部落格網站的需求與規劃

網站之需求與功能描述：

- 專案名稱myblog
- 透過admin管理介面張貼、編輯以及刪除貼文，且此介面支援Markdown語法
- 使用Bootstrap網頁框架
- 在主頁中可以顯示每一篇文章的標題、簡短摘要以及張貼日期
- 在主頁中加入側邊欄，可以加入自訂的HTML以及Javascript網頁碼
- 在顯示文章時>可以解析Markdown語法並正確顯示出排版後的樣子

由於是簡單的入門示範網站，所以在設定上只有管理者一人可以張貼以及管理文章，

因此不需要有使用者的註冊、登入等權限管理。此外，在資料庫中僅儲存文章的原始資料，但此資料支援Markdown語法>可以在文章顯示時做為排版的依據，不提供所視即所得(WYSIWYG)的文章編輯介面。另外，所有的圖形檔採用第三方網站儲存的方式，本部落格要顯示的圖片>需以外部連結的方式，透過Markdown語法設定在文章中，並在顯示該篇文章時顯示在指定的文章位置。

產生第一個網站框架

- 要建立的個人部落格名稱為myblog
- 請依照在上一堂課學習到的內容,先在Bitbucket中建立一個同名的倉庫,以供未來在不同電腦間開發之用。

```
minhuang@ubuntu:~/myDjango$ source VENV/bin/activate
(VENV) minhuang@ubuntu:~/myDjango$ django-admin startproject mblog
(VENV) minhuang@ubuntu:~/myDjango$ cd mblog
(VENV) minhuang@ubuntu:~/myDjango/mblog$ python manage.py startapp mainsite
(VENV) minhuang@ubuntu:~/myDjango/mblog$ cd ..
(VENV) minhuang@ubuntu:~/myDjango$ tree mblog
```

```
cd C:\Users\shiuny\VENV01\Scripts
activate
cd C:\Users\shiuny\Dropbox\文件\ex
django-admin startproject myblog
cd myblog
python manage.py startapp mainsite
```

```
(VENV) minhuang@ubuntu:~/myDjango$ tree mblog
mblog
├── mainsite
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── manage.py
└── mblog
    ├── __init__.py
    ├── __init__.pyc
    ├── settings.py
    ├── settings.pyc
    ├── urls.py
    └── wsgi.py

3 directories, 14 files
```

tree myblog /F

執行框架預載範例

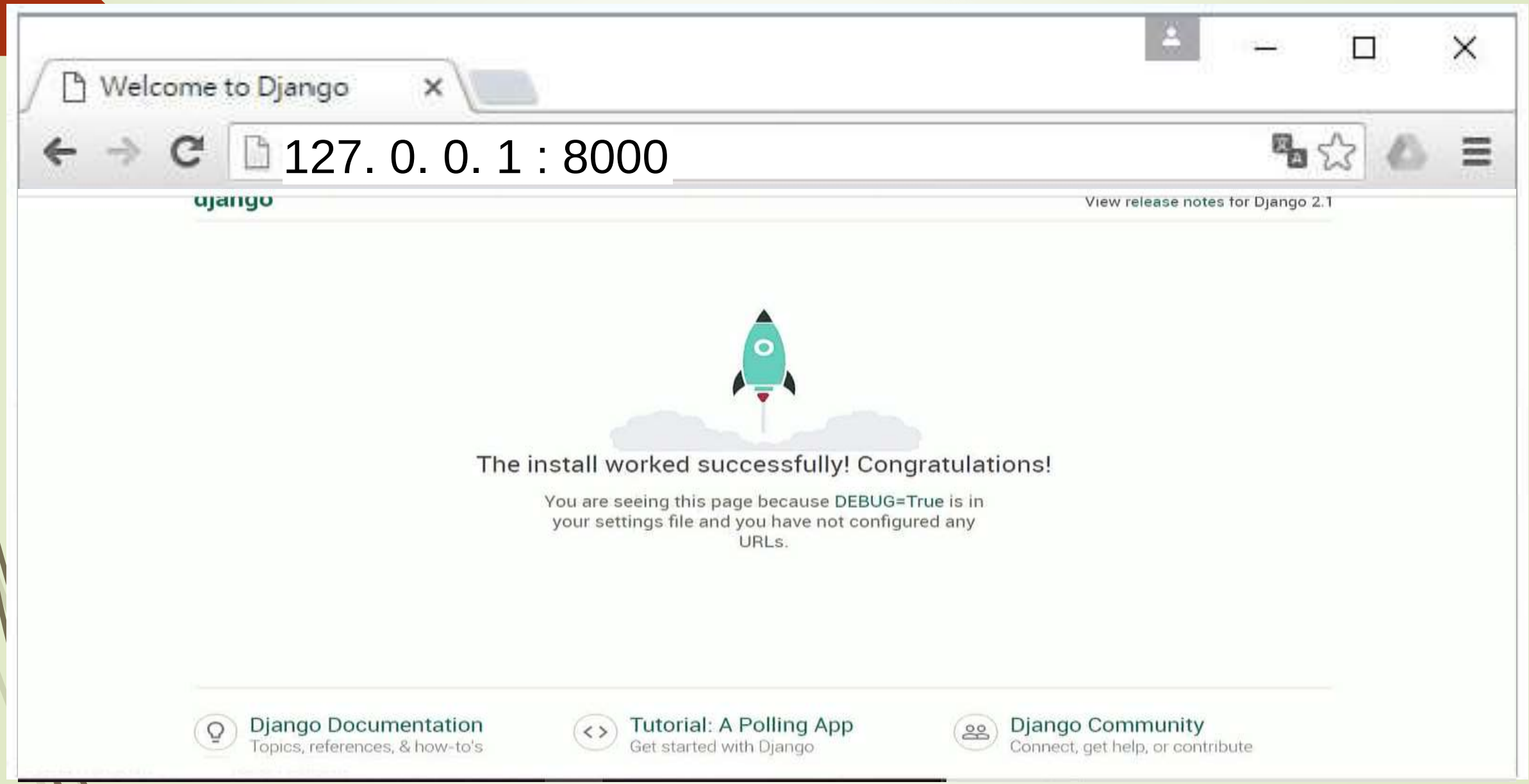
- cd myblog
- python manage.py runserver 127.0.0.1:8000

```
Performing system checks...

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work properly until they are
applied.
Run 'python manage.py migrate' to apply them.

May 27, 2016 - 06:50:38
Django version 1.9.6, using settings 'mblog.settings'
Starting development server at http://192.168.161.131:8000/
Quit the server with CONTROL-C.
```


Welcome to Django
127. 0. 0. 1 : 8000
View release notes for Django 2.1
The install worked successfully! Congratulations!
You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.
Django Documentation
Topics, references, & how-to's
Tutorial: A Polling App
Get started with Django
Django Community
Connect, get help, or contribute

產生第一個網站框架

- 執行以下指令測試一下 (但是要使用ip a查詢一下此台虛擬機的網路位址，在此例中為192.168.45.84，由於有指定特定 ip而非使用localhost，所以得在mYblog資料夾內的mYblog資料夾裡，開啟setting.py設定
- ALLOWED_HOSTS = ['192.168.161.131',] 或是 ALLOWED_HOSTS = ['*',]) ：
  - (VENV) minhuang@ubuntu:~/myDjango$ cd mblog
  - (VENV) minhuang@ubuntu:~/myDjango/mblog$ python manage.py runserver 192.168.161.131:8000
  - Performing system checks...
  - System check identified no issues (0 silenced).
  - You have unapplied migrations; your app may not work properly until they are applied.
  - Run 'python manage.py migrate' to apply them.
  - December 11, 2017 - 02:43:34
  - Django version 2.0, using settings 'mblog.settings'
  - Starting development server at http://192.168.161.131:8000/
  - Quit the server with CONTROL-C.

1.3 活用版本控制系統

在程式檔案愈來愈多，以及可能會在不同的電腦上開發同一套網站，甚至是一群人共同開發同一套網站專案的情況之下，如何才能夠協調所有的成員以及協同各台電腦，共同維護同一個版本的程式碼，是非常重要的一個課題。而版本控制則是用來解決這些問題的主要核心技術之一。

1.3.1 版本控制系統 Git 簡介

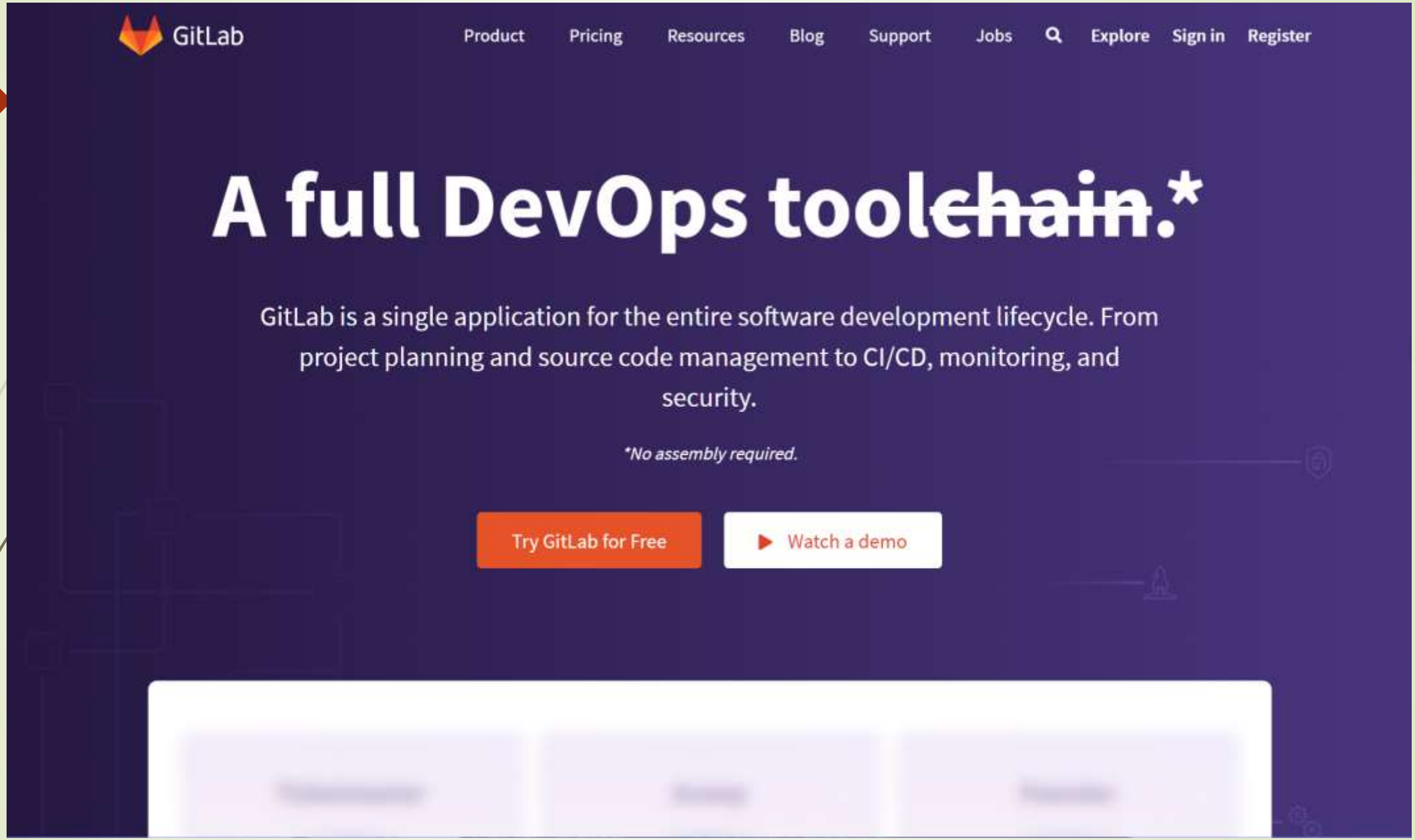
想要一個人在不同的電腦之中開發相同的網站專案，或是由許多不同的人一起共同開發同一個專案，版本控制是非常重要的技巧。版本控制有許多不同的方法，但是主要的精神就在於，讓所有的人永遠瞭解整個專案的全貌，以及自己目前手上的這一份程式碼在整個專案中的位置和扮演的角色。而版本控制系統隨著運作的邏輯和程式碼資料保存的位置的不同，主要分為集中式和分散式兩大類，Git 是分散式版本控制系統中最受歡迎的專案，由於 Git 就是 Linux 的發明人所開發的，而且也是 Linux 用來控制版本的工具，所以幾乎每一個版本的 Linux 作業系統（當然也包括我們之前安裝的 Ubuntu）預設都已經有 Git 了，只要直接使用即可。

因為我們主要的目的在於自己一個人在不同的電腦中開發同一個網站專案，所以我們要做的設定主要是：

1. 在本地虛擬機環境中建立 Git 的本地倉庫（其實就是一個 Git 所管理的目錄）
2. 在 Git 的倉庫中進行專案的開發，然後使用 Git 指令維護這些程式碼相關檔案
3. 在遠端（Bitbucket）建立一個遠端倉庫
4. 在每一次結束本地專案編輯時，把本地的倉庫和遠端的倉庫進行同步的作業
5. 日後每一次在任一台電腦中開始編輯專案之前，透過 Git 指令把遠端倉庫中的內容，同步到本地倉庫中

只要在每一次開始專案編輯的時候都秉持以上的原則，就不用再擔心不同電腦之間網站程式碼內容不一致的情形了。再加上我們使用虛擬機建立相同的作業系統版本，自然可以做到在不同的電腦中開發同一專案網站的目標了。

在練習使用 Git 指令之前，先讓我們來申請並建立一個免費的 Git 遠端倉庫。



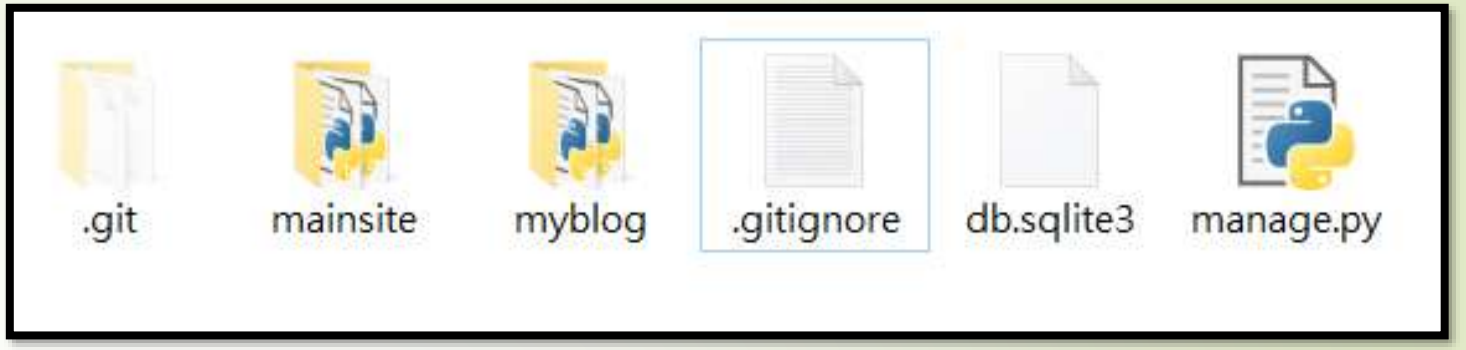
https://about.gitlab.com/

指令

我們把它節錄如下：

```
$ mkdir /path/to/your/project
$ cd /path/to/your/project
$ git init
$ git remote add origin https://wpgoin@bitbucket.org/wpgoin/myweb.git
```

指令

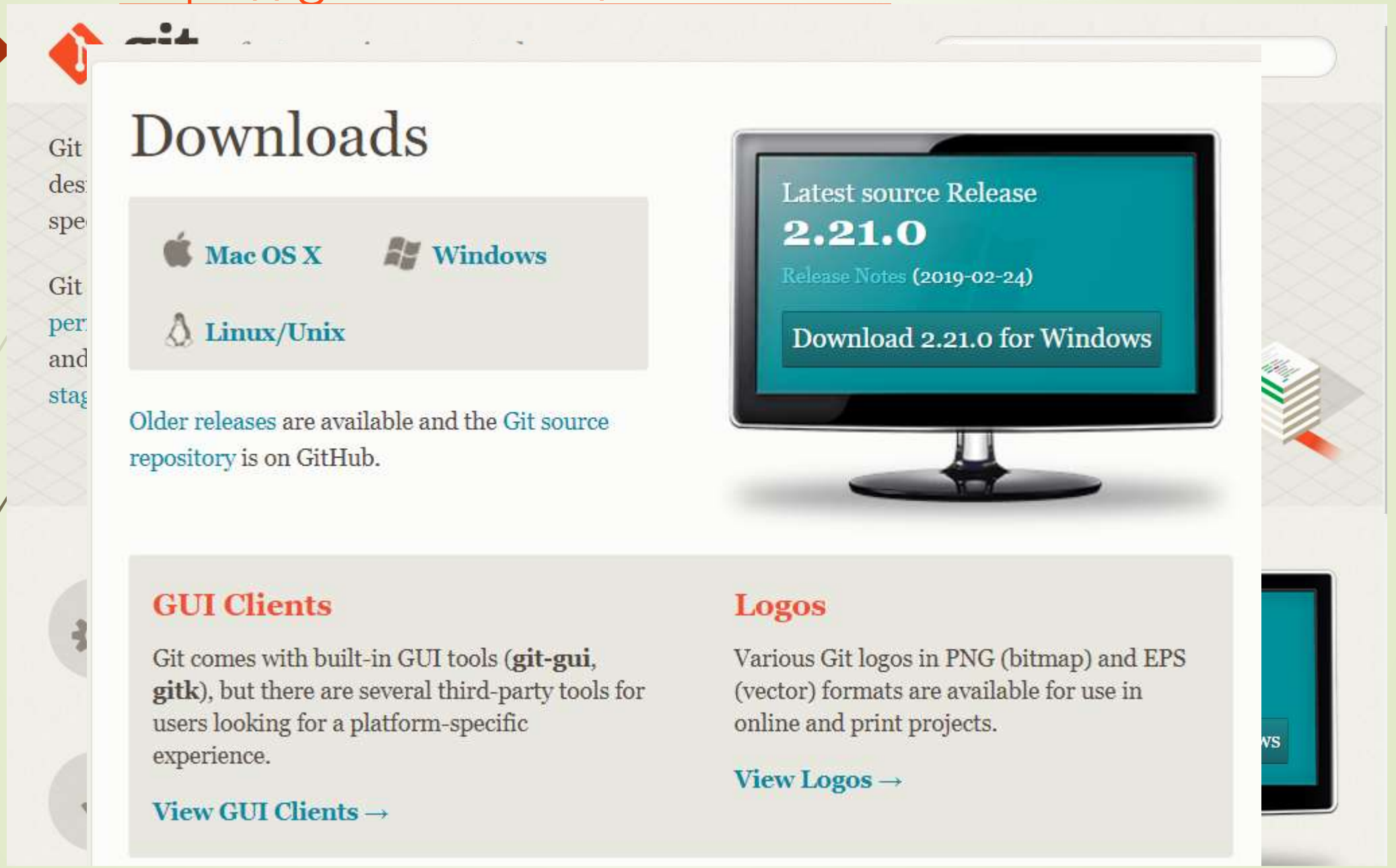


- git status 看狀態
- 建立「.gitignore」檔案
  - 裡面放不要同步的資料

Push an existing folder

```
cd existing_folder
git init
git remote add origin https://gitlab.com/shiunyi71/django_p01_myblog.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

- https://git-scm.com/downloads



建立git使用者資訊

- 安裝完git之後
- 建立git使用者資訊

- git config –global user.name “xxxxx”
- git config –global user.emal “xxxxx@xxxxx”

有一點必需要注意的地方，我們上傳的內容，是要編輯的開發中的 Django 網站專案（以此例為 myweb），所以上傳的內容只有網站本身，並不包含 Python 虛擬環境，當然也表示內容中沒有在此專案中使用 pip 安裝的套件。這些在虛擬環境中安裝的套件，也都會被記錄在虛擬環境目錄中（以此例為 VENV），因此，我們還必需使用 pip freeze 建立套件清單才行。習慣上，我會把這些套件清單，以 pip freeze 建立在網站專案的同一個目錄中（以此例為 myweb），指令如下（在目錄 myweb 下操作）：

```
(VENV) $ pip freeze > requirements.txt
(VENV) $ git add .
(VENV) $ git commit -m 'add requirements.txt'
(VENV) $ git push
```

1.3.4　在不同的電腦之間開發同一個網站

在前一小節中，我們使用 git push 指令把本地倉庫的檔案內容（以此例，是在 myweb 資料夾底下所有的檔案以及資料夾）通通都存放在位於 Bitbucket 的遠端倉庫中。而且，只要一開始的設定是正確的，之後在本地的資料夾中的任何內容如果有所更動，只要再一次使用 git push 指令即可完成更新的作業。

那麼，如果這時候我們使用的是另外一台電腦，而且都還沒有在這一台電腦中編輯過這些網站專案（myweb）的話，要如何開始呢？同樣地，也是先把 git 安裝好，並假設我們在另外一台電腦也是使用虛擬主機，並安裝了同模版本的作業系統，第一次使用時，不需要再重新建立目錄以及安裝，只要把這個專案複製（clone）下來即可。而這個動作對於每一台你新使用的電腦只要執行一次即可。指令如下：

```
(VENV) $ git clone https://wpgoin@bitbucket.org/wpgoin/myweb.git
```

也就是我們的遠端倉庫的位置，同樣地，在輸入正確的密碼之後，Git 就會在本地端建立一個 myweb 的資料夾，把所有該資料夾中的內容通通複製一份到 myweb 資料夾中。由於 Python 的虛擬環境以及額外安裝的套件並沒有儲存在倉庫中，所以虛擬環境要自行建立之後再使用 git clone 指令，同時在 clone 下來之後，還要使用 pip install 指令，把使用到的套件在本地電腦再安裝（或更新）一遍，指令如下：

```
(VENV) $ cd myweb
(VENV) $ pip install -r 'requirements.txt'
```

接下來就可以放心地開始編輯這個網站專案了。在結束編輯工作時，如果有新安裝的 Python 套件，則使用 pip freeze 更新 requirements.txt，再使用 git push 同步本地和遠端的倉庫即可。

git clone 的動作只要做一次，以後，在不同的電腦（但是已設定過的）開始編輯作業時，基本程序如下：

- 使用 source VENV/bin/activate 進入虛擬環境

- 切換到專案目錄之下
- 使用 git pull 從遠端倉庫拉取最新版本的網站資料到本地端倉庫
- 使用 pip install –r 'requirements.txt'，安裝所有使用到的套件
- 開始編輯作業
- 測試完畢，要結束此電腦的作業時，再使用 pip freeze > requirements.txt 確定目前使用到的所有的套件
- 使用 git push 上傳所有的更新作業

當然，Git 還有很多指令可以使用，而且也有許多的觀念如果可以事先建立的話，在專案的開發上會更加地得心應手，請讀者務必參考相關的線上資源多瞭解一些分散式版本控制的觀念與實務。

基本架構

- manage.py
  - 管理納站組態，所有命令都是執行此程式，平常不會修改
- myblog 與專案同名，放專案設定檔
- urls.py 對應每個網址要對應的函數及方式，建新網頁先編輯
- wsgi.py 和其它伺服器溝通的介面
- settings.py 網站系統設計的所在位置，建新網站先編輯
  - 先把剛建立的mainsite APP加進去
- 用startapp mainsite建立app來運作

settings.py

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

然後再把檔案最後面的時區設定改一下，如下所示：

```
LANGUAGE_CODE = 'zh-Hant'
TIME_ZONE = 'Asia/Taipei'
```

建立資料庫

migrate

另外，在預設的情形下，Django 會使用 SQLite 來儲存資料庫的內容，在我們使用以下命令的時候，即會產生一個叫做 db.sqlite3 的檔案：

```
(VENV) minhuang@ubuntu:~/myDjango/mblog$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, contenttypes, auth, sessions
Running migrations:
  Rendering model states... DONE
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying sessions.0001_initial... OK
```

python manage.py migrate

之後，所有在此網站中新增到資料庫的資料，都會被放在 db.sqlite3 這個檔案中，這是一個簡化過的檔案型 SQL 關聯式資料庫系統。如果要搬移網站的時候，也記得要把這個檔案帶上。

資料庫與 Django 的關係

在預設的情況下，Django 的資料庫是以 Model 的方式來操作，也就是在程式中不直接面對資料庫以及資料表，而是以 class 類別的方式先建立出 Model，然後再透過對 Model 的操作，達到操作資料庫的目的。這樣的好處是把程式和資料庫之間的關係以一層中介層來做為連接的介面，日後如需要更換資料庫系統，就可以不需要更動到程式的內容。

也是因為這樣，對於第一次接觸到此種方式的網站開發者而言會不太直覺，不去直接定義資料庫中的資料表，而是以定義一個資料類別來當作是資料表，在定義了資料類別之後，還要再執行一些指令讓這個資料表的每一個資料欄位之名稱、格式、屬性可以和資料類別中的內容同步，的確是有些麻煩。但是，此種方式如果習慣了之後，你會發現，其實這是把資料庫連接層加以抽象化的好方法，為程式開發人員省去了花在各種不同資料庫操作細節的精力和時間。

因此，簡單地看，在 Django 要使用資料庫，有以下的幾個步驟：

1. 在 models.py 中定義所需要使用的類別（繼承自 models.Model）
2. 詳細地設定每一個在類別中的變數，亦即資料表中的每一個欄位
3. 使用 python manage.py makemigrations mainsite 建立資料庫和 Django 間的中介檔案
4. 使用 python manage.py migrate 同步更新資料庫的內容
5. 在程式中使用 Python 的語法操作所定義的資料類別，等於是在操作資料庫中的資料表

定義資料模型

在本堂課中的 mblog 需要一個用來儲存文章的資料表，所以，就需要修改 mainsite/models.py 的內容。一開始，models.py 的內容如下：

```
from django.db import models
# Create your models here.
```

修改為如下所示的內容：

```
from django.db import models
from django.utils import timezone
# Create your models here. 注意縮排

class Post(models.Model):
    title = models.CharField(max_length=200)
    slug = models.CharField(max_length=200)
    body = models.TextField()
    pub_date = models.DateTimeField(default=timezone.now)

    class Meta:
        ordering = ('-pub_date',)

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

補充

example

- __str__可用物件實體的名稱表示一段字串

```
#補充
class strtest:
    def __init__(self):
        print("init: this is only test")
    def __str__(self): #回傳必須是字串
        return "str: this is only test"

if __name__ == "__main__":
    st=strtest()
    print(st)
# init: this is only test
# str: this is only test
```

pub_date 是以 timezone.now 的方式讓它自動產生，這還需要一個 pytz 模組才行。請執行以下的指令安裝：

```
pip install pytz
```

pip install pytz

- 使模型生效：

python manage.py makemigrations mainsite

python manage.py migrate

要讓此模型生效，需執行以下指令：

```
(VENV) minhuang@ubuntu:~/myDjango/mblog$ python manage.py makemigrations
mainsite
Migrations for 'mainsite':
  0001_initial.py:
    - Create model Post
(VENV) minhuang@ubuntu:~/myDjango/mblog$ python manage.py migrate
Operations to perform:
  Synchronize unmigrated apps: staticfiles, messages
  Apply all migrations: admin, contenttypes, mainsite, auth, sessions
Synchronizing apps without migrations:
  Creating tables...
    Running deferred SQL...
  Installing custom SQL...
Running migrations:
  Rendering model states... DONE
  Applying mainsite.0001_initial... OK
```

此時即可在程式中直接操作此資料庫了。但是為了方便起見，請進入下一節，啟用Django提供的admin介面來操作，會更加地方便。

建立第一位使用者

- 超級管理員

啟用admin管理介面

admin 是 Django 預設的資料庫內容管理介面，在使用之前，有幾個要設定的步驟，第一步，是建立一個管理者的帳號以及密碼，如下所示：

```
(VENV) minhuang@ubuntu:~/myDjango/mblog$ python manage.py createsuperuser
Username (leave blank to use 'minhuang'): admin
Email address: ho@minhuang.net
Password:
Password (again):
Superuser created successfully.
```

- python manage.py createsuperuser

接著，要把上一小節定義的 Post 納入管理，請修改 mainsite/admin.py，原本是如下所示的內容：

```
from django.contrib import admin

# Register your models here.
```

登入admin

- http://127.0.0.1:8000/admin/


Django 管理
歡迎， ADMIN. 檢視網站 / 變更密碼 / 登出
網站管理
認證與授權
使用者
新增
變更
群組
新增
變更
最近的動作
我的動作
無可用的

admin套用po文模組 post

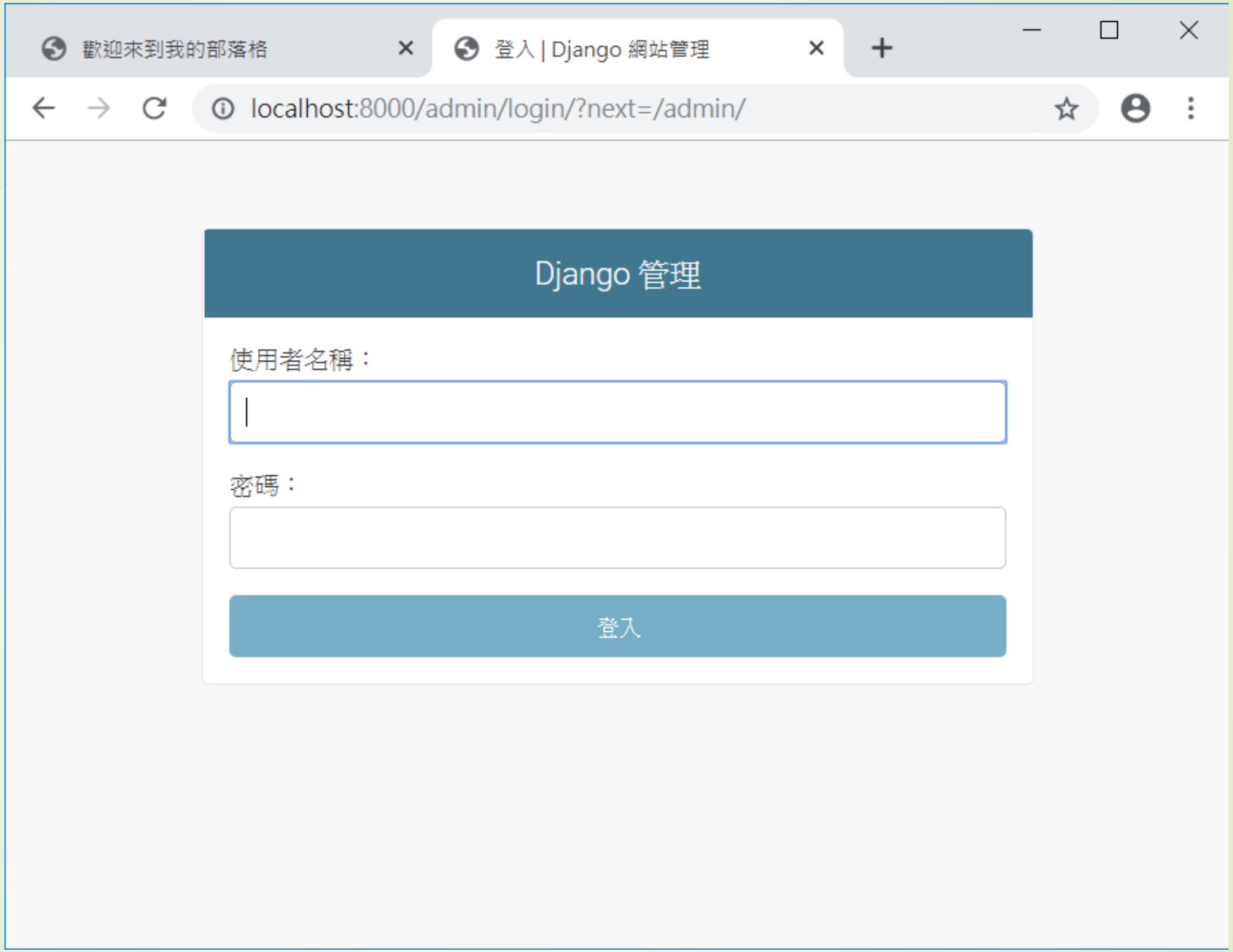
請改為如下所示的內容：

```
from django.contrib import admin
from .models import Post

# Register your models here.

admin.site.register(Post)
```

也就是先引入我們的 Post 類別，然後再透過 admin.site.register 註冊即可。完成以上設定之後，再次啟用此網站，然後透過瀏覽器連結到 http://localhost:8000/admin，就可以看到如圖 2-2 所示的登入畫面。



在輸入之前設定過的 superuser 帳號以及密碼之後，即可以看到一個美觀的資料庫


歡迎來到我的部落格
網站管理 | Django 網站管理
localhost:8000/admin/
Django 管理
歡迎， ADMIN. 檢視網站 / 變更密碼 / 登出
網站管理
MAINSITE
Posts
新增
變更
認證與授權
使用者
新增
變更
群組
新增
變更

如圖 2-3 的箭頭所指的地方，我們定義的 Post 也被順利地納入管理介面了。第一次點擊到 Posts 中時，還沒有任何內容，如圖 2-4 所示。

圖 2-4：第一次進入 Posts 的管理頁面


歡迎來到我的部落格
變更 post | Django 網站管理
localhost:8000/admin/mainsite/post/2/change/
Django 管理
歡迎， ADMIN. 檢視網站 / 變更密碼 / 登出
首頁 › Mainsite › Posts › 妙蛙種子
變更 post
歷史
Title：
妙蛙種子
Slug：
001
Body：
經常可見牠在太陽下睡午覺的樣子。在沐浴了充足的陽光之後，牠背上的種子就會成長茁壯。
Pub date：
日期
2019/11/09
今天
時間
14:30
現在
刪除
儲存並新增另一個
儲存並繼續編輯
儲存

在圖 2-5 中按下「儲存」按鈕，即可看到如圖 2-6 所示的畫面，該文章已被順利新增到資料庫中了。

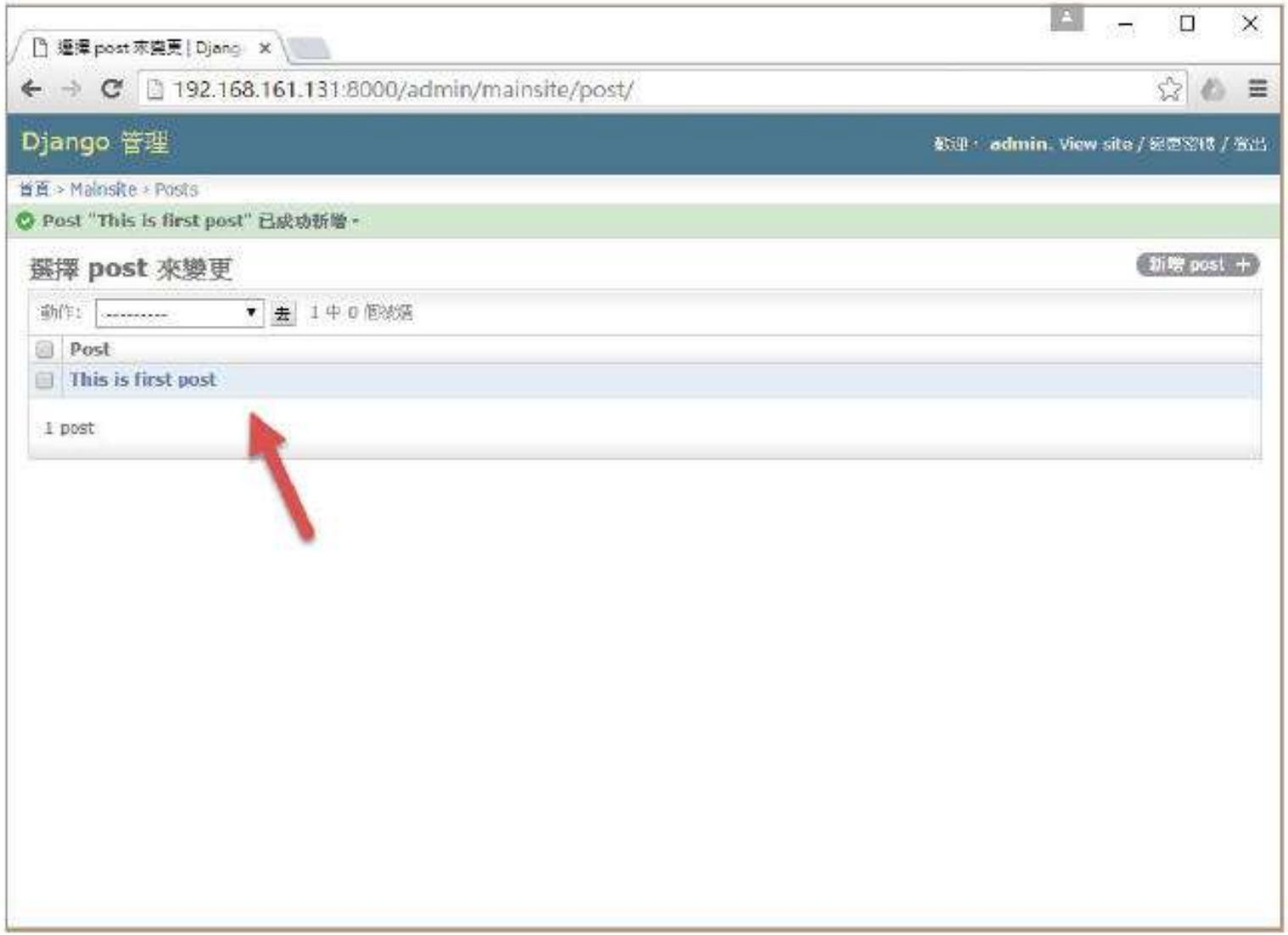
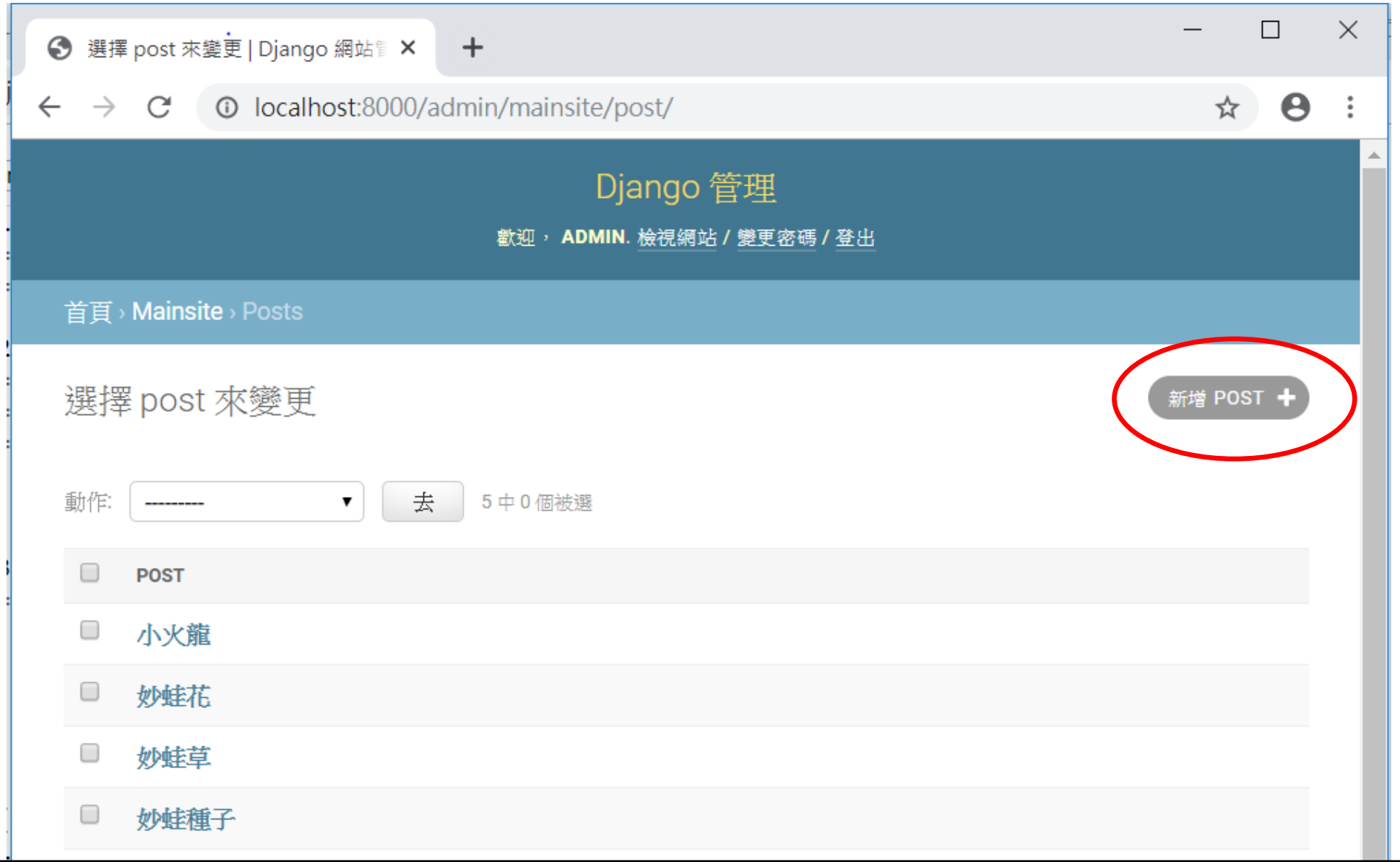


圖 2-6：已新增文章之後的 Posts 操作畫面

為了後續測試方便，請至少輸入 5 篇文章內容，中英文皆可，但是 slug 請使用英文或數字即可，而且中間不要使用任何符號以及空白字元。如圖 2-7 所示。



修改様式

而在本小節的最後，在 admin.py 中加入以下的程式碼（自訂 Post 顯示的方式之類別，繼承自 admin.ModelAdmin），讓文章在顯示的時候，除了 title 之外，還可以再加上張貼的日期和時間等內容：

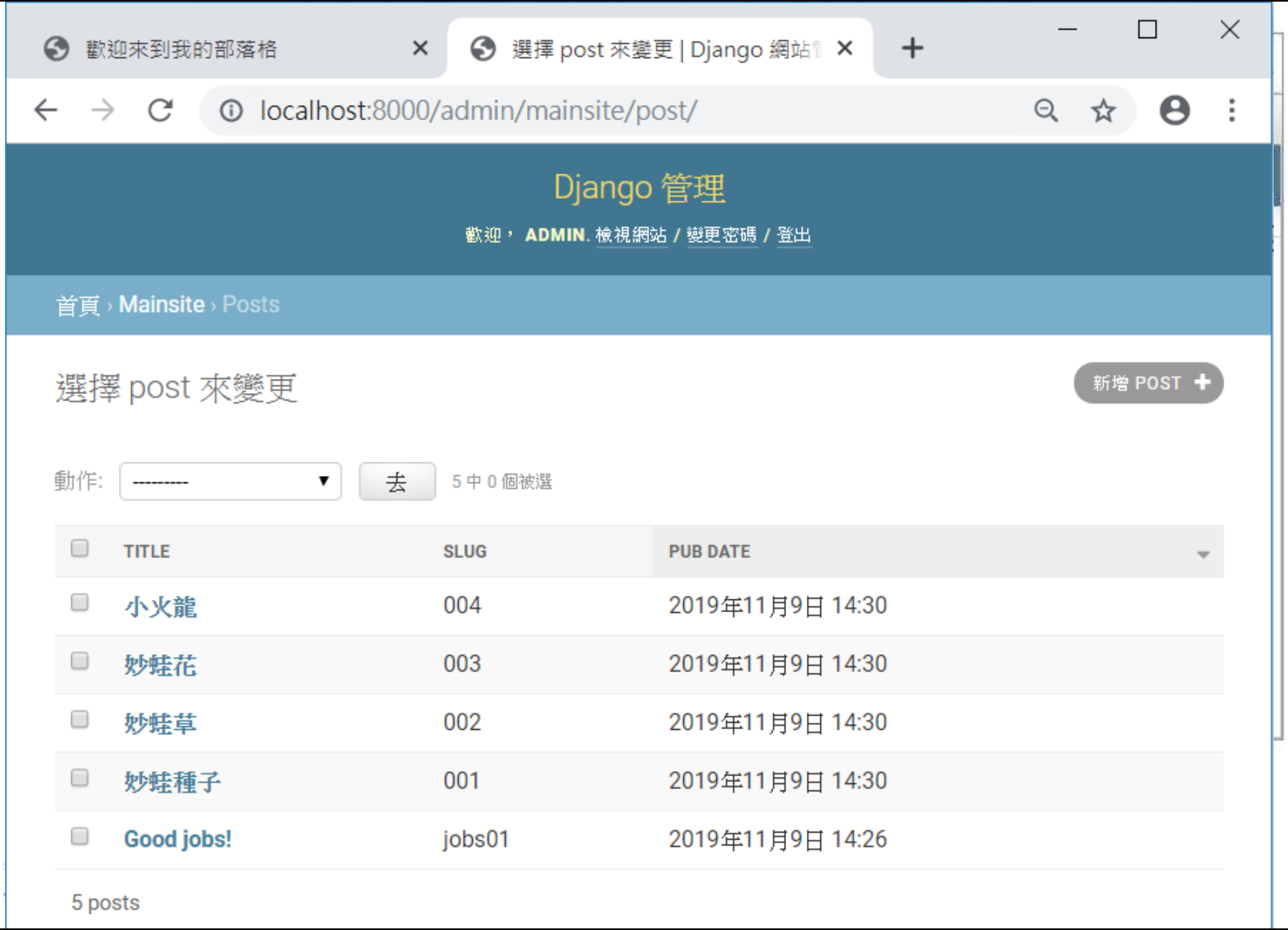
```
from django.contrib import admin
from .models import Post

# Register your models here.

class PostAdmin(admin.ModelAdmin):
    list_display = ('title','slug','pub_date')

admin.site.register(Post, PostAdmin)
```

則在顯示所有文章的時候，就變成如圖 2-8 所示的樣子。


歡迎來到我的部落格
選擇 post 來變更 | Django 網站管
localhost:8000/admin/mainsite/post/
Django 管理
歡迎， ADMIN. 檢視網站 / 變更密碼 / 登出
首頁 › Mainsite › Posts
選擇 post 來變更
新增 POST +
動作: --------- 去 5 中 0 個被選
TITLE SLUG PUB DATE
小火龍 004 2019年11月9日 14:30
妙蛙花 003 2019年11月9日 14:30
妙蛙草 002 2019年11月9日 14:30
妙蛙種子 001 2019年11月9日 14:30
Good jobs! jobs01 2019年11月9日 14:26
5 posts

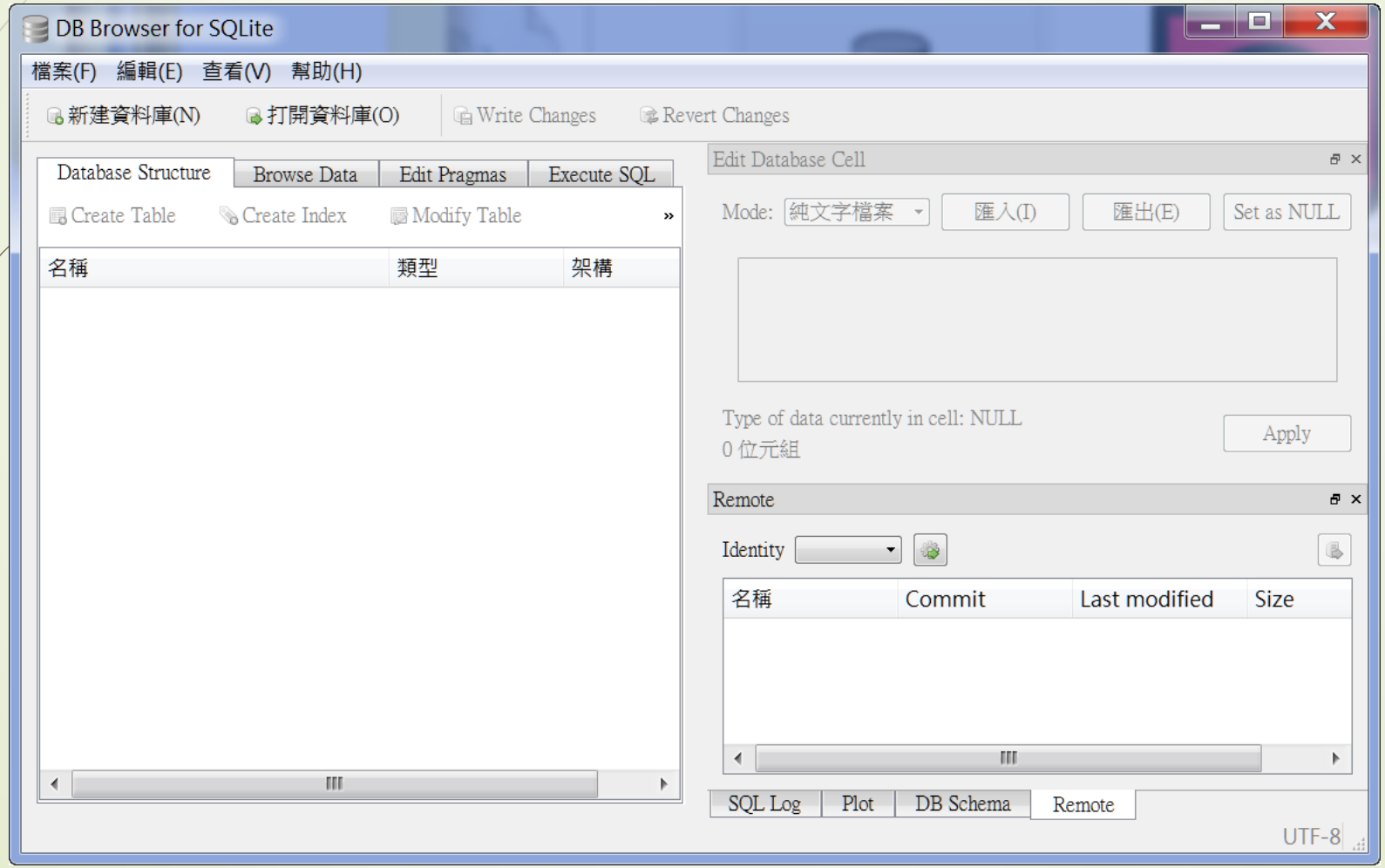
小練習

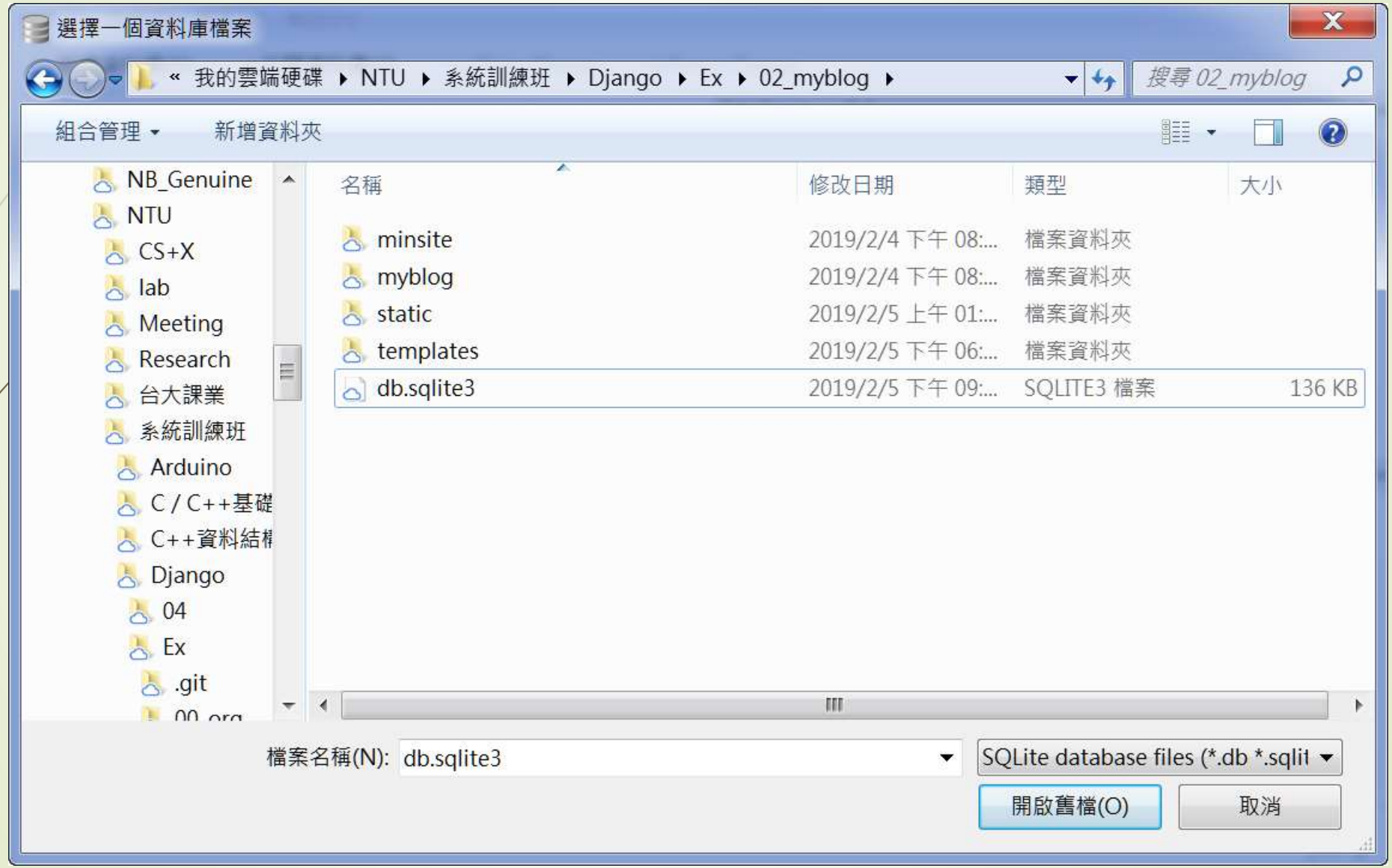
- 請試著輸入5篇文章 (slg請暫時用英文或數字)
- 完成後請用 git commit 本地備份
- 再上傳到遠端儲存倉

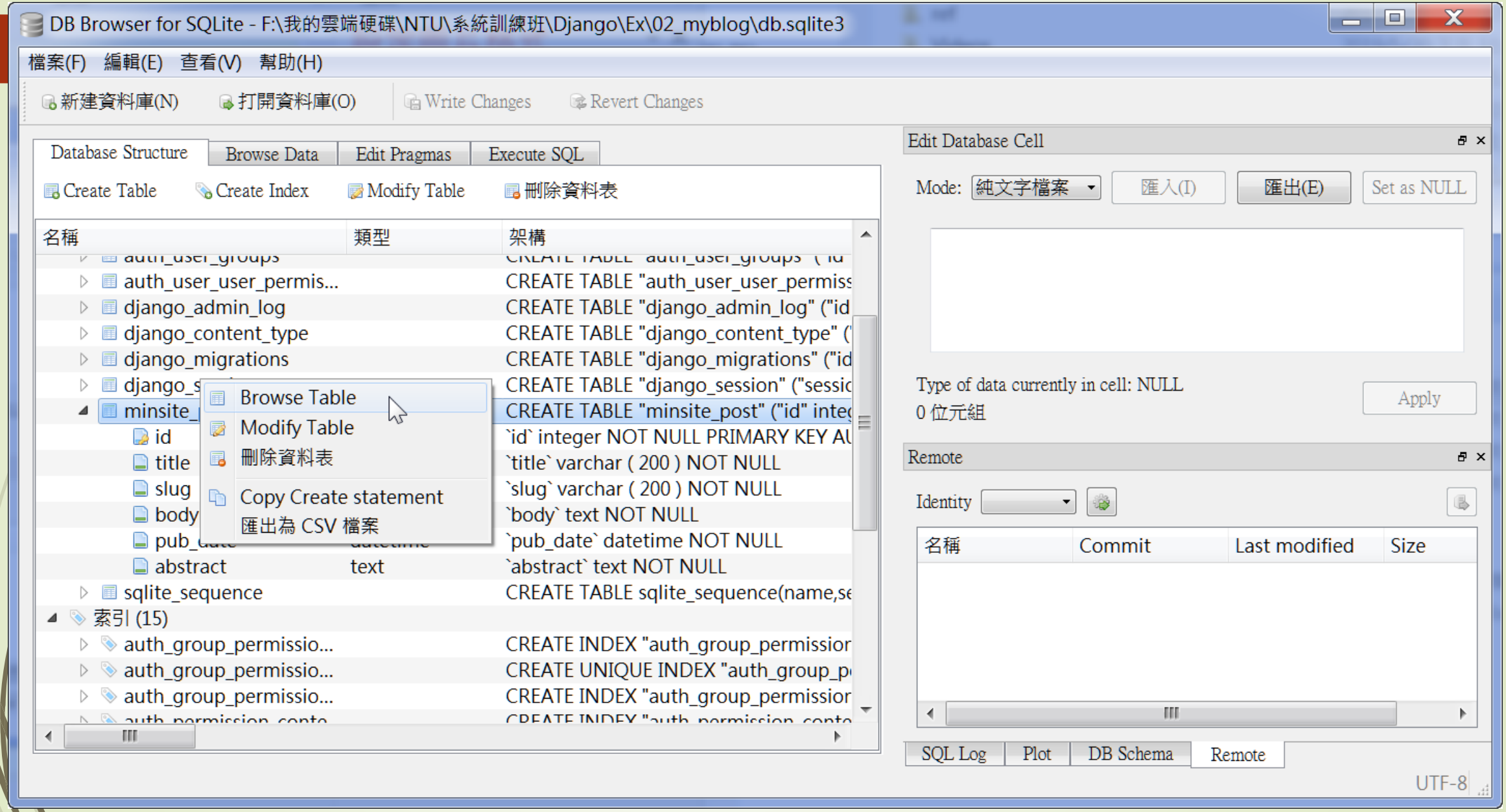
看看資料庫裡有什麼？

察看 sqlite的內容

- DBBrowserforSQLite https://sqlitebrowser.org/dl/




選擇一個資料庫檔案
« 我的雲端硬碟 ▸ NTU ▸ 系統訓練班 ▸ Django ▸ Ex ▸ 02_myblog ▸
搜尋 02_myblog
組合管理
新增資料夾
NB_Genuine
NTU
CS+X
lab
Meeting
Research
台大課業
系統訓練班
Arduino
C / C++基礎
C++資料結構
Django
04
Ex
.git
名稱
修改日期
類型
大小
minsite 2019/2/4 下午 08:... 檔案資料夾
myblog 2019/2/4 下午 08:... 檔案資料夾
static 2019/2/5 上午 01:... 檔案資料夾
templates 2019/2/5 下午 06:... 檔案資料夾
db.sqlite3 2019/2/5 下午 09:... SQLITE3 檔案 136 KB
檔案名稱(N): db.sqlite3
SQLite database files (*.db *.sqlit
開啟舊檔(O)
取消


DB Browser for SQLite - F:\我的雲端硬碟\NTU\系統訓練班\Django\Ex\02_myblog\db.sqlite3
檔案(F) 編輯(E) 查看(V) 幫助(H)
新建資料庫(N)
打開資料庫(O)
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Create Table
Create Index
Modify Table
刪除資料表
名稱
類型
架構
auth_user_user_permis...
CREATE TABLE "auth_user_user_permiss
django_admin_log
CREATE TABLE "django_admin_log" ("id
django_content_type
CREATE TABLE "django_content_type" (
django_migrations
CREATE TABLE "django_migrations" ("id
CREATE TABLE "django_session" ("sessic
minsite_
CREATE TABLE "minsite_post" ("id" inte
Browse Table
Modify Table
刪除資料表
Copy Create statement
匯出為 CSV 檔案
id
`id` integer NOT NULL PRIMARY KEY AU
title
`title` varchar (200) NOT NULL
slug
`slug` varchar (200) NOT NULL
body
`body` text NOT NULL
pub_date
`pub_date` datetime NOT NULL
abstract
text
`abstract` text NOT NULL
sqlite_sequence
CREATE TABLE sqlite_sequence(name,se
索引 (15)
auth_group_permissio...
CREATE INDEX "auth_group_permissior
auth_group_permissio...
CREATE UNIQUE INDEX "auth_group_p
auth_group_permissio...
CREATE INDEX "auth_group_permissior
Edit Database Cell
Mode: 純文字檔案
匯入(I)
匯出(E)
Set as NULL
Type of data currently in cell: NULL
0 位元組
Apply
Remote
Identity
名稱
Commit
Last modified
Size
SQL Log
Plot
DB Schema
Remote
UTF-8

- 只能編寫，不能讀取？

讀取資料庫中的內容

資料庫中有了文章內容之後，接下來是要讀取這些資料，然後在網站的首頁中把它們顯示出來的時候了。在此先簡單說明一下 Django 的 MTV（大約可以類比到 MVC）架構。為了把資料抽象化，Django 把資料的存取和顯示區分為 Model、Template、以及 View，分別對應到 models.py、template 資料夾、以及 views.py 這些檔案。

如前幾個小節做的事情，models.py 中主要負責定義要存取的資料模型，以 Python 的 class 類別方式來定義，在後端 Django 自動會為我們把這個類別中的設定對應到資料庫系統中，不管你使用的是哪一種資料庫。而如果把這些資料拿出來，或是要如何存進去的這些程式邏輯，則是在 View 中，也就是在 views.py 中處理，而這也是我們在這一小節中要編寫程式的地方。至於如何把這些取得的資料用美觀有彈性的方式輸出，則是在 Template 中加以處理，這是下一節的內容。

在此，先開啟 mainsite/view.py，一開始預設的內容如下：

```
from django.shortcuts import render

# Create your views here.
```

第一步要先把在 models.py 中自定義的 Model 引入，然後就可以使用 Post.objects.all() 取得所有的資料項目，針對此資料夾，即可利用 for 迴圈取出所有的內容，再透過 HttpResposne 輸出到網頁中，如下所示：

```
from django.shortcuts import render
from django.http import HttpResponse
from .models import Post
# Create your views here.

def homepage(request):
    posts = Post.objects.all()
    post_lists = list()
    for count, post in enumerate(posts):
        post_lists.append("No.{}:".format(str(count)) +str(post)+"<br>")
    return HttpResponse(post_lists)
```

在此例中，我們建立了一個叫做 homepage 的函數用來取得所有文章，並透過迴圈把它們搜集到一個變數 post_lists 中，最後再使用 return HttpResponse(post_lists) 把這個變數的內容，輸出到使用者端的瀏覽器畫面中。

但是，有了這個函數，要由誰來呼叫它呢？也就是使用者在瀏覽器中要透過哪一個網址才能夠執行到這個函數呢？答案是，要透過 urls.py 來負責網址和程序間的對應工作，不然的話，光是瀏覽網頁的根路徑，還是只會得到如圖 2-1 的畫面。請開啟 urls.py，分別引入來自於 views.py 的 homepage 函數並以 url 對應之，如下所示：

```
from django.conf.urls import include, url
from django.contrib import admin
from minsite.views import homepage

urlpatterns = [
    url(r'^$', homepage),
    url(r'^admin/', admin.site.urls),
]
```

string前面加上 'r'
表示是個raw string，不要轉意backslash '\'
由於正則表示式和 \ 會有衝突
使用了正則表示式後，最好在前面加上'r'

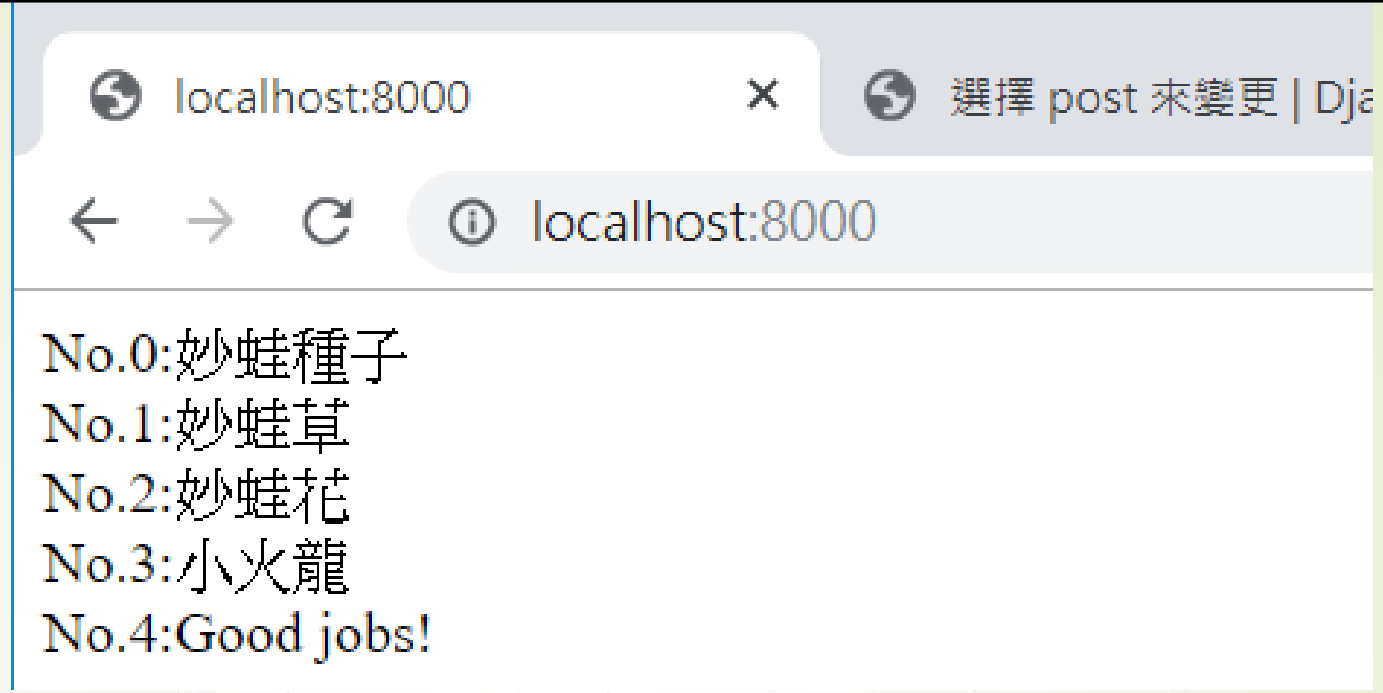
其中，url(r'^$', homepage) 這一行中，「^」表示字串開頭處，「$」表示字串結尾處，兩者接在一起的意思，就是指出，當有使用者瀏覽了網址而沒有加上任何字串的時候（即為根網址），就去呼叫 homepage 這個函數，就可以得到如圖 2-9 所示的執行畫面。

但是，有了這個函數，要由誰來呼叫它呢？也就是使用者在瀏覽器中要透過哪一個網址才能夠執行到這個函數呢？答案是，要透過 urls.py 來負責網址和程序間的對應工作，不然的話，光是瀏覽網頁的根路徑，還是只會得到如圖 2-1 的畫面。請開啟 urls.py，分別引入來自於 views.py 的 homepage 函數並以 url 對應之，如下所示：

```
from django.urls import include, path
from django.contrib import admin
from mainsite.views import homepage

urlpatterns = [
        path('admin/', admin.site.urls),
        path('', homepage),
]
```

其中，url(r'^$', homepage) 這一行中，「^」表示字串開頭處，「$」表示字串結尾處，兩者接在一起的意思，就是指出，當有使用者瀏覽了網址而沒有加上任何字串的時候（即為根網址），就去呼叫 homepage 這個函數，就可以得到如圖 2-9 所示的執行畫面。



除了文章的標題之外，我們也可以取出每一篇文章的內容，並使用 HTML 標記為顯示出的內容排版，讓畫面比較美觀一些，homepage 可以修改如下：

```
def homepage(request):
    posts = Post.objects.all()
    post_lists = list()
    for count, post in enumerate(posts):
        post_lists.append("No.{}:".format(str(count)) + str(post)+"<br>")
        post_lists.append("<small>"+str(post.body)+"</small><br><br>")

    return HttpResponse(post_lists)
```

修改之後的程式，瀏覽的畫面如圖 2-10 所示。

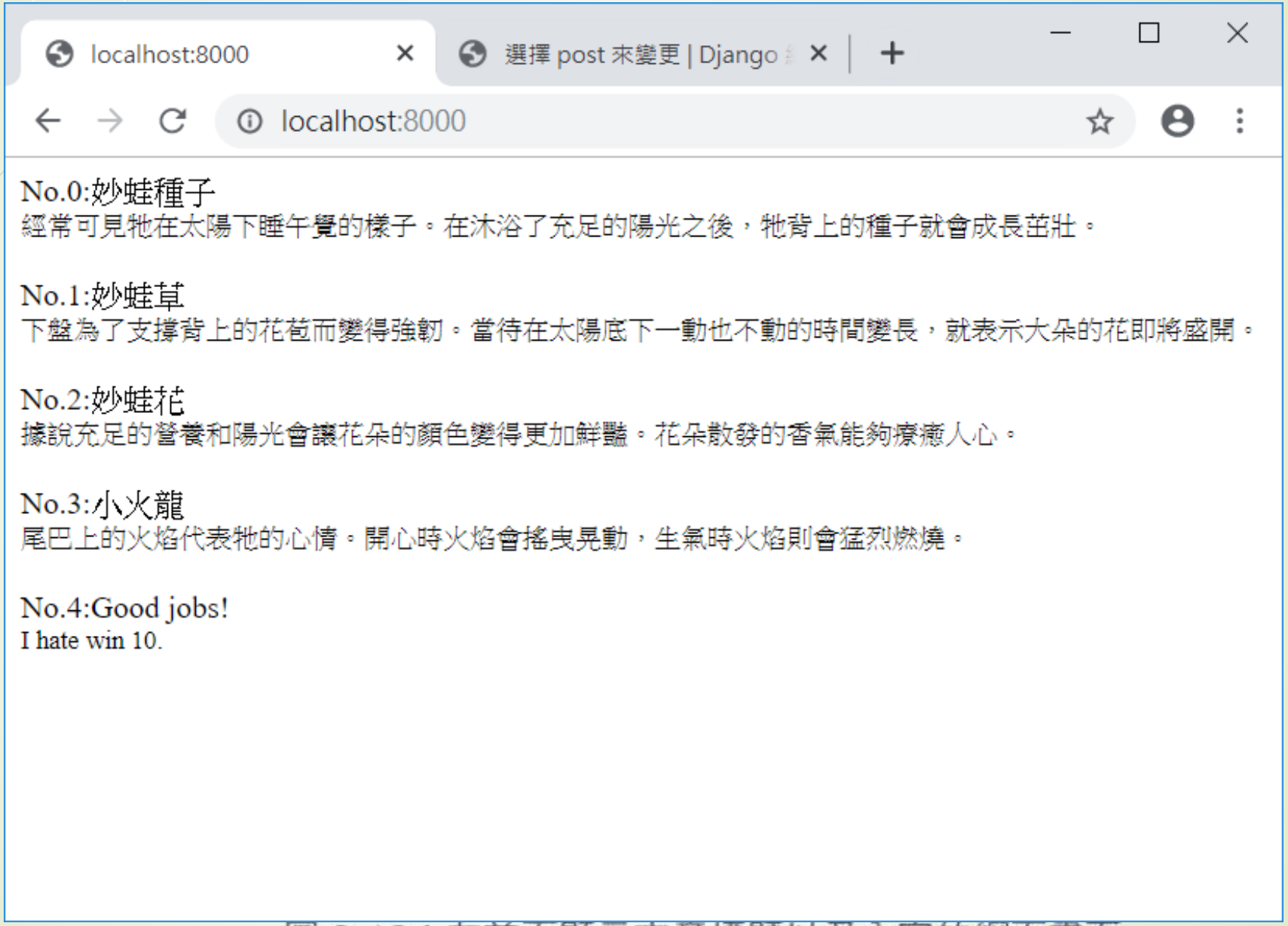


圖 2-10：在首頁顯示文章標題以及內容的網頁畫面

- 好像..... 有點.....醜醜的？
- 美化它

網址對應與頁面輸出

建立網頁輸出模版 template

在前一節中我們示範了如何建立資料模型、在 admin 介面中輸入及編輯資料，以及如何使用 Post.objects.all() 取出所有的資料並透過 HttpResponse 輸出到瀏覽器端。那如何把這些拿到的資料，再排版一下，變成比較美觀的樣子呢？答案就是模版 template。每一個輸出的網頁都可以準備一個或一個以上對應的模版，而這些模版是以 .html 的檔案型式儲存在指定的資料夾中（一般都會命名為 templates），當網站有資料需要輸出的時候，再透過渲染函數（render）把資料放到模版指定的位置中，得到結果後再交給 HttpResponse 輸出給瀏覽器。基本的步驟如下所示：

Steps：

1. 在 setting.py 中設定模版資料夾的位置
2. 在 urls.py 建立網址和 views.py 中的函數對應的關係
3. 建立 .html 檔案（例如 index.html），做好排版並安排資料要放置的位置
4. 執行程式，以 objects.all() 在 views.py 取得資料，並放入變數中，例如 posts
5. 以 render 函數，把資料（例如 posts）送到指定的模版檔案（例如 index.html）中

以上的第一個步驟，在本堂課的例子中需先在此專案的目錄中建立 templates 這個資料夾，建立完畢後之目錄結構如下：

```
├── db.sqlite3
├── mainsite
│   ├── admin.py
│   ├── __init__.py
│   ├── migrations
│   │   ├── 0001_initial.py
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── manage.py
├── mblog
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── requirements.txt
└── templates
```

然後把此資料夾名稱加到 settings.py 的 TEMPLATES 區塊中，如下所示（只要修改 DIRS 那一行即可）：

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

接著，我們打算把 posts 和 now（現在時刻）丟到模版（打算叫做 index.html）中顯示，所以把 views.py 重新修改如下：

```python
from django.template.loader import get_template
from django.http import HttpResponse
from datetime import datetime
from .models import Post
# Create your views here.
def homepage(request):
    template = get_template('index.html')
    posts = Post.objects.all()
    now = datetime.now()
    html = template.render(locals())
    return HttpResponse(html)
```

在這裡我們用了一個小技巧把變數丟到模版，就是使用 locals() 這個函數。這個函數會把目前記憶體中的所有區域變數使用字典型態打包起來，剛好可以在這裡派上用場，在模版中因為接受到了所有的區域變數，當然也可以把 posts 和 now 都拿來使用。 Locals()

在 templates 目錄下，建立一個名為 index.html 的模版檔案，如下所示：

```
<html>
<head>
      <meta charset='utf-8'>
      <title>歡迎光臨我的部落格</title>
</head>
<body>
      <h1>歡迎光臨我的部落格</h1>
      <hr>
      {{posts}}
      <hr>
      <h3>現在時間：{{now}}</h3>
</body>
</html>
```

存檔之後並執行網站測試，再一次瀏覽網站時，即可看到如圖 2-11 所示的網頁畫面。


歡迎光臨我的部落格
localhost:8000
歡迎光臨我的部落格
<QuerySet [<Post: 巴大蝴>, <Post: 皮卡丘>, <Post: 妙蛙種子>, <Post: 小火龍>]>
現在時間：2020年2月9日 13:31

什麼是HTML?

什麼是HTML？

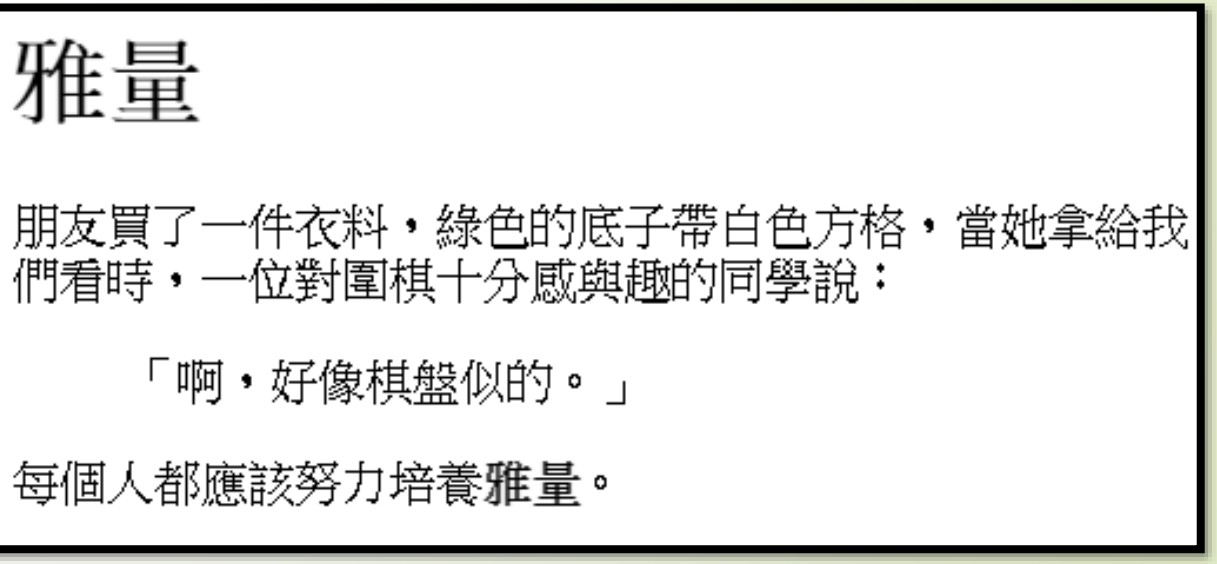
- HyperText Markup Language**超文件標示語言**
- 建立網頁的標準標示語言

HTML ▾

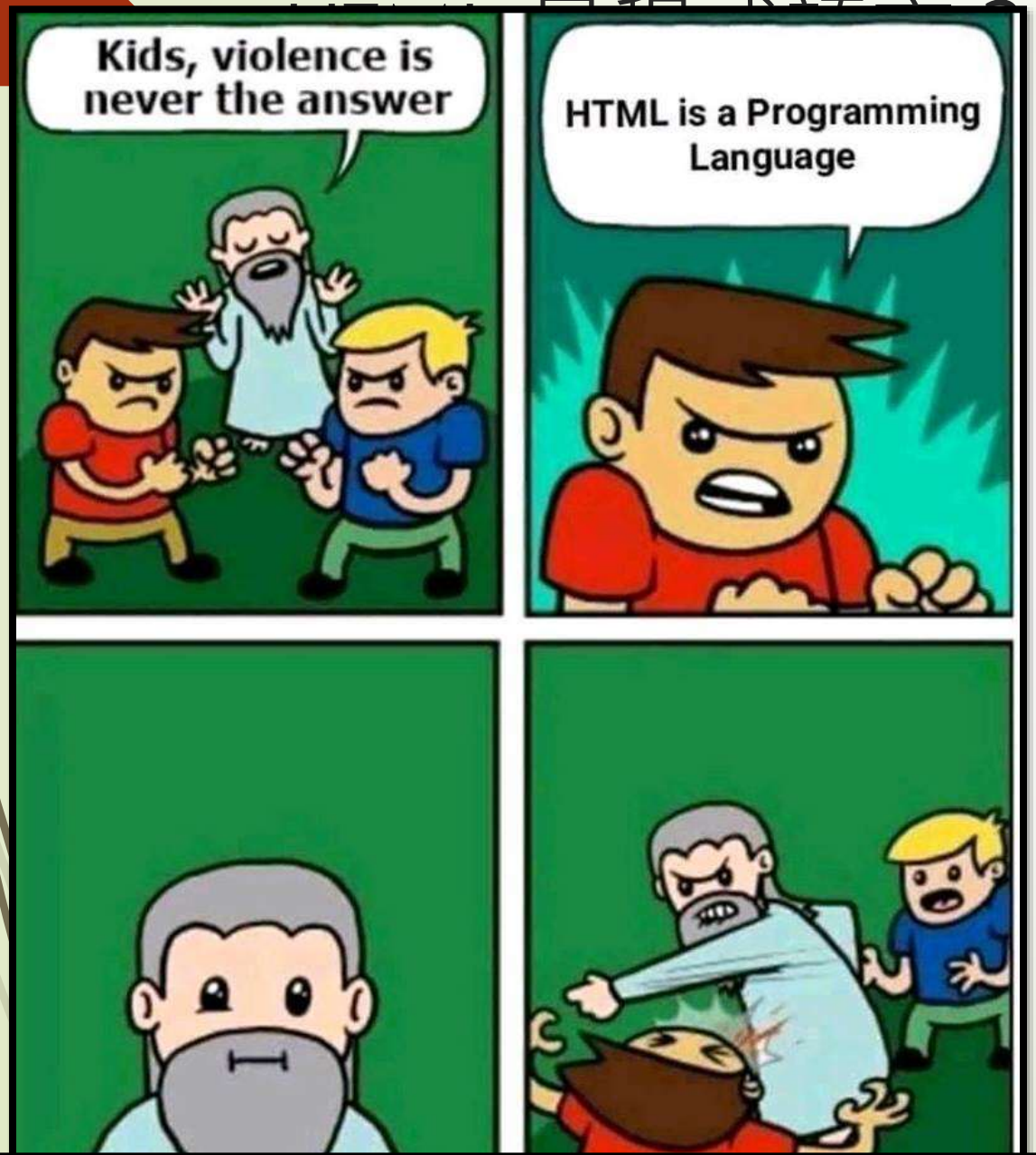
```
<h1>雅量</h1>

<p>朋友買了一件衣料，綠色的底子帶白色方格，當她拿給我們看時，一位對圍棋十分感興趣的同學說：</p>
<blockquote>「啊，好像棋盤似的。」</blockquote>

<p>每個人都應該努力培養<strong>雅量</strong>。</p>
```



HTML是程式語言嗎？


Kids, violence is never the answer
HTML is a Programming Language


Sir, this seat is reserved for people with disabilities
I am a Programmer.
My favourite language is HTML
Sorry to bother you, sir

標籤 Tag

用小於 **<** 和大於 **>** 兩個符號所包住的文字，我們稱為標籤

- 像是 <html>、<body>，html 和 body 我們稱為標籤的名稱。
- 標籤通常成對存在，有開始標籤和結束標籤。
- 夾在開始和結束標籤中的內容稱為標籤的內文(inner HTML or inner Text)。

This is bold text
<u>這是有底線的文字</u>

This is bold text
這是有底線的文字

- 有些element僅由單一標籤構成，如

*<input><hr><img>
*...等。

標籤的內文

- 標籤的內文可以是**純文字**或者包含**其他標籤**。

Example

```
<body><b>Hello World</b></body>
```

我們說 body 標籤包含 b 標籤。
b 標籤中包含純文字串 Hello World

HTML 文件的基本要素

- 以下為基本 HTML 文件的格式，所有網頁都建議套用此格式。

*<!– html 文件最上層**必須**是 html 標籤 -->*

```
<!doctype html>
<html>
    <head>
        <meta charset="big5" />
        <title>Arduino家庭自動化</title>
    </head>
    <body>
    這是我的第一個網頁。
    </body>
</html>
```

提供網頁資訊給瀏覽器的檔頭區

網頁本體（內文區）

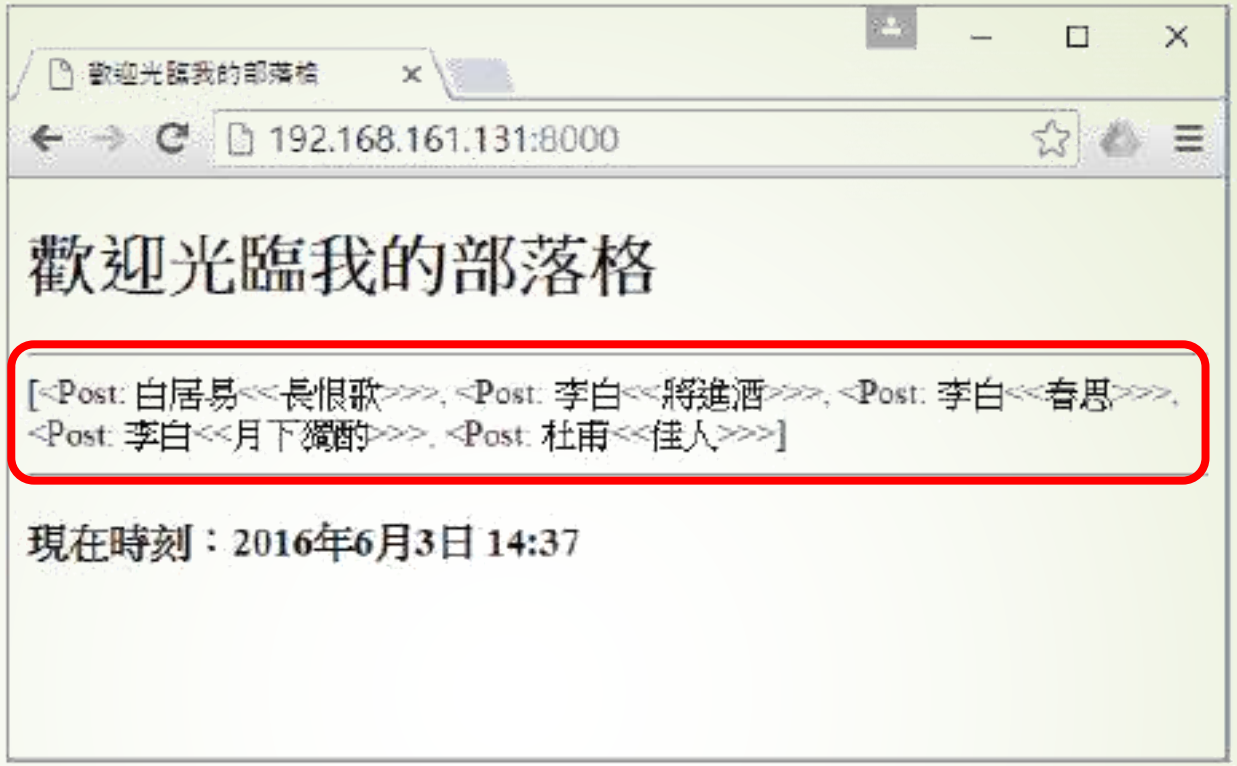


圖 2-11：加上 index.html 模版的網頁

由 index.html 的內容可以看得出來，HTML 的標籤和傳統的 HTML 檔案無異，但是多了「{{ }}」大括號用來輸出收到的資料。now 這個資料指的就是現在時刻，顯示出來的樣子不難理解。但是 posts 是一個完整的資料集，其中還包括許多的欄位和項目，顯示成圖 2-11 這樣子並不妥當。好在，其實在 template 也有一套模版語言可以用來在模版檔案中解析這些資料項目，在此先說明如何透過 for 迴圈把資料集中的項目一個一個取出使用。

```
<html>
<head>
    <meta charset="utf-8">
    <title>
        歡迎來到我的部落格
    </title>
</head>
<body>
    <h1> 歡迎來到我的部落格 </h1>
    <hr>
    {% for post in posts %}
        <p style='font-family: 微軟正黑體;font-size:16pt;font-weight:bold;'>
            {{post.title}}
        </p>
        <p style='font-family: 微軟正黑體;font-size:10pt;letter-spacing:1pt;'>
            {{post.body}}
        </p>
    {% endfor %}
    <hr>
    <h3>現在時間:{{now}}</h3>
</body>
</html>
```

由上述的程式可以看出，每一個資料項目的欄位，是以 post.body, post.title 的方式取出，而迴圈指令則是使用 {% for %} 和 {% endfor %} 成對使用。此外，還利用 CSS 的字型指令做些簡單的排版。在後續的章節中會說明如何運用 CSS 做更進一步的網頁版面安排。在內容的部份，把標題和內容分開顯示，網頁看起來就如圖 2-12 所示的樣子。

小練習

- 增加貼文發佈時間

- 顯示所有的東西很奇怪？
- 應該只顯示標題，點進去有內容才對

不過，一般來說網站的首頁是不會把所有的內容都顯示出來，而應該是先把標題顯示出來，但是在每一個標題上製作連結，當瀏覽者點擊連結的時候才會開啟另外一個頁面顯示出該篇文章的內容。因此，我們接著把 index.html 進一步修改如下：

```
<html>
<head>
    <meta charset="utf-8">
    <title>
        歡迎來到我的部落格
    </title>
</head>
<body>
    <h1> 歡迎來到我的部落格 </h1>
    <hr>
    {% for post in posts %}
        <p style='font-family: 微軟正黑體;font-size:16pt;font-weight:bold;'>
            <a href='/post/{{post.slug}}'>{{post.title}}</a>
        </p>
   {% endfor %}
   <hr>
   <h3>現在時間:{{now}}</h3>
</body>
</html>
```

影像與超連結標籤

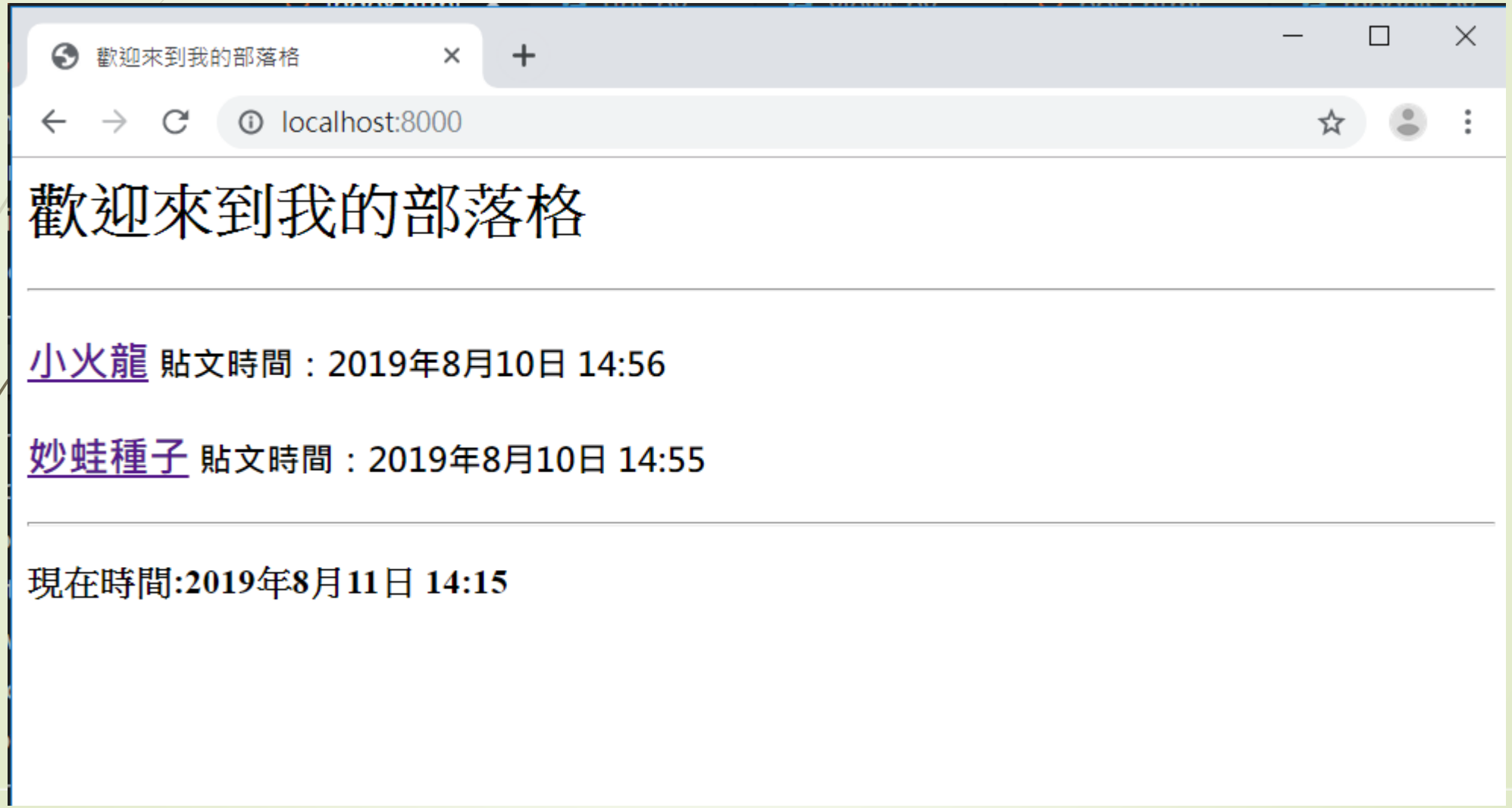
- HTML img 圖片標籤
- <img src='圖片 URL' border='圖片邊框' ....>
- Ex: <img src='http://img00.deviantart.net/f040/i/2009/167/6/2/hilltop_view_at_night_by_relhom.jpg' height='300px'>

HTML img 圖片參數

參數	用法說明
src	圖片網址，必要項目。
border	圖片邊框，例如 border="0" 代表邊框為 0。
alt	圖片替代文字，當圖片顯示失效，則顯示 alt 文字。
title	圖片文字標示，當滑鼠移經圖片，自動顯示的文字。
width	圖片寬度，例如 width="120px" 代表寬度限制在 120px。
height	圖片高度，例如 height="100px" 代表高度限制在 100px。

- 超連結
- 顯示文字
- Ex: Google

透過 <a href> 這個 HTML 標籤取出 post.slug 建立為連結網址，並放在 post/ 之下，執行的結果如圖 2-13 所示。



小練習

- 增加貼文發佈時間
- 請試著在index.html
  網頁中加入你喜歡的圖片
- 完成後請用 git commit 本地備份
- 再上傳到遠端儲存倉

網址？

網址對應 urls.py

注意到圖 2-13 箭頭所指的地方，就是點擊了任一篇文章標題的時候，瀏覽器會傳遞給網站的網址。以此例是 http://192.168.161.131:8000/post/baigui01。其中，/post/ 是我們加在 index.html 中為顯示單篇文章用的前置詞，而後面的 baigui01 則是在建立文章時，我們自行設定的自訂網址。也就是說，要辨識出這些網址以對應到要顯示的單篇文章內容，應該有如下的幾個步驟：

1. 在 urls.py 設定，只要是 /post/ 開頭的網址，就把後面接著的文字當作是參數傳遞 slug 給 post_detail 這個顯示單篇文章的函數
2. 在 views.py 中新增一個 post_detail 函數，除了接收 request 參數之外，亦接收 slug 這個參數
3. 在 templates 資料夾中建立一個用來顯示單篇文章用的 post.html
4. 在 post_detail 函數中，以 slug 為關鍵字，搜尋資料集，找出是否有符合的項目
5. 如果有符合，則把找到的資料項目傳遞給 render 函數，找出 post.html 這個模版頁出來渲染，再把結果交給 HttpResponse 回傳給瀏覽器
6. 如果沒有符合的項目，再把網頁轉回首頁

urls.py (舊式寫法)

在網址的對應方面，需做如下的修改：

```
from django.contrib import admin
from django.conf.urls import include, url
from minsite.views import homepage, showpost

urlpatterns = [
    url(r'^$', homepage),
    url(r'^post/(\w+)$', showpost),
    #url(r'^post/(.*)$', showpost),
    url(r'^admin/', admin.site.urls),
]
```

透過 url(r'^post/(\w+)$', showpost) 這行的設定，把所有 post/ 開頭的網址後面的字串都找出來，當作是第 2 個參數（第 1 個是預設的 request）傳遞給 showpost 這個函數，而 showpost 則是在 import 的地方要記得引入，同時到 views.py 中新建這個函數來處理接收到的參數，如下所示：(views.py)

urls.py

在網址的對應方面，需做如下的修改：

```
from django.urls import include, path
from django.contrib import admin
from mainsite.views import homepage, showpost

urlpatterns = [
        path('admin/', admin.site.urls),
        path('', homepage),
        path('post/<slug:slug>/', showpost),
]
```

透過 url(r'^post/(\w+)$', showpost) 這行的設定，把所有 post/ 開頭的網址後面的字串都找出來，當作是第 2 個參數（第 1 個是預設的 request）傳遞給 showpost 這個函數，而 showpost 則是在 import 的地方要記得引入，同時到 views.py 中新建這個函數來處理接收到的參數，如下所示：(views.py)

views.py

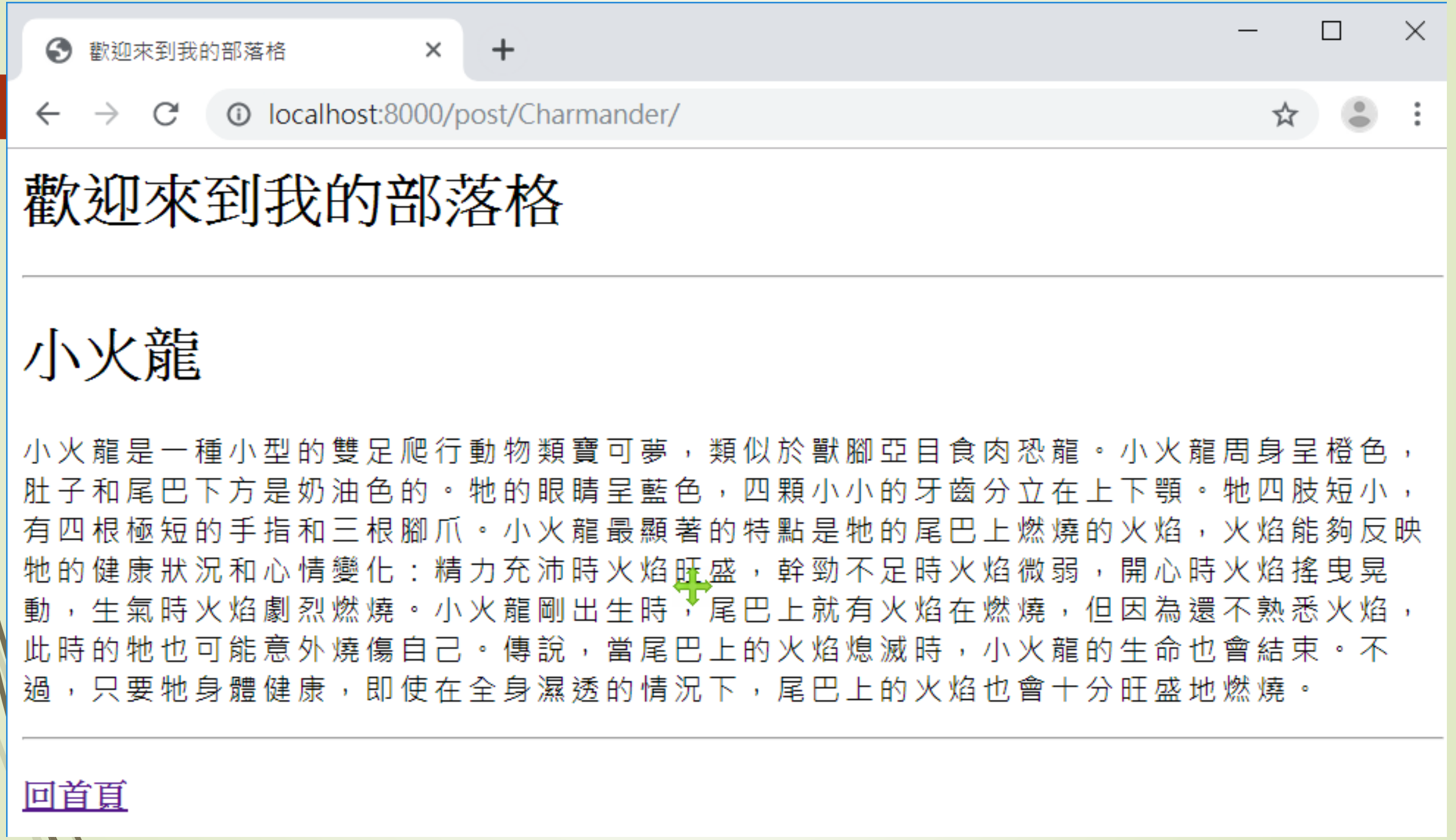
```
from django.template.loader import get_template
from django.http import HttpResponse
from django.shortcuts import redirect
from datetime import datetime
from .models import Post

# …中間略…

def showpost(request, slug):
    template = get_template('post.html')
    try:
        post = Post.objects.get(slug=slug)
        if post != None:
            html = template.render(locals())
            return HttpResponse(html)
    except:
        return redirect('/')
```

考慮到有可能會有自行輸入錯誤網址以至於找不到文章的情形，除了在以Post.objects.get(slug = slug)搜尋文章時加上例外處理，也在發生例外的時候以redirect('/')的方式直接返回首頁，因此也不要忘了需在前面引入redirect模組。至於顯示文章的post.html內容如下：

```
<html>
<head>
    <meta charset="utf-8">
    <title>
        歡迎來到我的部落格
    </title>
</head>
<body>
    <h1> 歡迎來到我的部落格 </h1>
    <hr>
    <h1>{{post.title}} </h1>
    <p style='font-family: 微軟正黑體;font-size:12pt;letter-spacing:2pt;'>
        {{post.body}}
    </p>
    <hr>
    <h3><a href='/'>回首頁</a></h3>
</body>
</html>
```


歡迎來到我的部落格
localhost:8000/post/Charmander/
歡迎來到我的部落格
小火龍
小火龍是一種小型的雙足爬行動物類寶可夢，類似於獸腳亞目食肉恐龍。小火龍周身呈橙色，肚子和尾巴下方是奶油色的。牠的眼睛呈藍色，四顆小小的牙齒分立在上下顎。牠四肢短小，有四根極短的手指和三根腳爪。小火龍最顯著的特點是牠的尾巴上燃燒的火焰，火焰能夠反映牠的健康狀況和心情變化：精力充沛時火焰旺盛，幹勁不足時火焰微弱，開心時火焰搖曳晃動，生氣時火焰劇烈燃燒。小火龍剛出生時，尾巴上就有火焰在燃燒，但因為還不熟悉火焰，此時的牠也可能意外燒傷自己。傳說，當尾巴上的火焰熄滅時，小火龍的生命也會結束。不過，只要牠身體健康，即使在全身濕透的情況下，尾巴上的火焰也會十分旺盛地燃燒。
回首頁

小練習

- 增加貼文發佈時間
- 請試著在post.html
  網頁中加入你喜歡的圖片
- 完成後請用 git commit 本地備份
- 再上傳到遠端儲存倉

歡迎來到我的部落格

localhost:8000/post/Charmander/

歡迎來到我的部落格

小火龍

貼文時間：2019年8月10日 14:56

小火龍是一種小型的雙足爬行動物類寶可夢，類似於獸腳亞目食肉恐龍。小火龍周身呈橙色，肚子和尾巴下方是奶油色的。牠的眼睛呈藍色，四顆小小的牙齒分立在上下顎。牠四肢短小，有四根極短的手指和三根腳爪。小火龍最顯著的特點是牠的尾巴上燃燒的火焰，火焰能夠反映牠的健康狀況和心情變化：精力充沛時火焰旺盛，幹勁不足時火焰微弱，開心時火焰搖曳晃動，生氣時火焰劇烈燃燒。小火龍剛出生時，尾巴上就有火焰在燃燒，但因為還不熟悉火焰，此時的牠也可能意外燒傷自己。傳說，當尾巴上的火焰熄滅時，小火龍的生命也會結束。不過，只要牠身體健康，即使在全身濕透的情況下，尾巴上的火焰也會十分旺盛地

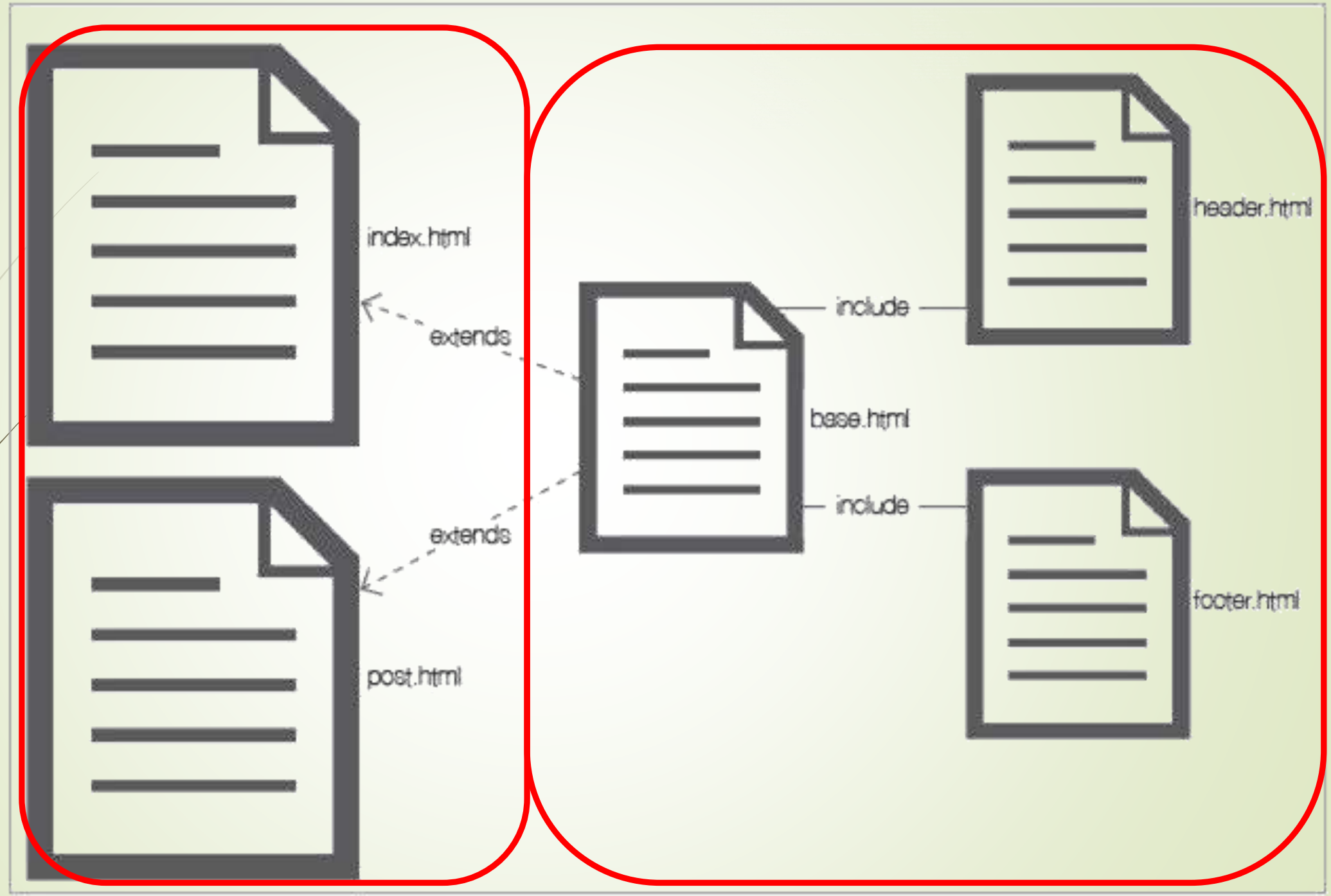
- 只有單一功能不夠
- 切割網站
- 將每個部份獨立出來
- 把網站做得更大更完整

共用模版的使用

- 幾乎所有的商用網站在每一頁都會有一些共同的元素
- 以強調網站的風格，
- 如果像上一小節那樣每一樣者分開設計的話，不僅要多花許多不必要的時間和精力，而且如果有所更動的時候，也很難同步修改到所有網頁共同的部份。
- 因此，把每一個網頁共同的部份獨立出來成為另外一個檔案，才是最正確的做法，而Django就提供了共同模版的方式處理這部份的機制。

共用模版的使用

檔案名稱	用途說明
base.html	網站的基礎模版，提供網站的主要設計外觀風格
header.html	網站中每一個網頁共用的標題元素，通常是放置網站Logo的地方
footer.html	網站中每一個網頁的共用頁尾，用來放置版權聲明或是其它參考資訊
index.html	此範例網站的首頁
post.html	此範例網站用來顯示單篇


index.html
extends
post.html
extends
base.html
include
header.html
include
footer.html

如圖 2-15 所示的樣子，設計一個 base.html 的主要基礎模版，在此 base.html 中引入 header.html 和 footer.html，等於是讓 header.html 和 footer.html 可以分開設計。而我們主要呈現的檔案 index.html 以及 post.html 則用 extends 指令繼承自 base.html，以保持一致的網頁風格。

先看看 base.html 的內容：

```
<!-- base.html -->
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>
        {% block title %} {% endblock %}
    </title>
</head>
<body>
    {% include 'header.html' %}
    {% block headmessage %} {% endblock %}
    <hr>
    {% block content %} {% endblock %}
    <hr>
    {% include 'footer.html' %}
</body>
</html>
```

在此例中，就是一般的 HTML 檔案，再加上 {% … %} 的模版指令，在這些模版指令中，沒有意外，使用 include '.html 檔案名稱' 即可引入指定的模版檔案，此檔案中分別在適當的地點引入了 header.html 和 footer.html。此外，透過 block 指令可以在後面加上此 block（區塊）的名稱，也就是到時候在 index.html 中要填入內容的位置。在此 base.html 中，分別在適當的地點指定了 title, headmessage 和 content。因為在 base.html 指定了 3 個區塊，所以接下來繼承此 base.html 的任何檔案，一定要提供這 3 個區塊的內容才行。先來看看 index.html 的內容：

```
<!-- index.html -->
{% extends 'base.html' %} 繼承基礎模板
{% block title %} 歡迎光臨我的部落格-首頁 {% endblock %}
{% block headmessage %}
<h3 style='font-family:標楷體;'>本站文章列表
{% endblock %}
{% block content %}
    {% for post in posts %}
        <p style='font-family:微軟正黑體;font-size:14pt;font-weight:bold;'>
        <a href='/post/{{post.slug}}'>{{ post.title }}</a>
        </p>
    {% endfor %}
{% endblock %}
```

如上所述，一開始以 {% extends 'base.html' %} 指定要繼承的檔案為 base.html，然後下方就以 {% block title %}{% endblock %} 分別指出 3 個區塊要填寫的內容，至於其他如 <html></html> 這些共用的標籤就不需要了，因為都已在 base.html 中出現過了。同理，post.html 也就簡單多了，如下所示：

```
<!-- post.html -->
{% extends 'base.html' %}
{% block title %} {{ post.title }} - 我的貼文 {% endblock %}
{% block headmessage %}
    <h3 style='font-family:微軟正黑體;'>{{ post.title }}</h3>
    <a style='font-family:微軟正黑體;' href='/'>回首頁</a>
{% endblock %}
{% block content %}
    <p style='font-family:微軟正黑體;font-size:12pt;letter-spacing:2pt;'>
        {{ post.body }}
    </p>
{% endblock %}
```

那 header.html 和 footer.html，也只要負責它們自己的部份就好了，header.html 如下所示：

```
<!-- header.html -->
<h1 style="font-family:微軟正黑體;">歡迎光臨 我的部落格</h1>
```

以下則是 footer.html：

```
<!-- footer.html -->
{% block footer %}
    {% if now %}
        <p style='font-family:微軟正黑體;'>現在時刻：{{ now }}</p>
    {% else %}
        <p style='font-family:微軟正黑體;'>本文內容取自網絡，如有侵權請來信通知下架...</p>
    {% endif %}
{% endblock %}
```

在 footer.html 中我們多使用了一個模版指令的技巧 {% if now %}，它是用來判斷 now 這個變數是否有內容的指令，如果有，就顯示現在時刻，如果沒有，就只顯示出版權聲明。主要的原因是，我們在 index.html 中設計有在頁尾顯示現在時刻，而在顯示單篇文章的 post.html 則不顯示現在時刻，因此就需要 if 指令以提供此功能。使用共同模版功能的網站，如圖 2-16 所示。

圖 2-16：套用模版的範例程式執行畫面

歡迎光臨我的部落格-首頁

localhost:8000

歡迎光臨 我的部落格

本站文章列表

巴大蝴

皮卡丘

妙蛙種子

小火龍

現在時刻：2020年2月9日 14:55

妙蛙種子 – 我的貼文

localhost:8000/post/001/

歡迎光臨 我的部落格

妙蛙種子

回首頁

妙蛙種子是一種四足的寶可夢，外表類似蟾蜍。牠有著鮮紅色的眼睛，白色的瞳孔和鞏膜，頭頂上長著一對耳狀凸起。牠的鼻子短而鈍，且有一張大嘴。當牠的嘴巴張開時，可以看到上顎的一對小而尖的牙齒。牠的皮膚呈藍綠色，並附有深綠色的斑紋。牠每條粗腿的腳掌上都分出三趾，且末端都長有白色的小爪子。這隻寶可夢最顯著的特點就是牠背後那個鱗莖狀的種子，牠與妙蛙種子之間存在著共生關係，從妙蛙種子出生起就與牠的身體一同生長了。 雖然妙蛙種子通常用四條腿行走，但是妙蛙種子可以抬高牠的後腿。當牠

- 雖然功能切割開了
- 但還是醜醜的？
- 套用圖型樣式

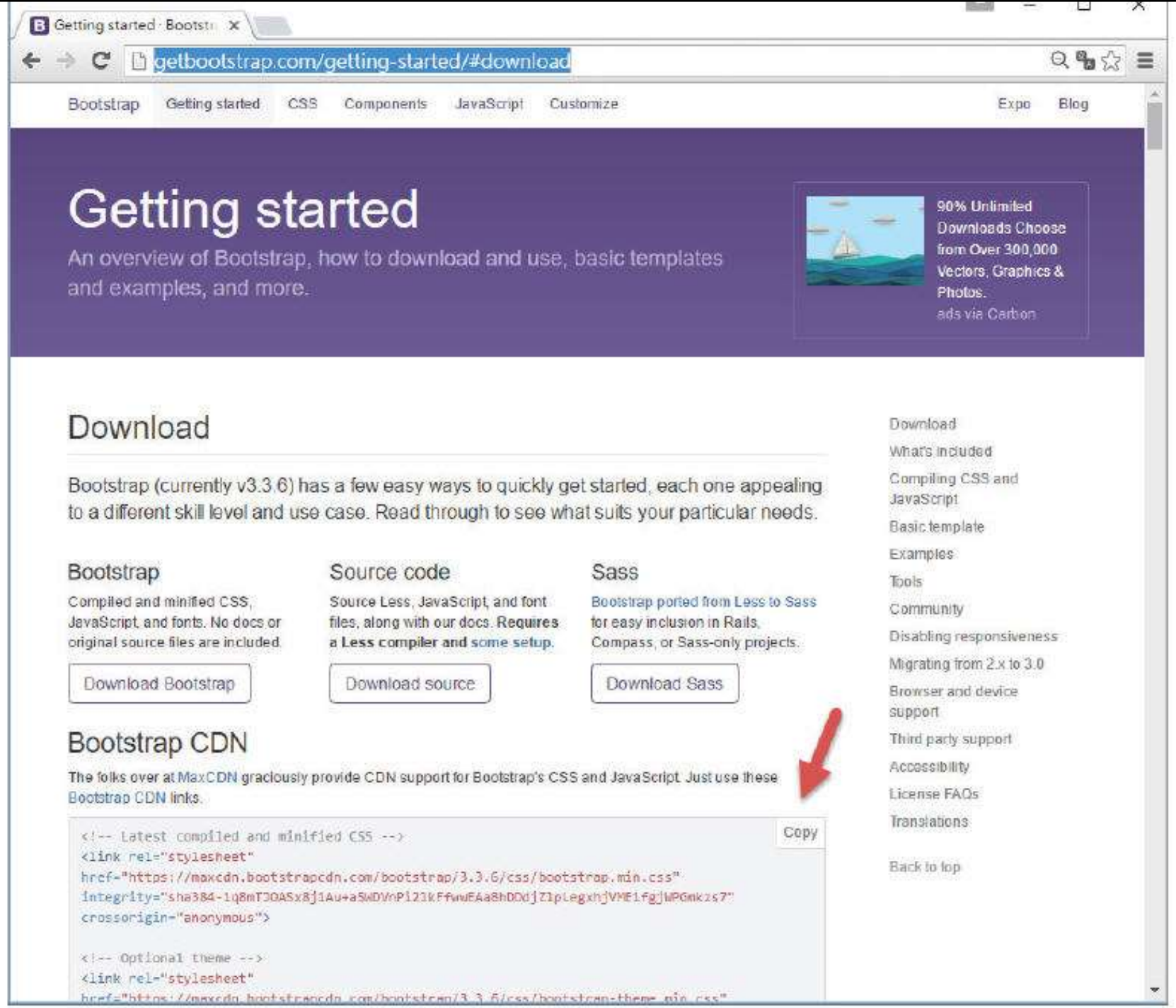
進階網站功能運用

- 一個成熟的部落格網站，除了前面所設計的功能之外，還需具備顯示圖形的能力，當然，首頁的設計也需要有版面的概念。此外，在文章內容的編排方面，如何提供編寫者具有簡易排版的能力，能在文章中設計版面、插入圖形以及建立連結等等，都是本節說明的重點。

Javascript 以及 CSS檔案的引用

本小節首先要討論的，是如何引用現有的 CSS 和 Javascript 網頁框架。HTML5 和 CSS3 以及 Javascript 的功能日趨複雜，一個網站不可能從無到有一點一點自行編輯設計，大部份都是使用一些現成的網頁框架，直接套用並加以修改後完成。免費的網頁框架種類不少，但還是以 Bootstrap 最受歡迎，而且使用也非常容易，可以選擇下載到本地端加以連結執行，或是直接利用 CDN 連結的方式套用即可，為了簡化步驟，在此使用後者。官方網址為：http://getbootstrap.com/getting-started/#download，畫面如圖 2-17 所示。

https://getbootstrap.com/docs/3.3/getting-started/#download


Getting started · Bootstr
getbootstrap.com/getting-started/#download
Bootstrap
Getting started
CSS
Components
JavaScript
Customize
Expo
Blog
Getting started
An overview of Bootstrap, how to download and use, basic templates and examples, and more.
90% Unlimited Downloads Choose from Over 300,000 Vectors, Graphics & Photos.
ads via Carbon
Download
Bootstrap (currently v3.3.6) has a few easy ways to quickly get started, each one appealing to a different skill level and use case. Read through to see what suits your particular needs.
Bootstrap
Compiled and minified CSS, JavaScript, and fonts. No docs or original source files are included.
Download Bootstrap
Source code
Source Less, JavaScript, and font files, along with our docs. Requires a Less compiler and some setup.
Download source
Sass
Bootstrap ported from Less to Sass for easy inclusion in Rails, Compass, or Sass-only projects.
Download Sass
Bootstrap CDN
The folks over at MaxCDN graciously provide CDN support for Bootstrap's CSS and JavaScript. Just use these Bootstrap CDN links.
Copy
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
crossorigin="anonymous">
<!-- Optional theme -->
<link rel="stylesheet"
Download
What's included
Compiling CSS and JavaScript
Basic template
Examples
Tools
Community
Disabling responsiveness
Migrating from 2.x to 3.0
Browser and device support
Third party support
Accessibility
License FAQs
Translations
Back to top

如圖 2-17 所示，按下箭頭所指的地方，即可把 CDN 的連結複製下來，然後放到 base.html 這個模版檔案中（請加在 </head> 的前面即可），接著在所有的模版檔案中就可以使用 Bootstrap 的功能。例如，可以在 header.html 中使用 well 這個大標題格式，如下所示：

```
<div class='well'>
        <h1 style="font-family:微軟正黑體;">歡迎光臨</h1>
</div>
```

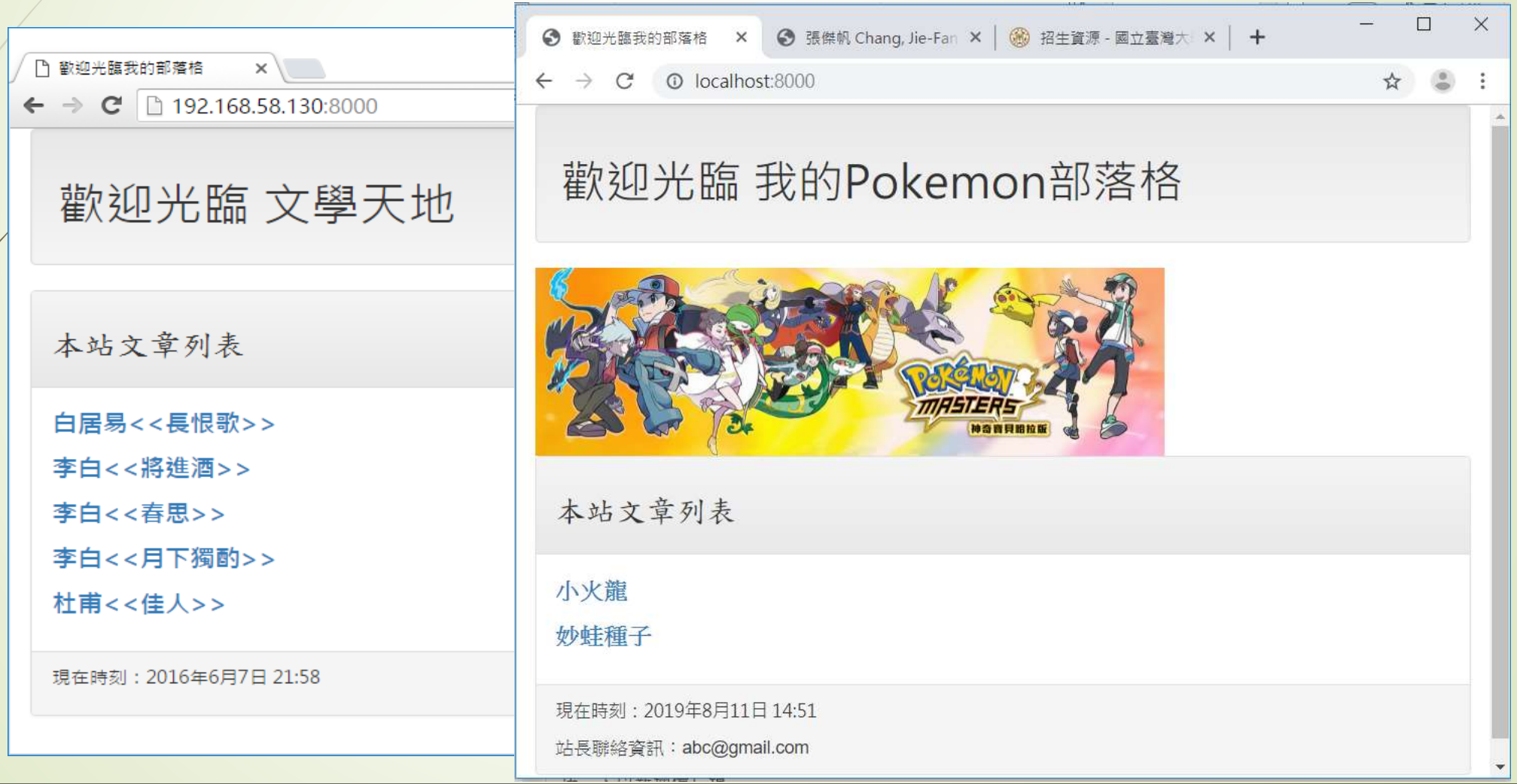
然後在 base.html 中使用 Panel 來安排首頁的外觀，如下所示：

```
<!-- base.html -->
<!DOCTYPE html>
{% load staticfiles %}
<html>
<head>
    <meta charset='utf-8'>
    <title>{% block title %} {% endblock %}</title>
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
integrity="sha384-
1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
crossorigin="anonymous">
```

```
<!-- Optional theme -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-
theme.min.css" integrity="sha384-fLW2N01lMqjakBkx3l/M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV+0En5r"
crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" integrity="sha384-
0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS" crossorigin="anonymous"></script>
</head>
<body>
    <div class='container-fluid'>
        {% include 'header.html' %}
        <div class='panel panel-default'>
            <div class='panel-heading'>
                {% block headmessage %} {% endblock %}
            </div>
            <div class='panel-body'>
                {% block content %} {% endblock %}
            </div>
            <div class='panel-footer'>
                {% include 'footer.html' %}
            </div>
        </div>
    </div>
</body>
</html>
```

套用Bootstrap後的結果



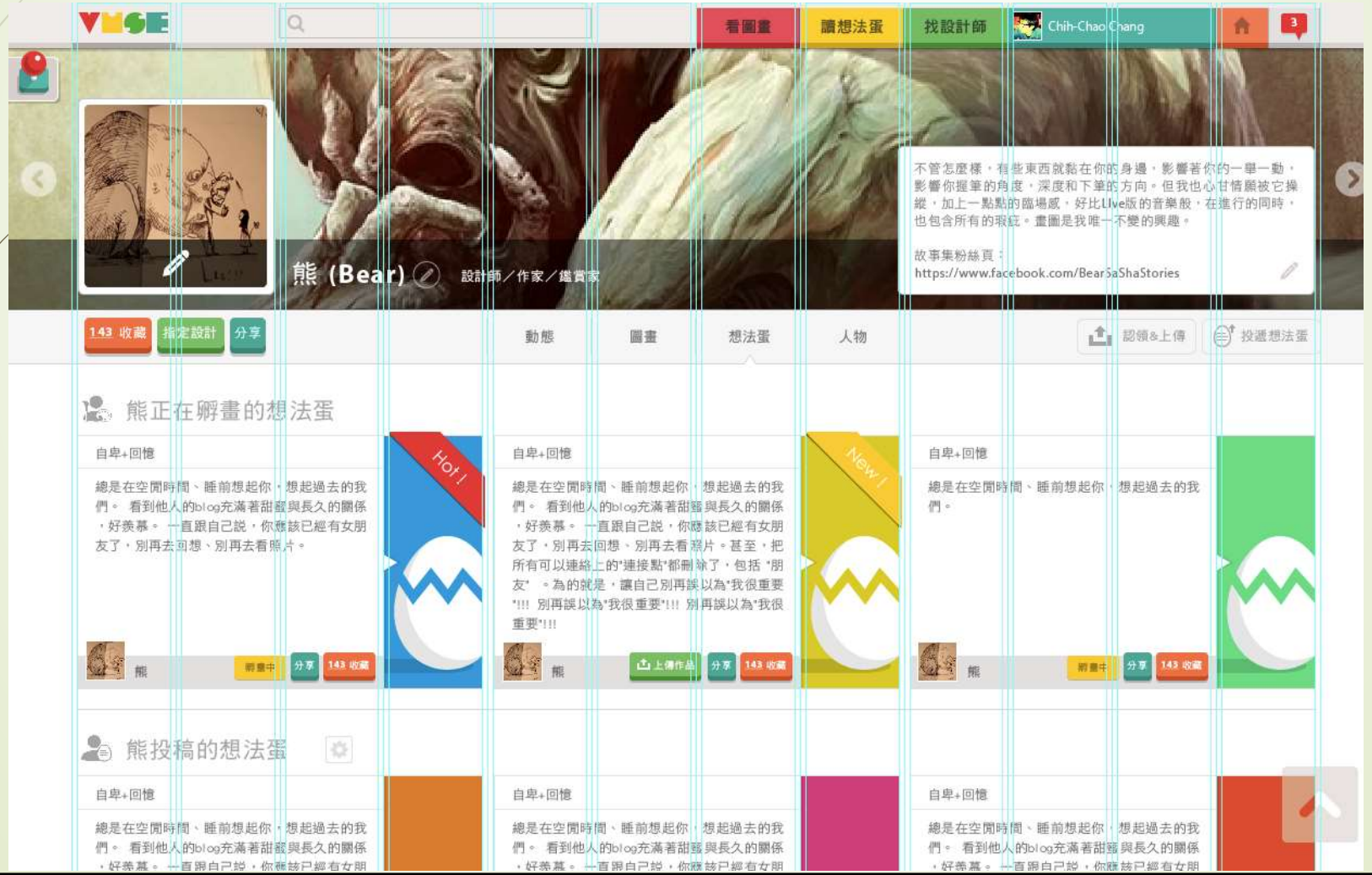
加入可自動排版的側邊欄

CSS Grid System

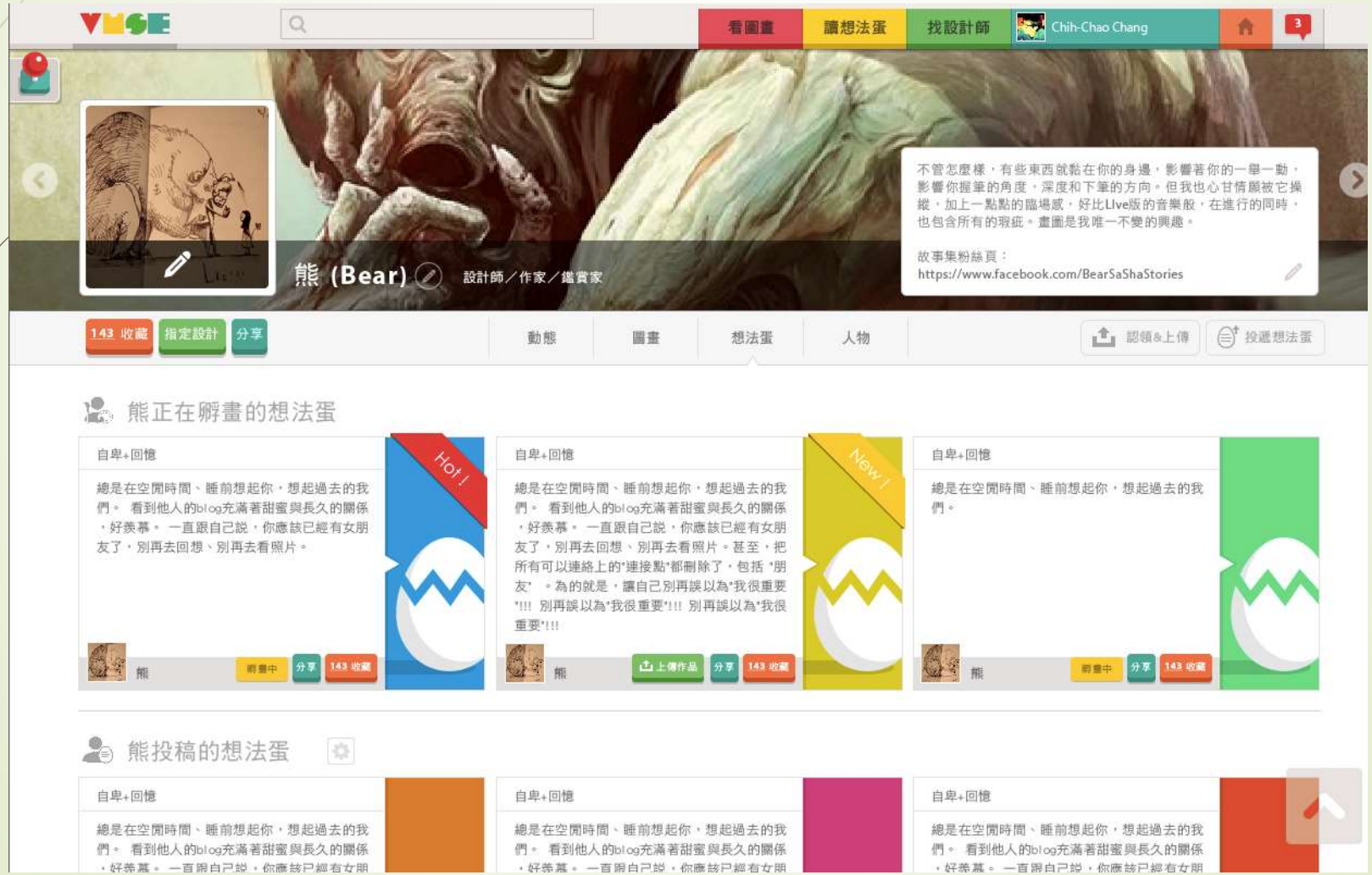
頁面排版

Grid System

12 欄網格設計


看圖畫
讀想法蛋
找設計師
Chih-Chao Chang
熊 (Bear)
設計師／作家／鑑賞家
不管怎麼樣，有些東西就黏在你的身邊，影響著你的一舉一動，影響你握筆的角度，深度和下筆的方向。但我也心甘情願被它操縱，加上一點點的臨場感，好比Live版的音樂般，在進行的同時，也包含所有的瑕疵。畫圖是我唯一不變的興趣。
故事集粉絲頁：
https://www.facebook.com/BearSaShaStories
143 收藏
指定設計
分享
動態
圖畫
想法蛋
人物
認領&上傳
投遞想法蛋
熊正在孵畫的想法蛋
自卑+回憶
總是在空閒時間、睡前想起你，想起過去的我們。 看到他人的blog充滿著甜蜜與長久的關係，好羨慕。 一直跟自己說，你應該已經有女朋友了，別再去回想、別再去看照片。
Hot !
New !
熊
分享
143 收藏
上傳作品
熊投稿的想法蛋

12 欄網格設計


看圖畫
讀想法蛋
找設計師
Chih-Chao Chang
熊 (Bear)
設計師／作家／鑑賞家
不管怎麼樣，有些東西就黏在你的身邊，影響著你的一舉一動，影響你握筆的角度，深度和下筆的方向。但我也心甘情願被它操縱，加上一點點的臨場感，好比Live版的音樂般，在進行的同時，也包含所有的瑕疵。畫圖是我唯一不變的興趣。
故事集粉絲頁：
https://www.facebook.com/BearSaShaStories
143 收藏
指定設計
分享
動態
圖畫
想法蛋
人物
認領&上傳
投遞想法蛋
熊正在孵畫的想法蛋
自卑+回憶
總是在空閒時間、睡前想起你，想起過去的我們。 看到他人的blog充滿著甜蜜與長久的關係，好羨慕。 一直跟自己說，你應該已經有女朋友了，別再去回想、別再去看照片。
Hot !
New !
熊
分享
143 收藏
熊投稿的想法蛋

md

螢幕寬度不大於 992px 時：垂直排列

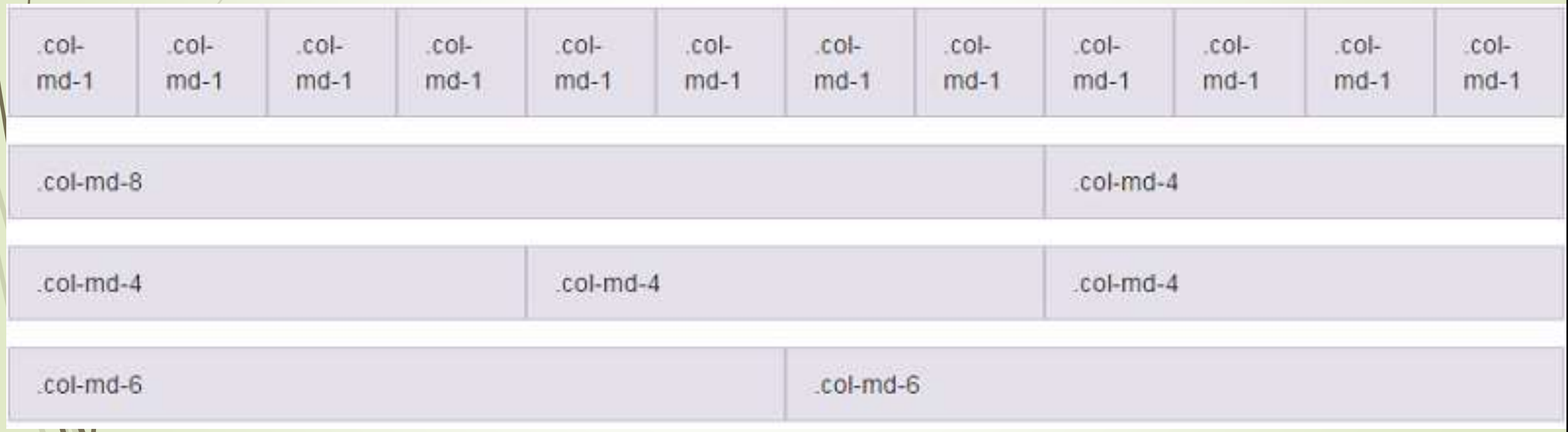
螢幕寬度大於 992px 時：水平排列

另有 .col-xs-*、.col-sm-*、.col-lg-*

Grid Options

- **Media Queries 的分段點**
- Mobile – xs (< 768px)
- Tablet – sm (768~991px)
- Desktop – md (992~1200px)
- Large Desktop - lg (>= 1200px)

- 因此我們可以依此做基本的設定，在 Desktop 的結果如下：



- 螢幕小於
  指定寬度就會：


.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-8
.col-md-4
.col-md-4
.col-md-4
.col-md-4
.col-md-6
.col-md-6

接著，可再引入 Bootstrap 的 Grid 觀念，透過 row 和 col 的設定，做出一般部落格網站側邊欄的效果。同樣地，也是在 base.html 中修改，如下所示（僅顯示 <body></body> 中的內容，計數器程式碼請至 feedjit.com 複製，勿重新輸入）：

滿版

```
<body>
    <div class='container-fluid'>
        {% include 'header.html' %}
        <div class='row'>
            <div class='col-sm-4 col-md-4'>
                <div class='panel panel-default'>
                    <div class='panel-heading'>
                        <h3>MENU</h3>
                    </div>
```

```
        <div class='panel-body'>
            <div class='list-group'>
                <a href='/' class='list-group-item'>HOME</a>
                <a href='/admin' class='list-group-item'>管理頁面 </a>
                <a href='http://tar.so/news' class='list-group-item'>實時新聞</a>
                <a href='http://drho.tw/news' class='list-group-item'>電視新聞</a>
            </div>
            <script type="text/javascript"
            src="http://feedjit.com/serve/?vv=1515&amp;tft=3&amp;dd=0&amp;wid=&amp;p
            id=0&amp;proid=0&amp;bc=FFFFFF&amp;tc=000000&amp;brd1=012B6B&amp;lnk=135
            D9E&amp;hc=FFFFFF&amp;hfc=2853A8&amp;btn=C99700&amp;ww=190&amp;wne=6&amp
            ;srefs=0"></script><noscript><a href="http://feedjit.com/">Live Traffic
            Stats</a></noscript>
        </div>
    </div>
</div>
```

```
        <div class='col-sm-8 col-md-8'>
            <div class='panel panel-default'>
                <div class='panel-heading'>
                    {% block headmessage %} {% endblock %}
                </div>
                <div class='panel-body'>
                    {% block content %} {% endblock %}
                </div>
                <div class='panel-footer'>
                    {% include 'footer.html' %}
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

我們使用 和 的搭配，讓左側邊欄佔用 4 個格子（Bootstrap 把螢幕的橫向分為 12 個格子），而內文的部份佔用 8 個格子，接著在各自的格子中使用 panel 來建立其內容。圖 2-19 即為修改之後的首頁輸出效果（由於側邊欄已設計了回到首頁的連結，因此 post.html 中的回首頁連結即可去除）。

Javascript以及CSS檔案的引用

- 使用Bootstrap為網站建立側邊欄

加入本地端圖片或資源

圖形檔的應用

- 圖形檔大部份會被放置在image資料夾下，而.css和.js檔案則會被放在css和js資料夾下
- Django把這一類型的檔案統稱為static files（靜態檔案）
- 統一把這些檔案（.js, .css, .jpg, .png等等）都放在static的資料夾之下，然後.js放在js子目錄，.css放在css子目錄，而圖形檔則都放在images子目錄中
- 在settings.py中，就要加入如下所示的設定：

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static'),
]
```

到的 logo 檔案 logo.png 放在 static/images 底下，到 header.html 中加入對於圖形檔的存取操作，如下所示：

```
{% load static %}
<div class='well'>
<img height='70px' src="{% static "images/logo.png" %}">
<h1 style="font-family:微軟正黑體;display: inline;">歡迎光臨</h1>
</div>
```

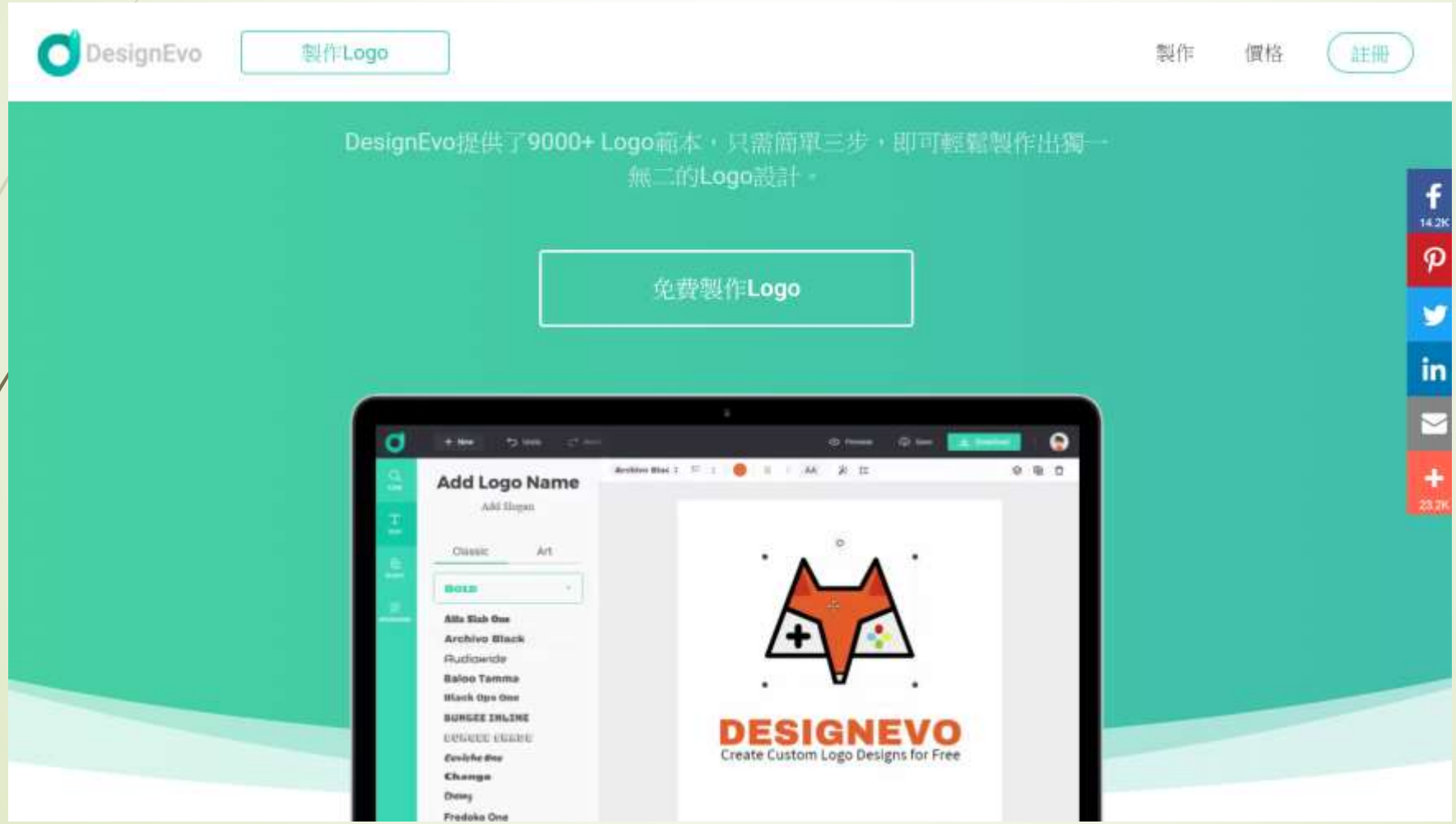
此檔案要留意的地方在於檔案的第 2 行 {% load staticfiles %} 只要使用一次，提醒 Django 去載入所有的靜態檔備用，這一行指令在同一個檔案中只要使用一次即可。而在真正引入圖形檔案的地方，使用了 {% static "images/logo.png" %} 這個模版語言，Django 會依照當時的執行環境把此檔案的可存取網路位址傳遞給瀏覽器。在我們的例子中，在 header.html 中把原本的歡迎文字標題改為 logo.png 圖形檔，執行的結果如圖 2-20。

圖形檔的應用

- 使用圖形檔當做是網站Logo的示範頁面

Logo 設計網站

- https://www.designevo.com/tw/



小練習

- 請完成以上操作
- 加入你設計的logo
- 完成後請用 git commit 本地備份
- 再上傳到遠端儲存倉

在主網頁顯示文章摘要

在部落格網站中還有一個重要的特色，就是在每一篇標題下顯示這篇文章的摘要。要顯示摘要一般有兩種處理方式，一種是直接在資料庫定義的時候，也就是在建立 Model 的時候，就把摘要這個資料項目加進去，讓版主在建立文章的時候就可以選擇輸入摘要，然後在 template 中把它顯示出來，而另外一種方式，也是本小節介紹的方法，就是根據文章的內容，直接擷取前面固定字數的字元，把它們另外再顯示出來。

之前，我們在 template 檔案中如果要輸入變數中的資料，都是以 {{ post.title }} 的方式，把變數的內容依照原來的樣子顯示出來，而其實，在輸出之前，還有所謂的 filter 過濾器可以使用，指定過濾器的方式，就是在變數之後加上「|」即可，例如 {{ post.title | filter_command }}。比較常用的過濾器，如下表所示：

filter

名稱	用途	示例
capfirst	把第一個字母改為大寫	{{value | capfirst}}
center	把字串的內容置中	{{value | center:"12"}}
cut	把字串中指定的字元移除	{{value | cut:"" }}
date	指定日期時間的輸出格式	{{value | date:"d M Y"}}
linebreaksbr	置換 \n 成為 	{{value | linebreaksbr}}
linenumbers	為每一行字串加上行號	{{value | linenumbers}}
lower	把字串轉換為小寫	{{value | lower}}
random	把前面的串列元素使用隨機的方式任選一個輸出	{{value | random}}
striptags	把所有的 HTML 標記全部移除	{{value | striptags}}
truncatechars	擷取指定字數的字元	{{value | truncatechars:40}}
upper	把字串轉為大寫	{{value | upper}}
wordcount	計算字數	{{value | wordcount}}

在主網頁顯示文章摘要

- Template filter

名稱	用途	示例
capfirst	把第一個字母改為大寫	{{value | capfirst}}
center	把字串的內容置中	{{value | center:"12"}}
cut	把字串中指定的字元移除	{{value | cut:" "}}
date	指定日期時間的輸出格式	{{value | date:"d M Y"}}
linebreaksbr	置換\n成為 	{{value | linebreaksbr}}
linenumbers	為每一行字串加上行號	{{value | linenumbers}}
lower	把字串轉換為小寫	{{value | lower}}
random	把前面的串列元素使用隨的方式任選一個輸出	{{value | random}}
striptags	把所有的HTML標記全部移除	{{value | striptags}}
truncatechars	擷取指定字數的字元	{{value | truncatechars:40}}
upper	把字串轉為大寫	{{value | upper}}
wordcount	計算字數	{{value | wordcount}}

我們希望在我們的顯示的首頁文章中可以列出摘要，顯然要使用 truncatechars 這個 filter，另外，也要顯示出每一篇文章的發佈時間，可以透過 date 這個 filter 來調整日期時間格式。除此之外，改良後的主網頁希望可以讓每一篇文章的標題、摘要以及發佈時間能夠有一體感，因為它們是屬於同一篇文章的三個顯示項目，此時透過 Boostrap 中的 Panel 設定，分別設定為 Panel 的 heading、body 以及 footer 剛好。此外，我們也在 Panel 中利用 CSS 指令設定背景顏色，讓每一篇文章之間能夠有所區分。重新設計過的 index.html 如下所示：

```
<!-- index.html -->
{% extends 'base.html' %}
{% block title %} 歡迎光臨我的部落格 {% endblock %}
{% block headmessage %}
        <h3 style='font-family:標楷體;'>本站文章列表
{% endblock %}
```

```
{% block content %}
    {% for post in posts %}
        <div class='panel panel-default'>
            <div class='panel-heading'>
                <p style='font-family:微軟雅黑;font-size:14pt;font-weight:bold;'>
                <a href='/post/{{post.slug}}'>{{ post.title }}</a>
                </p>
            </div>
            <div class='panel-body' style='background-color:#ffffdd'>
                <p>
                    {{ post.body | truncatechars:40 }}
                </p>
            </div>
            <div class='panel-footer' style='background-color:#efefef'>
                <p>
                    發佈時間：{{ post.pub_date | date:"Y M d, h:i:s" }}
                </p>
            </div>
        </div>
        <br>
    {% endfor %}
{% endblock %}
```

歡迎光臨

MENU

- HOME
- 管理頁面
- 實時新聞
- 電視新聞

本站文章列表

噴火龍

牠會為了尋找更強的對手而在天空中四處飛翔。能夠融化一切事物的猛烈火焰絕不會用在…

發佈時間：2019 五月 26, 01:05:00

小火龍

尾巴上的火焰代表目前的心情，高興時火焰會搖晃，生氣時火焰會燃燒的更猛烈。

小練習

- 請完成以上操作
- 並改成顯示60個字的摘要
- 改變發佈日期的格式 如：發佈時間：2019-11-09, 02:30
- 完成後請用 git commit 本地備份
- 再上傳到遠端儲存倉

https://docs.djangoproject.com/en/3.0/ref/templates/builtins/#date

在文章中加入 html

文章的HTML內容處理

本堂課所示範的部落格程式主要目的是讓讀者可以快速上手，為了讓此網站單純一些，對於文章中所使用到的圖形檔，是以使用第三方圖形檔案服務網站（例如 imgur.com）為主。也就是說，所有張貼文章所需要的圖形檔案，處理之後（包括圖形尺寸、浮水印、以及版權聲明等等）上傳到該網站，在取得連結之後再放在我們的文章中。例如，我們在 imgur.com 上傳了一個圖形檔案，在開啟該圖形之後，可以看到如圖 2-22 中箭頭所指的地方，許多不同系統可以使用的連結或是 HTML 碼。

圖 2-22：imgur.com 的圖形連結資訊

在取得了這個資訊之後（在此我們以 HTML 為例），新增部落格文章時，即可以在適當的地方直接貼上此段 HTML 程式碼即可，如圖 2-23 所示。


變更 post | Django 網站管
192.168.58.130:8000/admin/mainsite/post/7/
Django 管理
歡迎， admin. View site / 變更密碼 / 登出
首頁 › Mainsite › Posts › 夏威夷有名的鳳梨漢堡
變更 post
歷史
Title:
夏威夷有名的鳳梨漢堡
Slug:
hawaiihamburger
Body:
這是在夏威夷歐胡島上一家非常有名的賣鳳梨漢堡老店的拿手料理，有去夏威夷的朋友千萬不要錯過。

Pub date:
日期 2016/06/21 今天
時間 19:50 現在
刪除
儲存並新增另一個
儲存並繼續編輯
儲存

在按下儲存之後，此程式碼會被原封不動地儲存在網站的資料庫中，此篇文章即可在網頁中顯示。但是，當我們顯示該篇文章內容的時候，卻有可能會出現如圖 2-24 所示的樣子。

我們插入的 HTML 連結，居然被完整地呈現出來了，這顯然不是好現象。好在，要解決這個問題其實非常地簡單，主因在於 Django 在預設的情況下，是不隨便解讀 HTML 碼的，主要是擔心網站安全性的問題。但由於這是我們自己的部落格網站，並不開放其他人新增資料，因此，只要在 post.html 中在輸出 post.body 的後面，再加上一個 safe 的過濾器即可，如下所示：

```
{{ post.body | safe }}
```

加上了 safe，此文章內的所有 HTML 碼就都可以被順利地解讀出來，當然我們放進去的圖形檔案，也就可以順利地顯示在文章中了，如圖 2-25 所示。

圖 2-25：加上 safe 過濾器，讓文章內的 HTML 碼可以順利地被解讀

同理，其他諸如 CSS 設定也可以透過此方式加上去，也因此就可以在文章中自由地使用 HTML 和 CSS 做出自已想要的排版內容。

- 將貼文內文改成用html顯示後
- 原文章中的'\n'不會發生換行作用
- 請問如何讓'\n'得以換行呢？
- linebreaksbr

- 什麼是 markdown ？
- https://markdown.tw/
- 在git 中建立 readme.md
- 並上傳儲存倉

Markdown語法解析與應用

雖說使用上一小節的方法可以讓我們在編輯文章的時候直接使用 HTML 語法來編輯版面已經是非常方便了，但是對於許多人來說，HTML 語法非常繁瑣而且其實也不太安全，一不小心，錯誤的語法容易造成整個網站的版面也跟著走位、甚至無法瀏覽。因此，有一些部落格網站提供了 Markdown 語法，讓部落客在編輯文章的時候可以兼顧彈性、便利以及安全。Markdown 的語法介紹，請參考網站：http://markdown.tw/ 上的說明。

而要在我們的網站中支援 Markdown 非常簡單，首先需在網站系統中安裝 django-markdown-deux 套件，安裝之後也別忘了使用 pip freeze 把它放到 requirements.txt 中，步驟如下：

```
pip install django-markdown-deux
pip freeze > requirements.txt

pip install -p requirements.txt #還原套件環境
```

接著到 setting.py 中的 INSTALLED_APP 段落中，把 markdown_deux 加進去，如下所示：

```
...略...
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'markdown_deux',
    'mainsite',
)
...略...
```

接著，再到我們會解析 Markdown 語法的 post.html 中載入 Markdown 語法標記以及過濾器，修改之後的 post.html，如下所示：

```
<!-- post.html -->
{% extends 'base.html' %}
{% load markdown_deux_tags %}
{% block title %} {{ post.title }} - 文學天地 {% endblock %}
{% block headmessage %}
   <h3 style='font-family:微軟正黑體;'>{{ post.title }}</h3>
   <a style='font-family:微軟正黑體;' href='/'>回首頁</a>
{% endblock %}
{% block content %}
   <p style='font-family:微軟正黑體;font-size:12pt;letter-spacing:2pt;'>
      {{ post.body | markdown }}</a>
   </p>
{% endblock %}
```

最重要的是 {% extends…%} 的下一行載入 markdown_deux_tags，然後在真正輸出文章內容的地方，把原來的 safe 過濾器置換為 markdown 就好了，就是這麼簡單。存檔之後，我們把原本的文章內容加上簡單的 Markdown 語法，如圖 2-26 所示。

如圖 2-26 所示，「##」是 Markdown 中的小標題，而「!」則是圖形連結。因為該 Markdown 模組會把所有的語法加以過濾處理，所以如果其中有 HTML 標記的話會被視為一般文字，因此只要是需要排版的地方，請不要再使用 HTML，而全部改用 Markdown 才行。在修改完畢之後，首頁顯示畫面由於是 index.html，我們並沒有做解讀，因此看起來如圖 2-27 所示，在顯示摘要的時候，Markdown 語法會被看做是一般文字。

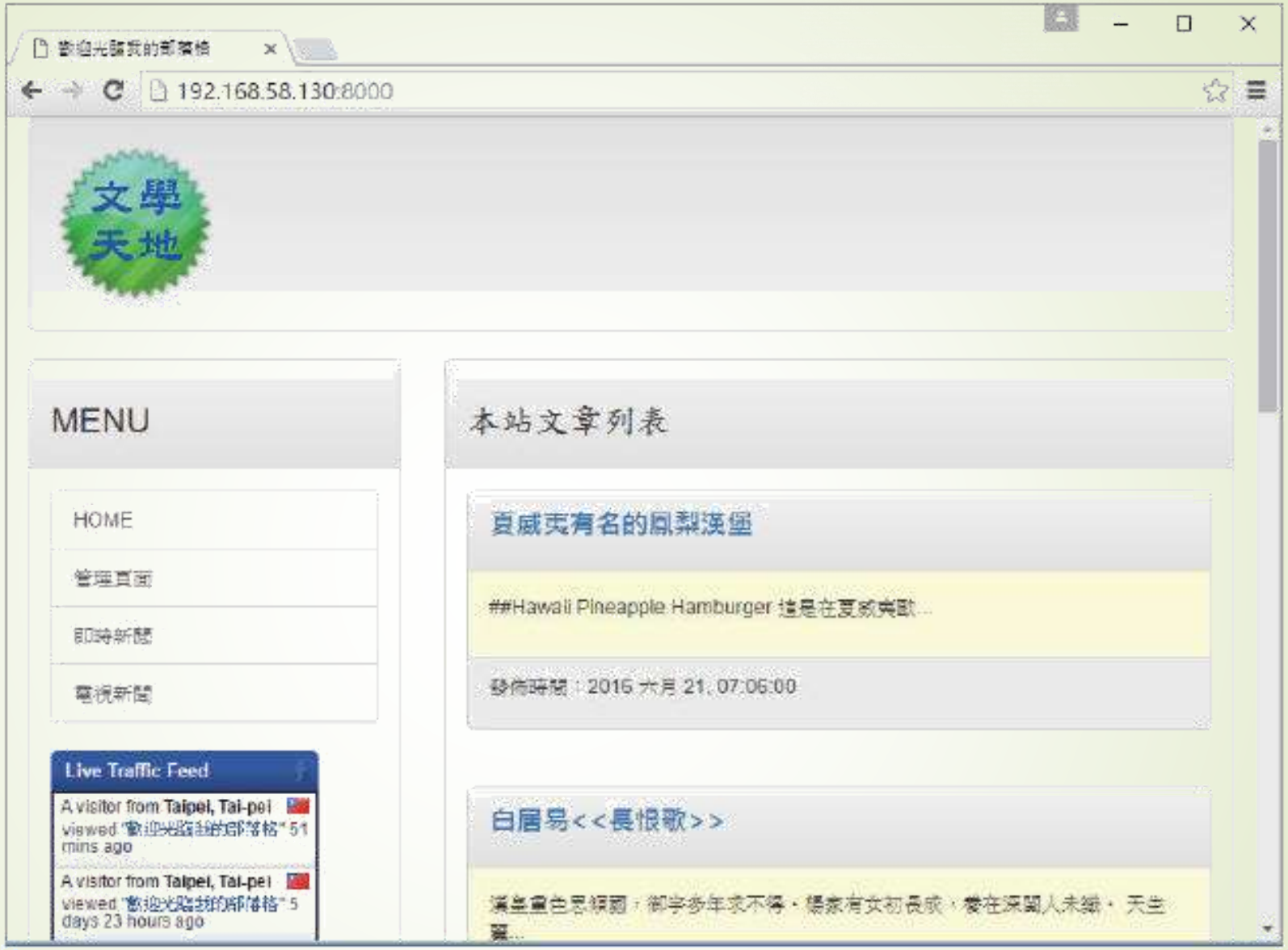


圖 2-27：文章列表會把 Markdown 語法視為一般文字

但是在點擊進入文章之後，就可以看到排版出來的樣子了，如圖 2-28 所示。

歡迎光臨

MENU

- HOME
- 管理頁面
- 實時新聞
- 電視新聞

噴火龍

回首頁

NO.006

噴火龍

リザードン

google

牠會為了尋找更強的對手而在天空中四處飛翔。能夠融化一切事物的猛烈火焰絕不會用在比自己弱的對手身上。

- 令markdown可以使用html
- 在settings中增加

```
MARKDOWN_DEUX_STYLES = {
    "default": {
        "extras": {
        "code-friendly": None,
    },
        "safe_mode": False,
    },
}
```

- https://github.com/trentm/django-markdown-deux
- https://github.com/trentm/python-markdown2/wiki/code-friendly

待解決問題

- 無法顯示 code block
- https://www.xtuz.net/detail-49.html

習題

- 試著建立一屬於自已的部落格站
- 請在首頁中加入也可以解析markdown的語法功能
- 嘗試建立一些markdown語法排版的文章

- 進階練習1：新增摘要的欄位在資料模型(Model)中，將原有的自動20個字的摘要(abstract)，改成手動輸入

- 進階練習2：新增圖片連結到資料庫欄位中(pic)，並可以在index.html之中與title一同顯示出來

pic_url= models.URLField(max_length=250)

需要models.py中加入

abstract = models.TextField()

python manage.py makemigrations

python manage.py migrate

由於更動到資料庫的結構

1

'default abstract'

```
(VENV01) C:\Users\shiuny\Dropbox\文件\ex\myblog>python manage.py makemigrations
You are trying to add a non-nullable field 'abstract' to post without a default; we can't do that (the dat
mething to populate existing rows).
Please select a fix:
 1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
 2) Quit, and let me add a default in models.py
Select an option: 1
Please enter the default value now, as valid Python
The datetime and django.utils.timezone modules are available, so you can do e.g. timezone.now
Type 'exit' to exit this prompt
>>> 'default abstract'
Migrations for 'minsite':
  minsite\migrations\0004_post_abstract.py
    - Add field abstract to post
```

而正確的內容則要到/admin管理介面中去編輯。

{{ post.body | truncatechars:40 }}指令改為{{ post.abstract }}即可

```
<!-- index.html --> 加入以下兩行
{% load markdown_deux_tags %}
{{ post.body | markdown | truncatechars:40 }}
#或是
{{ post.abstract | markdown }}
```

回家作業

- 請完成一個有以下功能的部落格網站：
  - 引入boostrap排版
  - 可以使用markdown進行內文撰寫
  - 使用使用自訂html加入圖片或其它裝飾來美化網站
  - 新增摘要的欄位在資料模型中，將原有的自動20個字的摘要，改成手動輸入，並可使用markdown
  - 新增圖片連結到資料庫欄位中，並可以在index.html之中與title一同顯示出來
- 將整個網站資料夾用壓縮軟體壓縮起來後
- 在課程結束前於課程網站中繳交
- 作業格式務必按照格式命名

歡迎光臨

MENU

- HOME
- 管理頁面
- 實時新聞
- 電視新聞

噴火龍

回首頁

NO.006

噴火龍

リザードン

google

牠會為了尋找更強的對手而在天空中四處飛翔。能夠融化一切事物的猛烈火焰絕不會用在比自己弱的對手身上。

補充

- django-admin 與 manage.py
  https://docs.djangoproject.com/zh-hans/2.2/ref/django-admin/