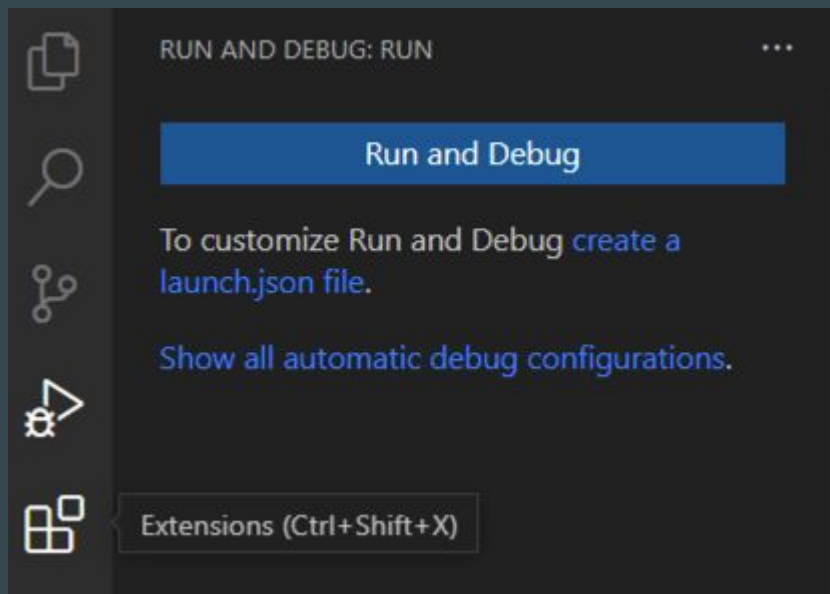


# 重新跑起伺服器



# 建立快速測試環境，不用每次打指令

之後可以用熱鍵 F5 來取代指令



```

1  {
2      // Use IntelliSense to learn about possible attributes.
3      // Hover to view descriptions of existing attributes.
4      // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5      "version": "0.2.0",
6      "configurations": [
7          {
8              "name": "Python: Django",
9              "type": "python",
10             "request": "launch",
11             "program": "${workspaceFolder}\\manage.py",
12             "args": [
13                 "runserver",
14             ],
15             "django": true,
16             "justMyCode": true
17         }
18     ]
19 }
20

```

# 新增 template

先建 templates 的 folder , 再往下建 first\_app 的 folder, 再建 hello\_template.html



# hello\_template.html

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="utf-8" />
```

```
    <title>Hello, Django</title>
```

```
  </head>
```

```
  <body>
```

```
    <strong>Hello, {{ name }}!</strong> Template 時間 {{ date | date:"l, d F, Y" }} at {{ date | time:"H:i:s" }}
```

```
  </body>
```

```
</html>
```

# 在 settings.py 加入 app

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'first_app'  
]
```

# 在 views.py 新增一個 function

```
def hello_template(request, name):
```

```
    return render(
```

```
        request,
```

```
        'first_app/hello_template.html',
```

```
        {
```

```
            'name': name,
```

```
            'date': datetime.now()
```

```
        }
```

```
    )
```

# 記得要去 app level 的 urls.py 新增路由

```
from django.urls import path
```

```
from first_app import views
```

```
urlpatterns = [
```

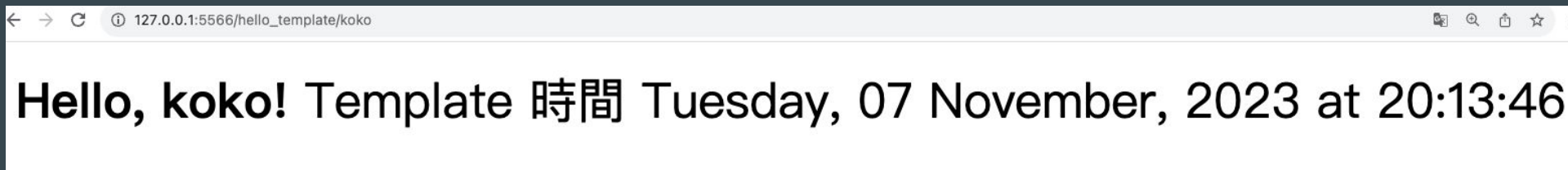
```
    path("", views.home, name="home"),
```

```
    path("hello_template/<name>", views.hello_template, name="hello_template"),
```

```
]
```



# 接著訪問網址



# 建立 Model 來儲存 ChatGPT 產生的故事


```
class StoryModel(models.Model):  
  
    id = models.AutoField(primary_key=True)  
  
    text = models.TextField()  
  
    path = models.TextField()
```

# Migrate 資料庫

```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
```


# 安裝 plugin 以便看 sqlite 的資料



## SQLite Viewer v0.3.13

Florian Klampfer [qwtel.com](https://qwtel.com) | 687,435 | ★★★★★ (41)

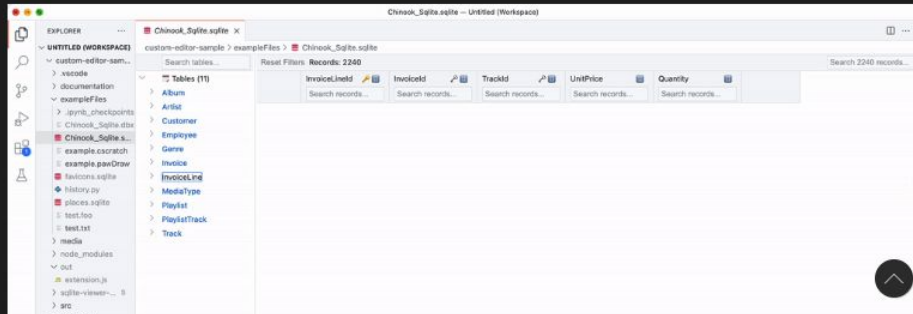
SQLite Viewer for VSCode

[Install](#) 

[DETAILS](#) [FEATURE CONTRIBUTIONS](#) [CHANGELOG](#)

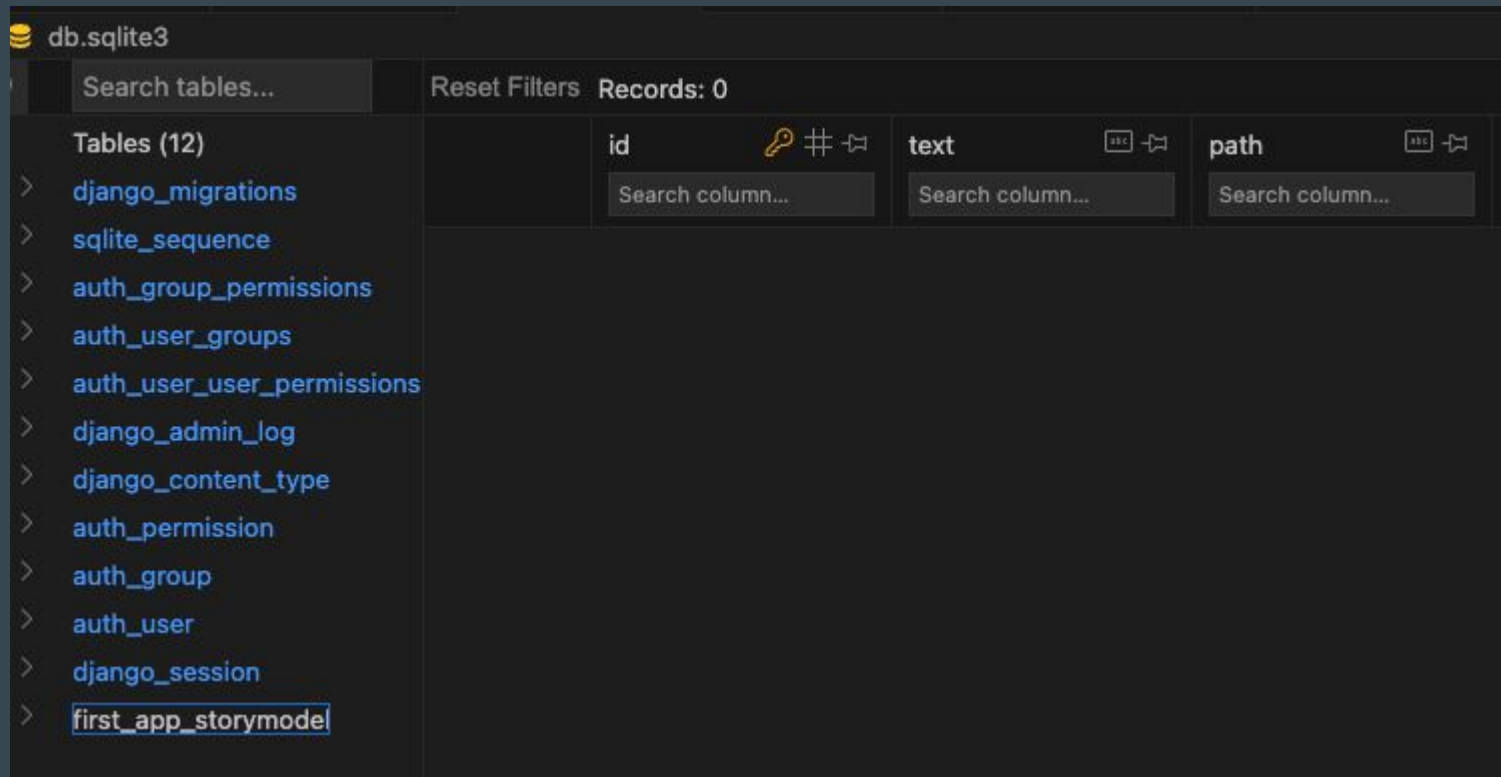
### SQLite Viewer for VSCode

A quick and easy SQLite viewer for VSCode, inspired by DB Browser for SQLite and Airtable.



The screenshot shows the SQLite Viewer interface within a VSCode workspace. The left sidebar displays the Explorer view with a file tree. The main editor area shows a table with columns: InvoiceLineId, InvoiceId, TrackId, UnitPrice, and Quantity. The table contains 2240 records. The interface is clean and modern, with a dark theme.

# 剛剛建立的 model 已經 migrate 成資料庫 table



# 使用 manage.py shell 來插入資料

```
$ python manage.py shell
```

```
>>> from first_app.models import StoryModel
```

```
>>> StoryModel.objects.create(text='Example', path='root')
```

ctrl + D 離開

# 接著就可以看到我們插入的資料

Search tables...	Reset Filters	Records: 1			
Tables (12)		id	text	path	
> django_migrations		Search column...	Search column...	Search column...	
> sqlite_sequence	1	1	Example	root	
> auth_group_permissions					
> auth_user_groups					
> auth_user_user_permissions					
> django_admin_log					
> django_content_type					
> auth_permission					
> auth_group					
> auth_user					
> django_session					
> first_app_storymodel					

# 接著我們來 render 資料到前端, 先在 views.py 加上

```
from first_app.models import StoryModel
```

```
def hello_model(request, id):
```

```
    return render(
```

```
        request,
```

```
        'first_app/hello_model.html',
```

```
        {
```

```
            'object': StoryModel.objects.get(id=id)
```

```
        }
```

```
    )
```



# 再新建一個 template 叫 hello\_model.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <meta charset="utf-8" />
```

```
    <title>Hello, Django</title>
```

```
</head>
```

```
<body>
```

```
    <h1>Text: {{ object.text }}!</h1>
```

```
    <h1>Path: {{ object.path }}.</h1>
```

```
</body>
```

```
</html>
```

# 記得要去 app level 新增路由

```
path("hello_model/<id>", views.hello_model, name="hello_model"),
```

**Text: Example!**

**Path: root.**

# 在 admin.py 把要看的 model 註冊進去

```
from first_app.models import StoryModel
```

```
admin.site.register(StoryModel)
```

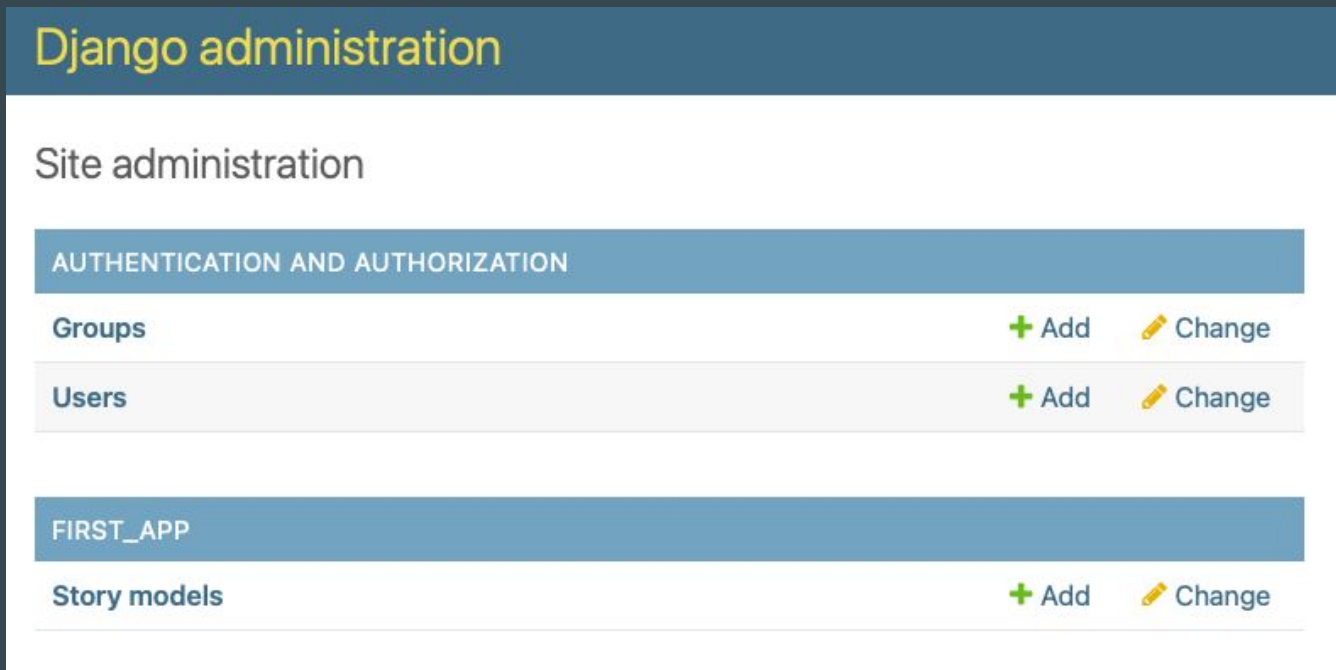
# 接著建立一個 super user

\$ python manage.py createsuperuser

```
Username (leave blank to use 'koko'): koko
Email address: qqkerk@hotmail.com
Password:
Password (again):
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: N
Password:
Password (again):
The password is too similar to the email address.
Bypass password validation and create user anyway? [y/N]: N
Password:
Password (again):
Superuser created successfully.
```

# 登入到 admin 後台

<http://127.0.0.1:5566/admin>



The screenshot displays the Django administration interface. At the top, a blue header bar contains the text "Django administration" in yellow. Below this, the "Site administration" section is visible. It features a blue bar labeled "AUTHENTICATION AND AUTHORIZATION". Under this bar, there are two rows: "Groups" and "Users". Each row has a green plus icon followed by the text "Add" and a yellow pencil icon followed by the text "Change". Below these rows is another blue bar labeled "FIRST\_APP". Under this bar, there is a row labeled "Story models", which also has a green plus icon followed by "Add" and a yellow pencil icon followed by "Change".

Django administration

Site administration

**AUTHENTICATION AND AUTHORIZATION**

Groups	<a href="#">+ Add</a>	<a href="#">Change</a>
Users	<a href="#">+ Add</a>	<a href="#">Change</a>

**FIRST\_APP**

Story models	<a href="#">+ Add</a>	<a href="#">Change</a>
--------------	-----------------------	------------------------

# 可以CRUD的管理後台就建起來了

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

FIRST\_APP

Story models + Add

Change story model

HISTORY

StoryModel object (1)

Text:

Example

Path:

root

SAVE

Save and add another

Save and continue editing

Delete

# 開始串接 ChatGPT 做小熊故事機，開始難度飆高



```
$ pip install azure-cognitiveservices-speech openai
```

注意這裡使用的是 Azure Open AI, 而不是 Open AI。



# 建立 .env 檔, 並放入下面參數

OPENAI\_API\_TYPE=azure

OPENAI\_API\_BASE=https://XXX.openai.azure.com/

OPENAI\_API\_VERSION=2023-08-01-preview

OPENAI\_API\_KEY=XXXX

AZURE\_SPEECH\_KEY=XXXX

AZURE\_SPEECH\_REGION=westus2

# 安裝 python-dotenv

這是一個很好用的環境變數讀取工具

```
$ pip install python-dotenv
```

然後進到 settings.py, 加上下面的程式碼

```
import dotenv
```

```
dotenv.load_dotenv()
```

# 配置 static 資源

在 settings.py 加上：

```
MEDIA_URL = "/media/"
```

```
MEDIA_ROOT = os.path.join(BASE_DIR, "media")
```

在 project level 的 url 加上：

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

```
from django.conf import settings
```

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

# 建立一個 utils.py

```
import os
import uuid
from openai import AzureOpenAI
import azure.cognitiveservices.speech as speechsdk
from django.conf import settings

speech_key, service_region = os.getenv("AZURE_SPEECH_KEY"), os.getenv("AZURE_SPEECH_REGION")
```

# 建立一個 utils.py, 注意 Open AI SDK 在 1.0 後大改版

```
def get_openai_response(prompt: str) -> str:
```

```
    client = AzureOpenAI(  
        api_version=os.getenv("OPENAI_API_VERSION"),  
        azure_endpoint=os.getenv("OPENAI_API_BASE"),  
        api_key=os.getenv("OPENAI_API_KEY"),  
    )
```

```
    completion = client.chat.completions.create(  
        model="jpchat35", # your deployment name  
        messages=[{"role": "system", "content": "你是一個童話故事作家, 請幫我根據user的動物與場景, 來編一個童話故事。"},  
                  {"role": "user", "content": prompt}],  
        temperature=0.7,  
        max_tokens=2000,  
        top_p=0.95,  
        frequency_penalty=0,  
        presence_penalty=0,  
        stop=None  
    )
```

```
    return completion.choices[0].message.content
```

# 建立一個 utils.py

```
def send_to_tts(story: str):
    speech_config = speechsdk.SpeechConfig(subscription=speech_key,
                                           region=service_region)
    speech_config.speech_synthesis_voice_name = "zh-TW-YunJheNeural"
    speech_config.speech_synthesis_language = "zh-TW"
    speech_config.set_speech_synthesis_output_format(speechsdk.SpeechSynthesisOutputFormat.Audio24Khz48KBitRateMonoMp3)

    audio_filename = uuid.uuid4().hex + ".mp4"
    static_path = os.path.join(settings.MEDIA_ROOT, audio_filename)

    audio_config = speechsdk.AudioConfig(filename=static_path)

    speech_synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config, audio_config=audio_config)

    result = speech_synthesizer.speak_text_async(story).get()
    # Check result
    if result.reason == speechsdk.ResultReason.SynthesizingAudioCompleted:
        return f"{settings.MEDIA_URL}/{audio_filename}"
    elif result.reason == speechsdk.ResultReason.Canceled:
        cancellation_details = result.cancellation_details
        print("Speech synthesis canceled: {}".format(cancellation_details.reason))
        if cancellation_details.reason == speechsdk.CancellationReason.Error:
            print("Error details: {}".format(cancellation_details.error_details))
```

# 建立一個 forms.py

```
from django import forms
```

```
class StoryForm(forms.Form):
```

```
    ANIMAL_CHOICES = [
```

```
        ('熊熊', '熊熊'), ('兔兔', '兔兔'), ('貓貓', '貓貓'), ('狗狗', '狗狗'),
```

```
    ]
```

```
    SCENE_RADIO = [
```

```
        ('都市', '都市'), ('森林', '森林'), ('學校', '學校'), ('海邊', '海邊'),
```

```
    ]
```

```
    animals = forms.MultipleChoiceField(
```

```
        widget=forms.CheckboxSelectMultiple(attrs={'class': 'form-control'}), choices=ANIMAL_CHOICES,
```

```
        label='選擇動物,
```

```
    )
```

```
    scene = forms.ChoiceField(
```

```
        widget=forms.RadioSelect(attrs={'class': 'form-control'}),
```

```
        choices=SCENE_RADIO,
```

```
        label='選擇場景,
```

```
    )
```

# Template 建立一個 index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>小熊故事機</title>
</head>

<body style="text-align: center">
  <h1>小熊故事機</h1>
  <form method="post" class="flex-container">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">說故事</button>
  </form>

  {% if audio %}
  <audio controls>
    <source src="{{ audio }}" type="audio/mpeg">
  </audio>
  {% endif %}

  {% if story %}
  <p>{{ story }}</p>
  {% endif %}
```



# 使用優雅的 Class-Based Views

```
from django.views import View
from .forms import StoryForm
from .utils import get_openai_response, send_to_tts
```

```
class StoryTellerView(View):
    template_name = 'storyteller/index.html'

    def get(self, request):
        form = StoryForm()
        return render(request, self.template_name, {'form': form})
```

```
    def post(self, request):
        form = StoryForm(request.POST)
        story = ""
        if form.is_valid():
            selected_animals = form.cleaned_data['animals']
            selected_animals_str = ','.join(selected_animals)

            selected_scene = form.cleaned_data['scene']

            story = get_openai_response(selected_animals_str + "在" + selected_scene + "。")
            print(story)

            audio_path = send_to_tts(story)

            StoryModel.objects.create(
                text=story,
                path=audio_path,
            )

        return render(request, self.template_name, {'form': form, 'story': story, 'audio': audio_path})
```

# 設定路由

```
from .views import StoryTellerView
```

```
    path("storyteller/", StoryTellerView.as_view(), name="storyteller"),
```

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

FIRST\_APP

Story models + Add

## Change story model

HISTORY

### StoryModel object (2)

#### Text:

從前從前，有一隻叫做貓貓的小貓咪和一隻叫做狗狗的小狗狗，他們住在一個神奇的森林裡。

這個森林充滿了奇妙的生物和美麗的景色。貓貓和狗狗經常一起探索這個森林，他們彼此成為了最好的朋友。

#### Path:

/media/6bb9d912112446249ec9214dccb4470f.mp4

# 實戰工作坊重點回顧

1. 了解 Django 的前世今生
2. 了解 Python 虛擬環境架設
3. 了解如何建立 Django 專案
4. 了解 MTV 架構
5. 了解路由
6. 了解 ORM 與資料庫串接
7. 了解 Admin 後台
8. 串接 ChatGPT做出小熊故事機

地方爸爸就不用買小熊故事機了！

(但是為了 yoyo 台的歌....



# 不錯的 Django 學習資源

## 我的教學影片

<https://www.youtube.com/watch?v=4V34CKV82sA>

## 微軟寫的實戰課程含部署

[https://learn.microsoft.com/en-us/training/paths/django-create-data-driven-websites/?wt.mc\\_id=AI-MVP-5003846](https://learn.microsoft.com/en-us/training/paths/django-create-data-driven-websites/?wt.mc_id=AI-MVP-5003846)

進階的實戰書: Django 4 by example



## 用 Django 快速打造 Chatbot

LIVE DEMO

- 開發環境架設
- Django 架構一覽
- LINE SDK 入門
- 本地伺服器架設
- LINE 串接
- LIFF

Ko Ko, Microsoft AI MVP

2023/03/11 @Chatbot 社群小聚 @彰化青年職涯發展中心

001 / 39:04



# 今天教 Python web，結果我寫的书不是用 Python

天瓏書局七月銷售排行第二名

第一本針對軟體工程師所撰寫之 ChatGPT 專書

用 .NET 和 Node.js 雙語言範例程式碼

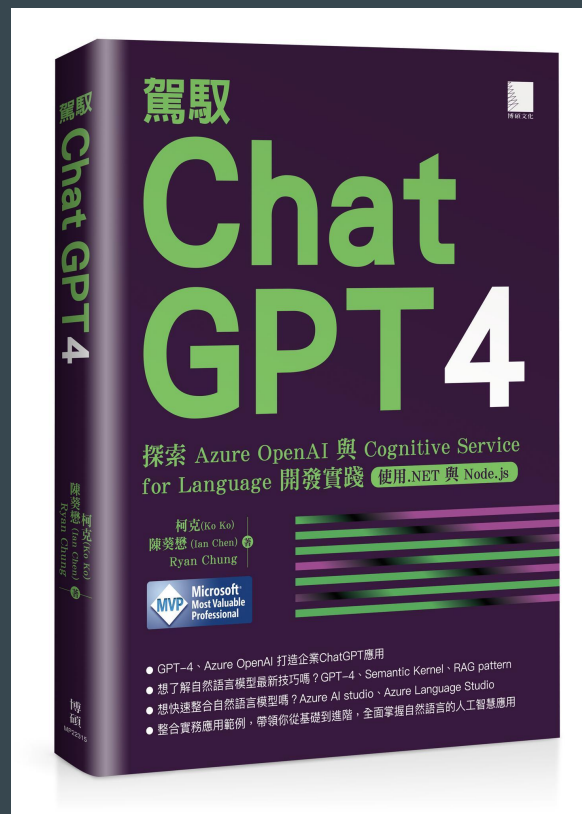
第一本針對 Azure OpenAI 所撰寫之專書

第一本針對 Cognitive Service for Language 專書

第一本教你如何 Fine tune GPT 模型之專書

第一本教你 Semantic Kernel 與 RAG pattern 專書

大量範例與實戰，連最新的 Bring your own data 都有





# Thank you and Q & A

