



django girls

學習指南

Table of Contents

Django Girls 學習指南	1.1
Django 介紹	1.2
安裝 Django	1.3
Project and Apps	1.4
Views and URLconfs	1.5
Templates	1.6
Models	1.7
Admin	1.8
Django ORM	1.9
Template Tags	1.10
Dynamic URL	1.11
Deploy	1.12
What's Next?	1.13

Django Girls 學習指南

這份學習指南適合所有 Django 初學者，為了更好的學習效果，我們希望你能具備：

- Web 的初步認識
- 了解如何使用 Command Line
- 略懂 Python 基礎語法
- 看得懂簡單的 HTML / CSS

學習前準備

在使用這份指南前，請先準備好：

1. 安裝 [Python 3.5](#)
2. 註冊 [PythonAnywhere](#)

學習範例

透過這份學習指南，你會學習到 Django 的程式架構，從創建一個專案，到最後將網站發佈到網路上，建立一個屬於自己的旅遊日記。

旅遊日記首頁

The screenshot shows a web browser window with the title bar "A Django Girl's Adventure" and the URL "djangogirlstrips.herokuapp.com". The main content area has a wooden background pattern. It displays two travel posts:

- 漫步夕陽下與星光共飲** (2014/09/23)
A photograph of a tropical beach at night with palm trees and a bright neon sign that reads "BORACAY" with a heart symbol. Below the photo is a short text summary:

長灘的夕陽就像一名絕色美女，美得令人窒息。但相機怎麼樣都拍不出它的十分之一，我只好把我的手機丢在一旁(也可能是我不夠專業，沒有帶單眼)，用肉眼紀錄此刻的畫面。一到晚上沙灘上全擺起了桌椅，夜晚的沙灘好不熱鬧，全是一間間露天餐廳，有的是年輕女孩在做火舞表演，有的則是樂團駐唱。當月亮高掛天空，夜色更深一點

[Read More →](#)
- 台北車站的太陽番茄拉麵** (2014/09/26)
A photograph of a bowl of太阳番茄拉麵 (Sun Tomato Ramen) with a large pile of cheese on top. Below the photo is a short text summary:

想到台北車站，就會想到過往的補習時光，以及逝去的青春(嘆)離題了.....台北凱撒大飯店B1美食街，地處人來人往的美食一級戰區，非常適合來挖寶，除了 MOS，上次也嘗試了CP值超高的米塔義式廚房。今天要來挑戰的就是每次經過都會被誘惑的「太陽番茄拉麵」！

[Read More →](#)

旅遊日記 - 單篇日記頁面

A Django Girl's Adventure

台北車站的太陽番茄拉麵

2014 / 09 / 26



想到台北車站，就會想到過往的補習時光，以及逝去的青春(嘆) 離題了.....台北凱撒大飯店B1美食街，地處人來人往的美食一級戰區，非常適合來挖寶，除了 MOS，上次也嘗試了CP值超高的米塔義式廚房。今天要來挑戰的就是每次經過都會被誘惑的「太陽番茄拉麵」！



Django 介紹

Django (/dʒæŋgəʊ/ jang-goh) 可以說是 Python 最著名的 Web Framework，一些知名的網站如 [Pinterest](#), [Instagram](#), [Disqus](#) 等等都使用過它來開發。

它有以下的特色：

- 免費開放原始碼
- 著重快速開發、高效能
- 遵從 [DRY \(Don't Repeat Yourself \)](#) 守則，致力於淺顯易懂和優雅的程式碼
- 使用類似 Model–view–controller (MVC) pattern 的架構

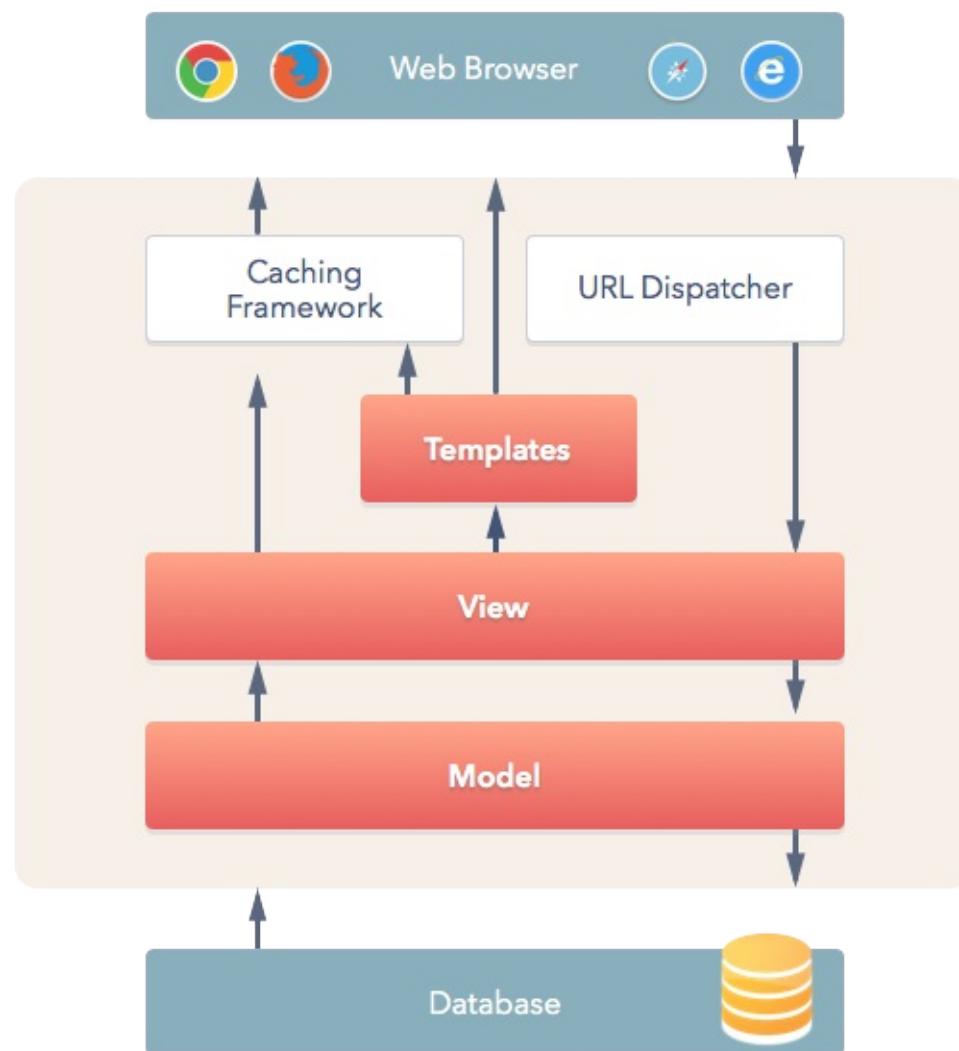
Web Framework

Web framework，簡單來說就是當你開發 Web 應用程式時所用的框架。它通常會提供：

1. 一個既定的程式骨架 -- 你必須按照它的規範寫程式，例如把資料庫相關的程式與跟畫面溝通的程式分開，而不是全部寫在同一個檔案。這對於程式的開發速度、再利用性、和程式可讀性等等都有相當大的好處。
2. 強大且豐富的函式庫 (**Libraries**) -- 通常會提供一些開發網站所需要且常用的功能，例如使用者認證、安全機制、URL mapping、資料庫連接等等。讓你在開發網站時可以直接使用函式庫，然後專注在客製化自己的功能。

Django 的 架構

如同一些比較著名的 Web framework，Django 同樣也使用了類似 MVC 的架構，只是在定義和解釋上略有不同，稱為 MTV (Model–Template–View)，我們可以透過下面這張圖來了解其運作方式：



安裝 Django

在這一章，我們會開始建立第一個 Django 專案，並瞭解如何使用虛擬環境。

首先，請開啓終端機，確定現在的位置是在家目錄底下：

我們先建立一個名為 `djangogirls` 的資料夾

```
mkdir djangogirls
```

並切換至剛剛建立的目錄

```
cd djangogirls
```

虛擬環境（virtualenv）

在安裝 Django 之前，我們要先建立一個虛擬環境（virtual environment）。

我們可以直接開始安裝 Django，但實務上，大多數人都會搭配使用虛擬環境。使用虛擬環境有許多優點：

- 你的專案會擁有一個專屬的獨立 Python 環境。
- 不需要 root 權限，就可以安裝新套件。
- 方便控管不同版本的套件，不用擔心升級套件會影響到其他專案。
- 如果需要多人協作或在不同機器上跑同一個專案時，使用虛擬環境也可以確保環境一致性。

創建虛擬環境

在較舊的 Python 版本中，建立虛擬環境需要另外安裝。但 Python 3.3 之後已經加入 `venv` 模組，可以直接使用。

那我們立刻開始，首先要創建一個虛擬環境資料夾 `djangogirls_venv`。

Windows

如果有按照安裝教學，使用 **Django Environment** 開啓終端機後，輸入以下指令：

```
C:\Users\YOUR_NAME\.djangogirls> python -m venv djangogirls_venv
```

Linux / OS X

Linux 或 OS X 需要使用 `python3` 來建立虛擬環境，指令如下：

```
~/djangogirls$ python3 -m venv djangogirls_venv
```

切換虛擬環境

虛擬環境建立完成後，我們可以透過 `activate` 這個 script 來啓動它。

記得未來在安裝新套件，或是要執行 Django 相關指令時，都要先啓動該專案的虛擬環境。

Windows

```
C:\Users\YOUR_NAME\.djangogirls> djangogirls_venv\Scripts\activate
```

Linux / OS X

```
~/djangogirls$ source djangogirls_venv/bin/activate
```

如果無法使用 `source` 的話，可以用下列指令替代：

```
~/djangogirls$ . djangogirls_venv/bin/activate
```

目前的虛擬環境

如果看到前面多了 (虛擬資料夾名稱)，則表示已經成功切換至該虛擬環境。

Windows

```
(djangogirls_venv) C:\Users\YOUR_NAME\.djangogirls>
```

Linux / OS X

```
(djangogirls_venv) ~/djangogirls$
```

安裝 Django 1.8 最新版本

開始安裝

Python 3.4 預先安裝了 `pip` 這個強大的套件管理工具，我們將使用它來安裝 Django：

```
(djangogirls_venv) ~/djangogirls$ pip install "django<1.9"
```

這裡需要特別注意，我們使用的指令是 `"django <1.9"`。這樣一來才可以確保我們安裝的是 **Django 1.8** 的最新版本

輸入了應該會看到如下的訊息，表示安裝成功

```
Installing collected packages: django
Successfully installed django-1.8.6
```

註：如果你看到以 *Fatal error in launcher* 開頭的輸出，而不是上面的安裝成功訊息，請改用 `python -m pip install "django<1.9"` 試試看。之後如果在使用 `pip` 時遇到類似問題，也可以試著在前面加上 `python -m`。

確認安裝成功

最後，讓我們最後來測試一下。

請在虛擬環境下指令輸入 `python`，進入互動式命令列環境

```
(djangogirls_venv) ~/djangogirls$ python
```

輸入以下的指令取得 Django 版本資訊：

```
>>> import django
>>> django.VERSION
(1, 8, 6, 'final', 0)
```

如果看見類似上面的訊息，就代表安裝成功囉！

Project and apps

每一個 Django project 裡面可以有多個 Django apps，可以想成是類似模組的概念。在實務上，通常會依功能分成不同 **app**，方便未來的維護和重複使用。

例如，我們要做一個類似 Facebook 這種網站時，依功能可能會有以下 apps：

- 使用者管理 -- accounts
- 好友管理 -- friends
- 塗鴉牆管理 -- timeline
- 動態消息管理 -- news

若未來我們需要寫個購物網站，而需要會員功能時，accounts app（使用者管理）就可以被重複使用。

這一章，你會學到如何使用 Django 命令列工具建立 Django project 和一個 Django app。

建立 Django project

建立專案資料夾 -- startproject

首先，使用 `django-admin.py` 來建立第一個 Django project `mysite`：

```
(djangogirls_venv) ~/djangogirls$ django-admin.py startproject mysite
```

此時會多了一個 **mysite** 資料夾。我們切換進去：

```
(djangogirls_venv) ~/djangogirls$ cd mysite
```

`startproject` 這個 Django 指令除了建立專案資料夾，也預設會建立一些常用檔案，你可以使用 `ls` 或 `dir /w` (Windows) 檢視檔案結構。

目前 project 的檔案結構如下：

```
mysite/
└── manage.py
└── mysite
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

瞭解 Django 的 Management commands

`manage.py` 是 Django 提供的命令列工具，我們可以利用它執行很多工作，例如同步資料庫、建立 app 等等，指令的使用方式如下：

```
python manage.py <command> [options]
```

如果你想要了解有什麼指令可以使用，輸入 `help` 或 `-h` 指令會列出所有指令列表：

```
python manage.py -h
```

而如果想了解其中一個指令，可以在指令名字後輸入 `-h`，你會看到簡單的的指令介紹以及用法說明。以 `runserver` 為例：

```
(djangogirls_venv) ~/djangogirls/mysite$ python manage.py runserver -h
usage: manage.py runserver [-h] [--version] [-v {0,1,2,3}]
                           [--settings SETTINGS] [--pythonpath PYTHONPATH]
                           [--traceback] [--no-color] [--ipv6] [--nothreading]
                           [--noreload] [--nostatic] [--insecure]
                           [addrport]

Starts a lightweight Web server for development and also serves static files.

positional arguments:
  addrport            Optional port number, or ipaddr:port

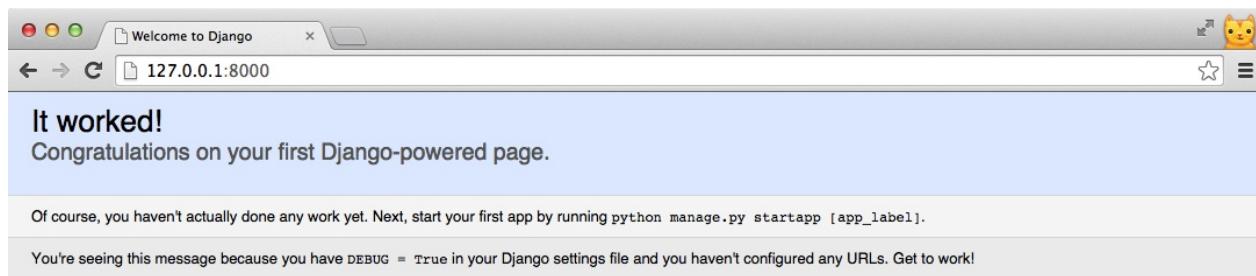
optional arguments:
  -h, --help          show this help message and exit
  --version          show program's version number and exit
  -v {0,1,2,3}, --verbosity {0,1,2,3}
                     Verbosity level; 0=minimal output, 1=normal output,
                     2=verbose output, 3=very verbose output
  --settings SETTINGS The Python path to a settings module, e.g.
                     "myproject.settings.main". If this isn't provided, the
                     DJANGO_SETTINGS_MODULE environment variable will be
                     used.
  --pythonpath PYTHONPATH
                     A directory to add to the Python path, e.g.
                     "/home/djangoprojects/myproject".
  --traceback         Raise on CommandError exceptions
  --no-color          Don't colorize the command output.
  --ipv6, -6          Tells Django to use an IPv6 address.
  --nothreading       Tells Django to NOT use threading.
  --noreload          Tells Django to NOT use the auto-reloader.
  --nostatic          Tells Django to NOT automatically serve static files
                     at STATIC_URL.
  --insecure          Allows serving static files even if DEBUG is False.
```

啓動開發伺服器 -- runserver

從說明中可以知道，`runserver` 會啓動一個簡單的 web server，方便於在開發階段使用：

```
(djangogirls_venv) ~/djangogirls/mysite$ python manage.py runserver
...
Django version 1.8.5, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

現在打開瀏覽器輸入 <http://127.0.0.1:8000/> 或是 <http://localhost:8000/>，會看到你的 django 專案已成功在 web server 上執行



最後我們可以在終端機按下 `CTRL+C`，關閉 web server 回到命令列。

如果無法看到成功畫面，瀏覽器上顯示錯誤訊息 - *A server error occurred. Please contact the administrator.*，請輸入：

```
(djangogirls_venv) ~/djangogirls/mysite$ python manage.py migrate
```

然後再次 `runserver` 啓動你的 web server，我們會在 **Django Models** 解釋 `migrate` 的作用。

建立 Django application (app)

讓我們利用 `startapp` 建立第一個 Django app -- **trips**:

```
(djangogirls_venv) ~/djangogirls/mysite$ python manage.py startapp trips
```

`startapp` 會按照你的命名建立一個同名資料夾和 app 預設的檔案結構如下：

```
trips
├── __init__.py
├── admin.py
├── migrations
├── models.py
├── tests.py
└── views.py
```

將新增的 Django app 加入設定檔

在前一個指令，我們透過 Django 命令列工具建立了 **trips** 這個 app。但若要讓 Django 知道要管理哪些 apps，還需再調整設定檔。

新增 app

打開 `mysite/settings.py`，找到 `INSTALLED_APPS`，調整如下：

```
# mysite/settings.py

...
# Application definition

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

請注意 app 之間有時候需要特定先後順序。在此，我們將自訂的 `trips` 加在最後面。

預設安裝的 Django app

Django 已將常用的 app 設定為 `INSTALLED_APPS`。例如，`auth`（使用者認證）、`admin`（管理後台）... 等等，我們可依需求自行增減。

小結

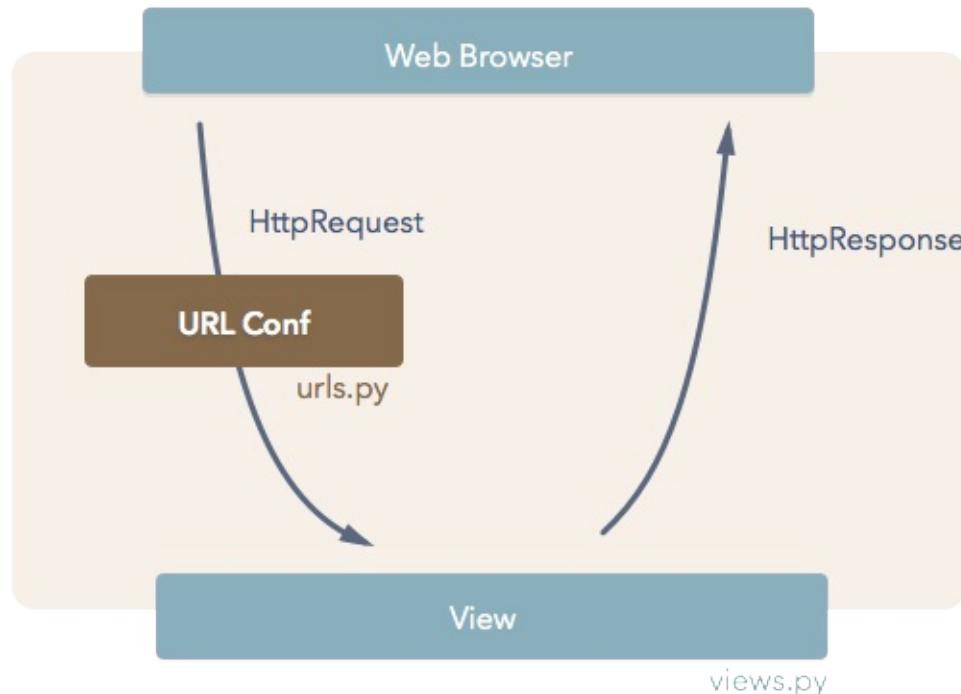
目前為止，我們使用 `startproject` 建立了一個名為 `mysite` 的 Django 專案，和一個名為 `trips` 的 Django app。

```
mysite
└── manage.py
└── mysite
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
└── trips
    ├── __init__.py
    ├── admin.py
    ├── migrations
    ├── models.py
    ├── tests.py
    └── views.py
```

最後，我們回顧一下本章學到的指令

指令	說明
<code>django-admin.py startproject <project_name></code>	建立 Django 專案
<code>python manage.py -h <command_name></code>	查看 Django commands 的使用方法
<code>python manage.py runserver</code>	啓動開發伺服器
<code>python manage.py startapp <app_name></code>	新增 Django app

Views and URLconfs



在前面的介紹，我們有提到 Django 的 MTV 架構。其處理 `request` 的流程如下：

1. 濕覽器送出 **HTTP request**
2. Django 依據 **URL configuration** 分配至對應的 View
3. View 進行資料庫的操作或其他運算，並回傳 `HttpResponse` 物件
4. 濕覽器依據 **HTTP response** 顯示網頁畫面

這一章，我們將透過 **Hello World** 範例，瞭解 Django 如何處理一個 `request` 的流程。

Django Views

Django view 其實是一個 function，處理 `HttpRequest` 物件，並回傳 `HttpResponse` 物件，大致說明如下：

- 會收到 `HttpRequest` 參數：Django 從網頁接收到 `request` 後，會將 `request` 中的資訊封裝產生一個 `HttpRequest` 物件，並當成第一個參數，傳入對應的 view function。
- 需要回傳 `HttpResponse` 物件：`HttpResponse` 物件裡面包含：

- `HttpResponse.content`
- `HttpResponse.status_code` ... 等

建立第一個 View

首先建立一個名為 `hello_world` 的 view。

在 `trips/views.py` 輸入下列程式碼：

```
# trips/views.py

from django.http import HttpResponse

def hello_world(request):
    return HttpResponse("Hello World!")
```

以上程式在做的事就是：

1. 從 `django.http` 模組中引用 `HttpResponse` 類別
2. 宣告 `hello_world` 這個 view
3. 當 `hello_world` 被呼叫時，回傳包含字串 **Hello World!** 的 `HttpResponse` 物件。

Django URL 設定

最後，Django 需要知道 **URL** 與 **view** 的對應關係。

例如：

有人瀏覽 <http://127.0.0.1:8000/hello/> 時，`hello_world()` 這個 view function 需要被執行。

而這個對應關係就是 **URL conf** (URL configuration)。

URL Conf

- 通常定義在 `urls.py`
 - 是一連串的規則 (URL patterns)
 - Django 收到 `request` 時，會一一比對 URL conf 中的規則，決定要執行哪個 view function
-

現在我們來設定 Hello World 範例的 URL conf。

首先打開 `mysite/urls.py`，先 import 剛剛寫的 view function：

```
from trips.views import hello_world
```

然後在 `urlpatterns` 中加入下面這行：

```
url(r'^hello/$', hello_world),
```

現在 `mysite/urls.py` 的內容應該會像下面這樣：

```
# mysite/urls.py

from django.conf.urls import include, url
from django.contrib import admin
# Import view functions from trips app.
from trips.views import hello_world

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

以上程式透過 `url()` function 傳入兩個參數 `regex` , `view` :

```
url(regex, view)
```

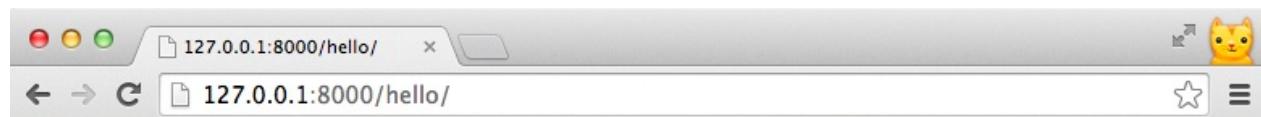
- **regex** -- 定義的 URL 規則
 - 規則以 regular expression (正規表示式) 來表達
 - `r'^hello/$'` 代表的是 `hello/` 這種 URL
- **view** -- 對應的 view function
 - 指的是 `hello_world` 這個 view

測試 Hello World

現在啓動你的 web server。 (如果剛剛沒關閉的話，通常 Django 會在你修改程式碼後，自動重新啓動 web server)

```
(djangogirls_venv) ~/djangogirls/mysite$ python manage.py runserver
```

在瀏覽器輸入 <http://127.0.0.1:8000/hello/>，你會看到網頁顯示我們在 `HttpResponse` 傳入的文字 `Hello World!`。



Hello World!

Templates

上一章的例子，只是很簡單的顯示一行字串。現在，讓我們加上一些 HTML/CSS 美化網頁，並動態顯示每次進來這個頁面的時間。

第一個 Template

實務上，我們會將前端的程式碼獨立出來，放在 `templates` 資料夾裡。不僅增加可讀性，也方便與設計師或前端工程師分工。

Template 資料夾

首先建立 Template 資料夾。開啓終端機 (如果不想關閉 web server，可以再開新一個新的終端機視窗)，並確認目前所在位置為 `djangogirls/mysite/`。

新增一個名為 `templates` 的資料夾：

```
(djangogirls_venv) ~/djangogirls/mysite$ mkdir templates
```

設定 Templates 資料夾的位置

建立好資料夾以後，我們需要修改 `mysite/settings.py` 中的 `TEMPLATES` 設定：

```
# mysite/settings.py

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates').replace('\\', '/')],
        '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',
            ],
        },
    },
]
```

我們將 'DIRS' 原本的 [] 修改成：

```
[os.path.join(BASE_DIR, 'templates').replace('\\', '/')]
```

好讓 Django 找得到剛剛建立的 templates 資料夾。

建立第一個 Template

新增檔案 templates/hello_world.html：

```
mysite
├── mysite
└── templates
    └── hello_world.html
└── trips
└── manage.py
```

並將下列的 HTML 複製到 hello_world.html：

```
<!-- hello_world.html -->

<!DOCTYPE html>
<html>
  <head>
    <title>I come from template!!</title>
    <style>
      body {
        background-color: lightyellow;
      }
      em {
        color: LightSeaGreen;
      }
    </style>
  </head>
  <body>
    <h1>Hello World!</h1>
    <em>{{ current_time }}</em>
  </body>
</html>
```

在 Template 中顯示變數

以上 Template 中，有個地方要特別注意：

```
<em>{{ current_time }}</em>
```

在 Template 裡面，我們會使用兩個大括號，來顯示變數 `current_time`。

`{{ <variable_name> }}` 是在 Django Template 中顯示變數的語法。

其它 Django Template 語法，我們會在後面的章節陸續練習到。

使用 render function

最後，將 view function `hello_world` 修改如下：

```
# trips/views.py

from datetime import datetime
from django.shortcuts import render

def hello_world(request):
    return render(request, 'hello_world.html', {
        'current_time': str(datetime.now()),
    })
```

1. 顯示目前時間：為了顯示動態內容，我們 import `datetime` 時間模組，並用 `datetime.now()` 取得現在的時間。

2. `render`：我們改成用 `render` 這個 function 產生要回傳的 `HttpResponse` 物件。

這次傳入的參數有：

- `request` -- `HttpRequest` 物件
- `template_name` -- 要使用的 template
- `dictionary` -- 包含要新增至 template 的變數

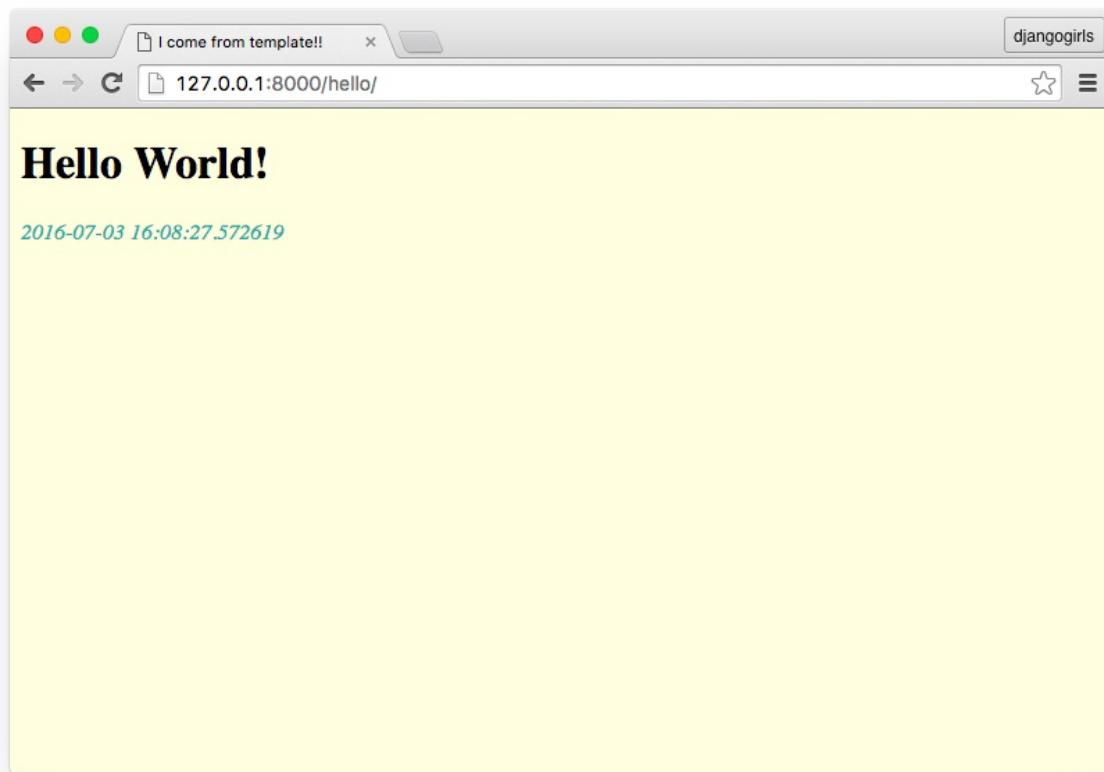
`render` : 產生 `HttpResponse` 物件。

`render(request, template_name, dictionary)`

大功告成

現在啓動 web server，連至 <http://127.0.0.1:8000/hello/> 後，會發現網頁不再是純文字。除了加上了一些樣式外，也會顯示當下的時間。

你可以重新整理網頁，試試看時間有沒有改變



Models

現今的網站，都不再只是僅單純展示網頁內容的靜態網頁。大多數網站，都會加上一些與使用者互動的功能，如留言版、討論區、投票等等。而這些使用者產出的資料，往往會儲存於資料庫中。

這一章，你會學到如何利用 Django Model 定義資料庫的結構（schema），並透過 Django 指令創建資料庫、資料表及欄位。

使用 Django Model 的好處

雖然資料庫的語法有其標準，但是各家資料庫還是或多或少有差異。使用 Django Model 的來操作資料庫的優點之一，就是資料庫轉換相當方便。

在大部份情況下，不再需要為不同的資料庫，使用不同語法來撰寫程式。只要修改設定，就可以輕易地從 SQLite 轉換到 MySQL、PostgreSQL、或是 Oracle 等等。

設定資料庫

為了開發方便，我們使用 Python 預設的資料庫引擎 - SQLite。打開 `mysite/settings.py`，看看 `DATABASES` 的設定。它應該長得像下面這樣：

```
# mysite/settings.py

...
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

在這裡我們設定了資料庫連線的預設值：

- **ENGINE** -- 你要使用的資料庫引擎。例如：
 - MySQL: `django.db.backends.mysql`

- SQLite 3: `django.db.backends.sqlite3`
- PostgreSQL: `django.db.backends.postgresql_psycopg2`
- **NAME** -- 你的資料庫名稱

如果你使用 MySQL 或 PostgreSQL 等等資料庫，可能還要設定它的位置、名稱、使用者等等。不過我們這裡使用的 SQLite 3 不需要這些性質，所以可以省略。

Django Models

我們在 `trips/models.py` 宣告一個 `Post` 類別，並定義裡面的屬性，而 Django 會依據這個建立資料表，以及資料表裡的欄位設定：

```
# trips/models.py

from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField(blank=True)
    photo = models.URLField(blank=True)
    location = models.CharField(max_length=100)
    created_at = models.DateTimeField(auto_now_add=True)
```

- Django 預設會為每一個 Model 加上 `id` 欄位，並將這個欄位設成 primary key（主鍵），簡稱 `pk`，讓每一筆資料都會有一個獨一無二的 ID。
- 為 `Post` 定義以下屬性：

屬性	資料型態	說明	參數
title	CharField	標題	<code>max_length=100</code> -- 標題不可以超過 100 個字元
content	TextField	內文	<code>blank=True</code> -- 非必填欄位（表單驗證時使用），預設所有欄位都是 <code>blank=False</code>
photo	URLField	照片網址	同 content，非必填欄位
location	CharField	地點	同 title
created_at	DateTimeField	建立時間	<code>auto_now_add=True</code> -- 物件新增的時間。若想設成物件修改時間，則用 <code>auto_now=True</code>

Model fields 可為 Django Model 定義不同型態的屬性。

- **CharField** -- 字串欄位，適合像 title、location 這種有長度限制的字串。
- **TextField** -- 合放大量文字的欄位
- **URLField** -- URL 設計的欄位
- **DateTimeField** -- 日期與時間的欄位，使用時會轉成 Python `datetime` 型別。

更多 Model Field 與其參數，請參考 [Django 文件](#)

同步資料庫

首先執行 `makemigrations` 指令：

```
(djangogirls_venv) ~/djangogirls/mysite$ python manage.py makemigrations
Migrations for 'trips':
  0001_initial.py:
    - Create model Post
```

這個指令會根據你對 Model 的修改刪除建立一個新的 migration 檔案，讓 `migrate` 指令執行時，可以照著這份紀錄更新資料庫。

接著用以下的指令，讓 Django 根據上面的紀錄，把 `models.py` 中的欄位寫入資料庫：

```
(djangogirls_venv) ~/djangogirls/mysite$ python manage.py migrate
```

結果應該類似下面這樣：

```
Operations to perform:
  Synchronize unmigrated apps: staticfiles, messages
  Apply all migrations: sessions, admin, auth, contenttypes
Synchronizing apps without migrations:
  Creating tables...
    Running deferred SQL...
  Installing custom SQL...
Running migrations:
  Rendering model states... DONE
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... 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 sessions.0001_initial... OK
  Applying trips.0001_initial... OK
```

`migrate` 指令會根據 `INSTALLED_APPS` 的設定，按照 `app` 順序建立或更新資料表，將你在 `models.py` 裡的更新跟資料庫同步。

Admin

大部份網站都設計有管理後台，讓管理者方便新增或異動網站內容。

而這樣的管理後台，Django 也有內建一個 App -- **Django Admin**。只需要稍微設定，網站就能擁有管理後台功能。

前一章，我們學到如何使用 Django Model 抽象地表達資料庫結構。現在，我們要透過 **Django Admin** 看到實際的資料，並跟資料庫進行互動。

完成本章後，你會瞭解如何設定 Django Admin，並使用 Django 管理後台，完成 Post 的新增、修改及刪除。

設定管理後台

將 **Django Admin** 加入 **INSTALLED_APPS**

後台管理的功能 Django 已預設開啓。因此，設定檔中的 `INSTALLED_APPS` 裡，已經有 `django.contrib.admin` 這個 app：

```
# mysite/settings.py

INSTALLED_APPS = (
    'django.contrib.admin',
    ...
)
```

當你在同步資料庫時，也會建立需要的資料表及欄位。

設定管理後台的 URL

為了讓你可以從瀏覽器進入管理後台，我們需要設定對應的 urls。

我們將管理後台的網址設定為 `/admin/`。確認 `mysite/urls.py` 中的 `urlpatterns` 包含下面這行：

```
url(r'^admin/', include(admin.site.urls)),
```

建立 superuser

要使用 Django 的管理後台，需要一個管理員帳號。

使用 `createsuperuser` 這個指令，建立一個 superuser：

```
(jangogirls_venv) ~/jangogirls/mysite$ python manage.py createsuperuser
Username (leave blank to use 'YOUR_NAME'):
Email address: your_name@yourmail.com
Password:
Password (again):
Superuser created successfully.
```

輸入帳號、Email、密碼等資訊，就完成 superuser 的新增了。

註冊 Model class

最後，我們需要在讓 Django 知道，有哪些 Model 需要管理後台。

修改 `trips app` 裡的 `admin.py`，並註冊 `Post` 這個 Model：

```
# trips/admin.py

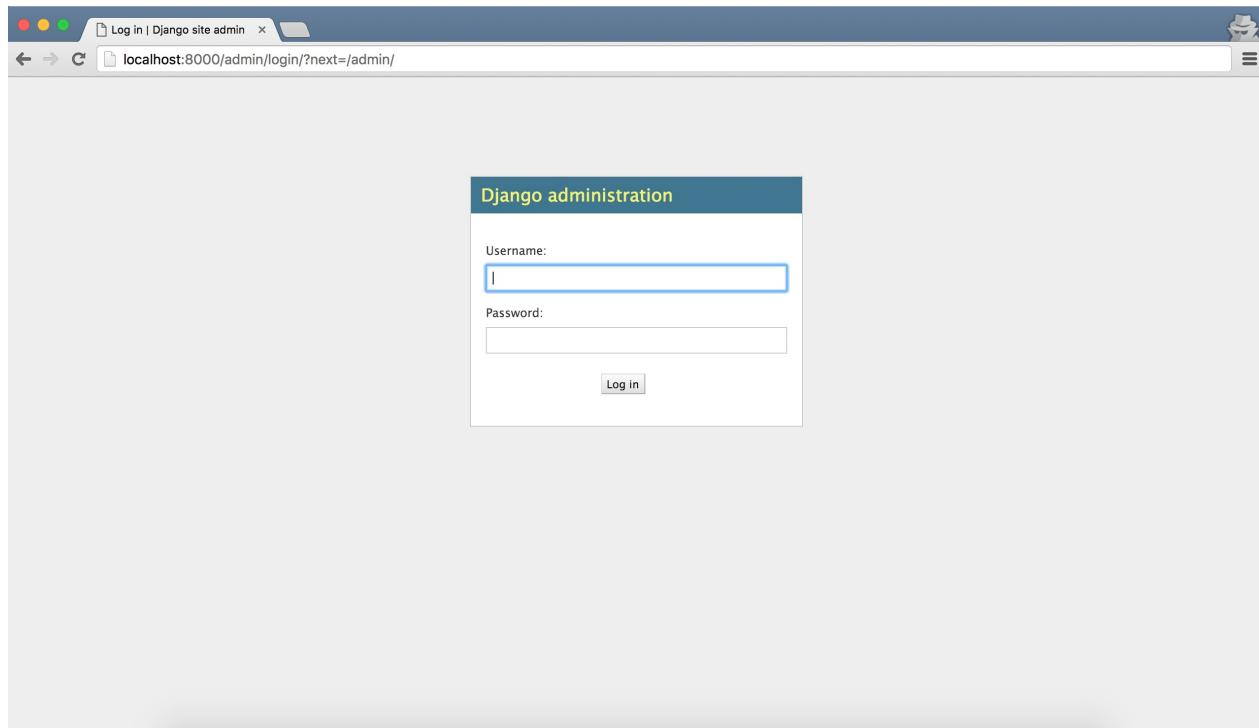
from django.contrib import admin
from .models import Post

admin.site.register(Post)
```

使用管理後台

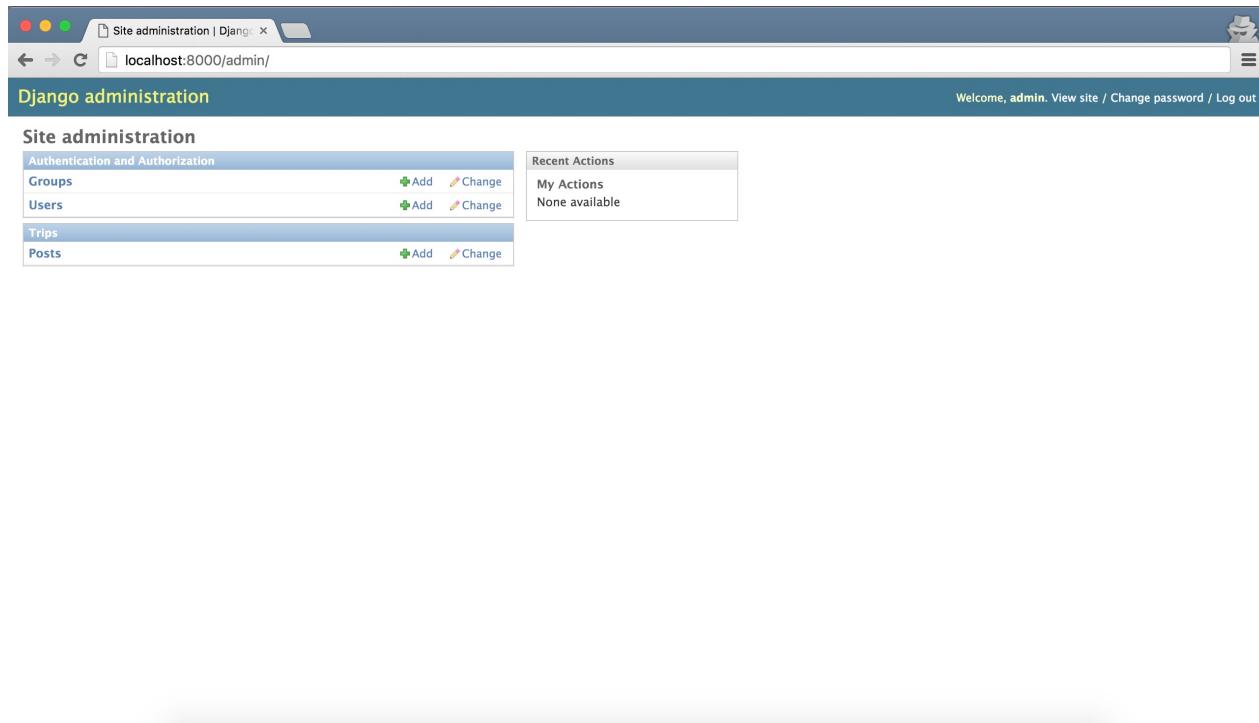
進入管理後台

連至 <http://127.0.0.1:8000/admin>，可以看到管理後台的登入頁面：



請輸入你剛創立的 superuser 帳號密碼，進入管理後台：

第一個區塊 **Authentication and Authorization**，可以管理使用者（User）和群組（Group）；第二個 **Trips** 區塊裡，則可以看到剛剛設定的 Post model。在這裡可以執行 Post 的新增、修改、刪除等功能。



新增一個 Post

現在試著建立一個新的 Post 看看：

The screenshot shows the 'Add post' form in the Django Admin. It includes fields for 'Title' (with a placeholder 'Title:'), 'Content' (a large text area), 'Photo' (a file input field), and 'Location' (another file input field). At the bottom right are three buttons: 'Save and add another', 'Save and continue editing', and a larger blue 'Save' button.

建立成功後會回到 Posts 頁面，你會發現有一筆資料顯示為 Post object :

The screenshot shows the 'Select post to change' page. A green banner at the top says 'The post "Post object" was added successfully.' Below it is a table with one row, showing a checkbox next to 'Post object'. The table has columns for Action, Post, and Post object, and a footer row indicating '1 post'.

Django 通常以 Post object 來表示 Post 物件，但此種顯示不易辨別。我們可以透過 def __str__ 更改 Post 的表示方式。

修改 trips/models.py :

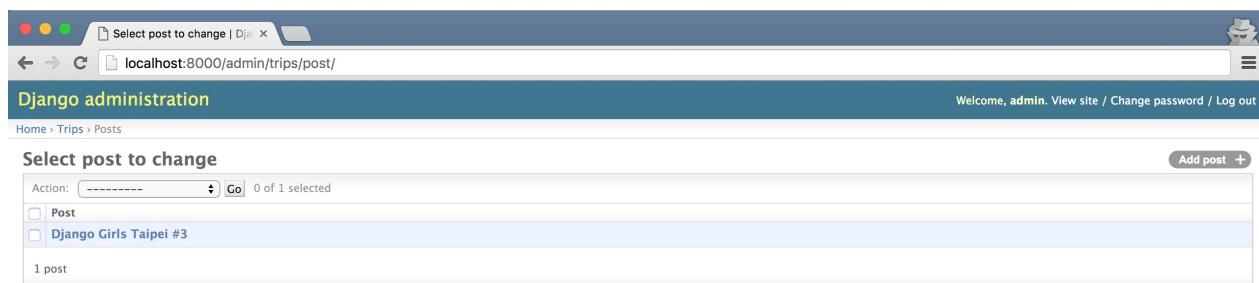
```
# trips/models.py

from django.db import models

class Post(models.Model):
    ...
    created_at = models.DateTimeField(auto_now_add=True)

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

重新整理 Posts 頁面後，Post 已經被定義成顯示標題：



小結

你現在已經學會：

- 設定 Django Admin
- 建立 superuser
- 註冊 Model 至 Admin

本章新學到的指令

指令	說明
<code>python manage.py createsuperuser</code>	新增 Django 管理者帳號

使用 Django ORM 操作資料庫

在前一章，我們利用 **Django Admin** 新增、修改及刪除 Post。而實際在寫程式時，我們會使用 Django 提供的 QuerySet API，來達成類似的資料庫操作。

本章你會學到：如何使用 Django QuerySet API 與資料庫互動 (CRUD)。

CRUD 指的是，**Create** (新增)、**Read** (讀取)、**Update** (修改)、**Delete** (刪除) 等常見的資料庫操作。

使用 Django Shell

與先前不同的是，在這裡我們不使用 Python Shell，而是 **Django Shell**。

在練習之前，我們先來安裝一個「加強版」的 Python shell：**IPython**。

IPython 是強化版的 Python 互動式命令列介面，它比預設的命令列介面多了許多進階功能，例如：

- 按 `tab` 鍵可以補齊未輸入完的指令、檔案及資料夾名稱。
- 按 `↑` 鍵和 `↓` 鍵可以瀏覽輸入過的程式碼，便於微調先前的程式碼（修改參數等等）。
- 在套件、模組或函數名稱後加上 `?` 可查看與之相關的資訊。
- `history` 指令可查看所有輸入過的指令。
- 可以使用 `shell` 指令，如：`ls`、`cd`。

我們一樣可以用 `pip` 來安裝這個強大的套件：

```
(djangogirls_venv) ~/djangogirls/mysite$ pip install ipython[terminal]
```

安裝完畢後，就可以使用 `shell` 指令，進入 Django Shell：

```
(djangogirls_venv) ~/djangogirls/mysite$ python manage.py shell
```

這個 `shell` 和我們之前輸入 `python` 執行的 `shell` 類似，只是它會預先為我們設定 Django 需要的環境，方便我們執行 Django 相關的程式。

QuerySet API

Create

首先，讓我們來試著新增幾筆資料：

```
>>> from trips.models import Post

>>> Post.objects.create(title='My First Trip', content='肚子好餓，吃什麼好呢?', location='台北火車站')
<Post: My First Trip>

>>> Post.objects.create(title='My Second Trip', content='去散散步吧', location='大安森林公園')
<Post: My Second Trip>

>>> Post.objects.create(title='Django 大冒險', content='從靜態到動態', location='台北市大安區復興南路一段293號')
<Post: Django 大冒險>
```

Read

若想顯示所有的 Post，可以使用 `all()`：

```
>>> from trips.models import Post
>>> Post.objects.all()
[<Post: My First Trip>, <Post: My Second Trip>, <Post: Django 大冒險>]
```

只想顯示部分資料時，則可以使用 `get` 或 `filter`：

```
>>> Post.objects.get(pk=1)
<Post: My First Trip>

>>> Post.objects.filter(location__contains='台北')
[<Post: My First Trip>, <Post: Django 大冒險>]
```

- **get**：返回符合條件的唯一一筆資料。（注意：如果找不到符合條件的資料、或是有多筆資料符合條件，都會產生 exception）
- **filter**：返回符合條件的陣列。如果找不到任何資料則會返回空陣列。

Update

當想修改資料時，可以使用 `update` 更新一筆或多筆資料：

首先，這裡使用 `contains` 針對 `title` 欄位，篩選出所有標題中包含 `Trip` 字眼的 Post

```
>>> posts = Post.objects.filter(title__contains='Trip')
```

注意：Django ORM 會使用雙底線 `__`，來區隔欄位 `title` 和篩選方法 `contains`，如果只用一個底線，Django 會因為找不到欄位 `title_contains` 而出錯。

共有 2 個 Post 符合上面的條件

```
>>> posts
[<Post: My First Trip>, <Post: My Second Trip>]
```

我們將 `location` 的值印出

```
>>> posts[0].location
'台北火車站'

>>> posts[1].location
'大安森林公園'
>>>
```

印出後發現，Post 的 `location` 分別為 '台北火車站' 和 '大安森林公園'。現在我們試試用 `update` 指令，把它們改成 '象山親山步道'

```
>>> posts.update(location='象山親山步道')
2
```

回傳的數字 `2` 指的是已被更新的資料筆數。我們可以驗證一下 `location` 是否皆已被正確更新

```
>>> posts[0].location
'象山親山步道'

>>> posts[1].location
'象山親山步道'
```

Delete

我們也可以使用 `delete` 刪除資料：

我們試著使用 `delete()`，將剛剛的那兩筆 Post 刪除。

```
>>> posts.delete()
```

最後確認一下，資料是否刪除

```
>>> Post.objects.all()  
[<Post: Django 大冒險>]
```

Template tags

在先前的 **Templates** 章節中，我們已經學會基礎的 Django Template 用法 (在 Template 裡呈現變數內容)。但為了產生完整的網頁，我們會需要能在 Template 裡執行一些簡單的 Python 語法，例如：

- 邏輯判斷 (if-else) -- 若使用者已經登入，則顯示使用者的暱稱；若未登入，則顯示登入按鈕
- 重覆 **HTML** 片段 (for loop) -- 列出所有好友的帳號和顯示圖片
- 格式化 **Template** 中的變數 -- 日期的格式化等等

Django template tags 讓你可以在 HTML 檔案裡使用類似 Python 的語法，動態存取從 view function 傳過來的變數，或是在顯示到瀏覽器之前幫你做簡單的資料判斷、轉換、計算等等。

在這一章，我們將使用 Django ORM 存取資料庫，撈出旅遊日記全部的 posts 傳入 template，並使用 Django 的 template tags 與 filters，一步步產生旅遊日記的首頁。

建立旅遊日記的首頁

確認首頁需求

在開始動工之前，我們先確認需求。

旅遊日記的首頁應該會有：

1. 標題
2. 照片
3. 發佈日期
4. 部份的遊記內文

建立首頁的 View

首先我們建立一個新的 view function - `home()` :

```
# trips/views.py

# ...

from django.shortcuts import render
from .models import Post

def home(request):
    post_list = Post.objects.all()
    return render(request, 'home.html', {
        'post_list': post_list,
    })
```

- 匯入所需的 **model** -- 記得 import 需要用到的 Model `Post`
- 取得所有 **posts** -- 透過 `Post.objects.all()` 從資料庫取得全部的 posts，並傳入 `home.html` 這個 template。

設定首頁的 URL

接下來，我們修改 `urls.py`，將首頁（正規表達式 `^$`）指向 `home()` 這個 view function：

```
# mysite/urls.py
from trips.views import hello_world, home

urlpatterns = [
    ...
    url(r'^$', home),
]
```

Template Tags

建立首頁的 Template 並印出 `post_list`

首先，在 `templates` 資料夾底下新增 `home.html`：

```
<!-- home.html -->

{{ post_list }}
```

打開瀏覽器進入首頁 <http://127.0.0.1:8000/>，可以看到 `post_list` 已呈現至網頁上了。



[<Post: Django 大冒險>, <Post: 風和日麗的好天氣>, <Post: 龍山寺一日遊>]

顯示 Post 中的資料

仔細觀察印出的 `post_list`，會發現是以 `list` 的形式顯示。但我們希望的則是：存取每個 `Post` 中的資料，並印出來。

為了達成這個功能，我們會用到 `for` 這個 template tag。

for 迴圈

在寫 Python 時，若想存取 `list` 裡的每一個元素，我們會使用 `for` 迴圈。而在 Django Template 中，也提供了類似的 template tags -- `{% for %}`。

{% for %}

在 template 中使用類似 Python 的 `for` 迴圈，使用方法如下：

```
{% for <element> in <list> %}  
...  
{% endfor %}
```

瞭解了 `for` 的用法後，我們試著印出首頁所需的資訊。修改 `home.html` 如下：

```
<!-- home.html -->

{% for post in post_list %}
    <div>
        {{ post.title }}
        {{ post.created_at }}
        {{ post.photo }}
        {{ post.content }}
    </div>
{% endfor %}
```

- 開始標籤為 `{% for %}` 開始；結束標籤為 `{% endfor %}`
- `post_list` 中有 3 個元素，所以 `for` 區塊中的內容會執行 3 次
- 迴圈中，使用標籤 `{{ var }}`，反覆印出每個 `post` 中的標題、建立時間、照片網址和文章內容

重新整理瀏覽器，網頁上會有首頁所需的 `post` 資訊：



顯示照片

現在網頁已經有照片網址，我們稍微修改 `template`，讓照片以圖片方式呈現。

把 `home.html` 的下面這一行：

```
    {{ post.photo }}
```

換成下面這樣：

```
<div class="thumbnail">
    
</div>
```

處理沒有照片的遊記

if ... else

另一個常用的 template tags 是 `{% if %}` 判斷式，用法如下：

```
{% if post.photo %}
<div class="thumbnail">
    
</div>
{% else %}
<div class="thumbnail thumbnail-default"></div>
{% endif %}
```

- 符合條件所想要顯示的 HTML 放在 `{% if <condition> %}` 區塊裡
- 不符合的則放在 `{% else %}` 區塊裡面
- 最後跟 **for** 一樣，要加上 `{% endif %}` 作為判斷式結尾。

在這裡，我們判斷如果 `post.photo` 有值就顯示照片，否則就多加上一個 CSS class `photo-default` 另外處理。

Template Filter

除了 template tags，Django 也內建也許多好用的 template filters。它能在變數顯示之前幫你做計算、設定預設值，置中、或是截斷過長的內容等等。使用方法如下：

```
{{<variable_name>|<filter_name>:<filter_arguments>}}
```

- `<variable_name>` -- 變數名稱
- `<filter_name>` -- filter 名稱，例如 `add`、`cut` 等等
- `<filter_arguments>` -- 要傳入 filter 的參數

變更時間的顯示格式

在這裡，我們只練習一種很常用的 filter `date`。它可以將 `datetime` 型別的物件，以指定的時間格式輸出。

我們試著將 `created_at` 時間顯示成年 / 月 / 日：

```
{{ post.created_at|date:"Y / m / d" }}
```

完整的 HTML 與 CSS

接著，補上完整的 HTML 標籤，並加上 CSS 樣式後，旅遊日記首頁就完成了。

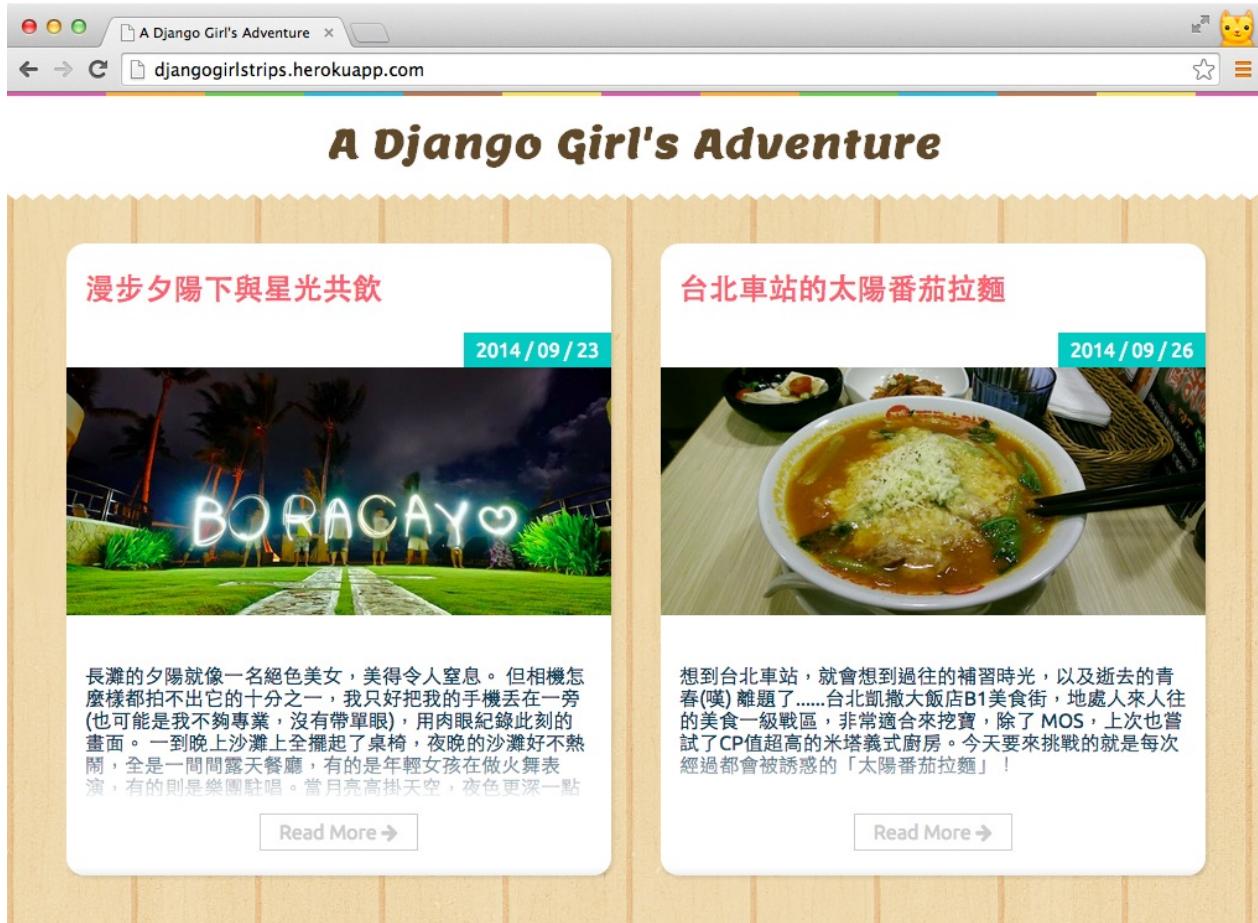
最終版 `home.html` 程式碼如下：

```
<!-- home.html -->

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>A Django Girl's Adventure</title>
    <link href="//fonts.googleapis.com/css?family=Lemon" rel="stylesheet" type="text/css">
    <link href="//maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css" rel="stylesheet" type="text/css">
    <link href="//djangogirlstaipei.github.io/assets/css/style.css" rel="stylesheet" type="text/css">
</head>
<body>
    <div class="header">
        <h1 class="site-title text-center">
            <a href="/">A Django Girl's Adventure</a>
        </h1>
    </div>
    <div class="container">
        {% for post in post_list %}
        <div class="post-wrapper">
            <div class="post">
                <div class="post-heading">
                    <h2 class="title">
                        <a href="#">{{ post.title }}</a>
                    </h2>
                    <div class="date">{{ post.created_at|date:'Y / m / d' }}</div>
                </div>
                {% if post.photo %}
                <div class="thumbnail">
                    
                </div>
                {% else %}
                <div class="thumbnail thumbnail-default"></div>
                {% endif %}
                <div class="post-content read-more-block">
                    {{ post.content }}
                </div>
                <div class="post-footer">
                    <a class="read-more" href="#">
                        Read More <i class="fa fa-arrow-right"></i>
                    </a>
                </div>
            </div>
        {% endfor %}
    </div>
</body>
```

```
</div>
  {% endfor %}
</div>
</body>
</html>
```

打開 <http://127.0.0.1:8000/> 看看你的成果吧！



小結

最後，我們複習一下本章學到的 **Template Tag** 與 **Template Filter**：

Template Tags

語法	說明
<code>{% for ... in ... %}...{% endfor %}</code>	類似 Python 的 for 迴圈，反覆執行 for 區塊中的內容
<code>{% if %} ... {% else %} ... {% endif %}</code>	在 Template Tags 中進行 if／else 的邏輯判斷

Template Filters

語法	說明
<code>{{ value date: <date_format> }}</code>	可以將`datetime`型別的物件，以指定的時間格式 Date Format 輸出

Dynamic URL

除了在首頁顯示文章的摘要外，通常也會希望每篇文章能有獨立的網址與頁面。例如，我們可能會希望 `http://127.0.0.1/post/3/` 能夠是 pk 為 3 那篇文章的網址，而頁面內容則是此篇日記的詳細資訊，而非摘要。

在這個章節，我們會學到如何設定動態網址的 URL conf，讓每篇旅遊日記，擁有獨一無二的網址與頁面。

建立單篇文章的 View

首先建立單篇文章所使用的 view function。在 `trips/views.py` 中，新增 `post_detail` 這個 view 如下：

```
# trips/views.py

# ...

from django.shortcuts import render
from .models import Post

def post_detail(request, pk):
    post = Post.objects.get(pk=pk)
    return render(request, 'post.html', {'post': post})
```

以訪客瀏覽 `http://127.0.0.1:8000/post/3/` 的例子，來解釋以上程式：

- 目前瀏覽文章的 `pk` 會傳入 `view` 中：當訪客瀏覽 `http://127.0.0.1/post/3/` 時，傳入 `view` 的 `pk` 會是 3。
 - URL 與 `pk` 的對應，會在稍後設定。這裡只需知道 `view` 中傳入的，會是當前瀏覽文章 `pk` 即可。
- 取得傳入 `pk` 的那篇 `Post` 資料：當傳入的 `pk=3`，代表訪客想看到 `pk=3` 那篇文章。我們可以利用之前學過的 ORM 語法 `get`，取得該篇日記的 `Post` 物件：

```
post = Post.objects.get(pk=pk) # 此時 pk = 3
```

- 回傳 **HttpResponse**：將取得的 post (pk=3) 傳入 Template *post.html*，並呈現 render 後的結果。

設定動態網址的對應

日記單頁的 view function 完成後，我們來設定網址與 view 的對應。修改 *mysite/urls.py*，加入以下內容：

```
# mysite/urls.py
from trips.views import hello_world, home, post_detail

urlpatterns = [
    ...
    url(r'^post/(?P<pk>\d+)/$', post_detail, name='post_detail'),
]
```

上面的修改完成後，只要連至 `http://127.0.0.1/post/3/` 就會對應到 `post_detail()` 這個 view，並且傳入的 `pk=3`。

使用 Regex 提取部份 URL 為參數

我們前面提過，Django 的 URL 是一個 *regular expression (regex)*。Regular expression 語法可用來描述一個字串的樣式。除了可以表示固定字串之外，還可以用來表示不確定的內容。我們一步一步解釋文章單頁所使用的 URL 設定：

`(?P<pk>\d+)`

1. `\d` 代表一個阿拉伯數字。

2. `+` 代表「一個以上」。

所以 `\d+` 代表一個以上的阿拉伯數字，例如「0」、「99」、「12345」。可是像「8a」就不符合，因為「a」不是數字。

3. `(?P<pk>)` 代表「把這一串東西抓出來，命名為 pk。」

所以 `(?P<pk>\d+)` 代表：抓出一個以上阿拉伯數字，並把抓出來的東西取名為 pk。

綜合以上的規則，`r'^post/(?P<pk>\d+)/$'` 會達成以下的效果：

URL	符合結果
<code>http://127.0.0.1/posts/</code>	不符合，因為前面不是 <code>post/</code> 開頭。
<code>http://127.0.0.1/post/</code>	不符合，因為後面抓不到數字。
<code>http://127.0.0.1/post/1/</code>	符合，抓到的 pk 是 1。
<code>http://127.0.0.1/post/1234/</code>	符合，抓到的 pk 是 1234。
<code>http://127.0.0.1/post/12ab/</code>	不符合，因為後面有不是數字的東西。

建立單篇日記頁的 Template

回顧一下前面寫的 view function (`post_detail`) 內容

```
return render(request, 'post.html', {'post': post})
```

我們取得所需 `post` 物件後，傳入 `post.html` 這個 template 中 `render`。現在我們就來完成這個 template。建立 `post.html` 如下：

```
<!-- templates/post.html -->

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>{{ post.title }} | A Django Girl's Adventure</title>
    <link href="//fonts.googleapis.com/css?family=Lemon" rel="stylesheet" type="text/css">
        <link href="//maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css" rel="stylesheet" type="text/css">
            <link href="//djangogirlstaipei.github.io/assets/css/style.css" rel="stylesheet" type="text/css">
</head>
<body>
    <div class="header">
        <h1 class="site-title text-center">
            <a href="/">A Django Girl's Adventure</a>
        </h1>
    </div>
    <div class="container post post-detail">
        <div class="post-heading">
            <h1 class="title">{{ post.title }}</h1>
            <div class="date">{{ post.created_at|date:'Y / m / d' }}</div>
        </div>
        <div class="location">
            <i class="fa fa-map-marker"></i>
            <span id="location-content">{{ post.location }}</span>
        </div>
        <div id="map-canvas" class="map"></div>
        <div class="post-content">
            {{ post.content }}
        </div>
        <hr class="fancy-line">
        
    >
    </div>
    <script src="//maps.googleapis.com/maps/api/js?v=3.exp&libraries=places&sensor=false"></script>
    <script src="//djangogirlstaipei.github.io/assets/js/map.js"></script>
</body>
</html>
```

這個 template 將 post 物件的屬性 (e.g. 標題、內文、時間.....等)，利用 `{{ var }}` 與 template filter 顯示並格式化於 HTML 中。若資料庫裡有 pk=3 的 Post，現在連至 <http://127.0.0.1:8000/post/3/> 即可看到此日記的單頁。

{% url %}

連結到特定 view 的 template tag

使用方法：

語法	說明
{% url '<url_name>' %}	根據在 urls.py 中設定的「name」值，找到對應的 URL

也可以傳入參數，如：

```
{% url '<url_name>' arg1=<var1> arg2=<var2> ...%}
```

其餘用法可參考[官方文件](#)。

加入到單篇日記頁的連結

最後，我們還需在首頁加上單篇日記的連結。我們可以使用 `{% url %}` 這個 template tag 達成。需要加入的地方有：

1. 每篇日記
2. 每篇日記的 Read More 按鈕

設定標題連結

打開 `home.html`，找到下面的內容：

```
<!-- home.html -->

<h2 class="title">
    <a href="#">{{ post.title }}</a>
</h2>
```

將它改成

```
<!-- home.html -->

<h2 class="title">
    <a href="{% url 'post_detail' pk=post.pk %}">{{ post.title }}</a>
</h2>
```

設定 **Read More** 按鈕的連結

在 *home.html* 中找到以下內容：

```
<!-- home.html -->

<a class="read-more" href="#">  
    Read More <i class="fa fa-arrow-right"></i>  
</a>
```

修改如下：

```
<!-- home.html -->

<a class="read-more" href="{% url 'post_detail' pk=post.pk %}">  
    Read More <i class="fa fa-arrow-right"></i>  
</a>
```

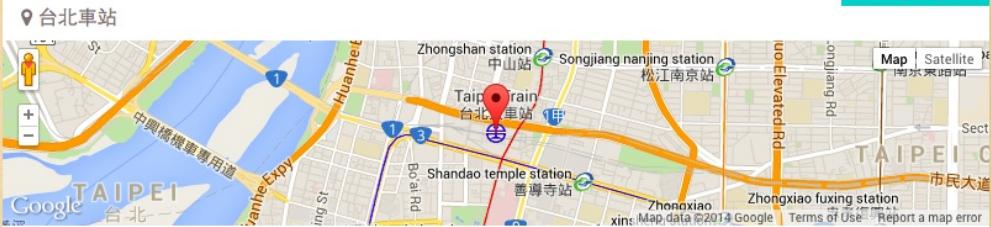
驗收成果

連至 <http://127.0.0.1:8000/>。現在只要點擊各個日記的標題或 **Read More** 按鈕，就會顯示那該篇日記的詳細頁面。

A Django Girl's Adventure

台北車站的太陽番茄拉麵

2014 / 09 / 26



想到台北車站，就會想到過往的補習時光，以及逝去的青春(嘆) 離題了.....台北凱撒大飯店B1美食街，地處人來人往的美食一級戰區，非常適合來挖寶，除了 MOS，上次也嘗試了CP值超高的米塔義式廚房。今天要來挑戰的就是每次經過都會被誘惑的「太陽番茄拉麵」！



Deploy

目前為止，我們所有的工作都是在自己的電腦完成，你也可以在自己的電腦上看到成果。但是，如果我們想要讓其他人隨時瀏覽這個網站，就必須將它部署（deploy）到穩定的伺服器上。

我們選擇 [PythonAnywhere](#) 作為範例。它對於 Python 的支援性相當好，免費帳號也足夠經營一個小型網站。

部署準備

為了將你的程式碼上傳到雲端，我們要先將整個專案打包成一個壓縮檔。在 `djangogirls` 專案目錄下，用以下的指令將整個專案壓縮成 `mysite.zip`：

```
(djangogirls_venv) ~/djangogirls$ python -m zipfile -c mysite.zip mysite
```

部署到雲端

在開始部署之前，請先確定已經在 [PythonAnywhere](#) 註冊帳號：

The screenshot shows the PythonAnywhere 'Plans and pricing' section. It features a 'Beginner: Free!' plan with a yellow background, a 'Startup' plan with a green background, and a 'Custom' plan with a grey background. The 'Education accounts' section is also visible.

Plan	Cost	Description
Hacker	\$5/month	Run your Python code in the cloud from one web app and the console
Web dev	\$12/month	If you want to host small Python-based websites for you or for your clients
Startup	\$99/month	Start a business and don't worry about having to scale to handle traffic spikes
Custom	\$5 to \$500/month	Want a combination that's not on the list? Create your own! All custom plans have:

Step 1: 上傳專案

使用 PythonAnywhere 的檔案介面，將專案的壓縮檔上傳。首先切換到 **Files** 分頁，按下 **Upload a file** 按鈕：

The screenshot shows the PythonAnywhere 'Files' interface. At the top, there are tabs for Consoles, Files, Web, Schedule, and Databases. The 'Files' tab is selected. Below the tabs, the URL is /> home > djangogirlstaipei. On the left, there's a 'Directories' section with a 'New directory' button. On the right, there's a 'Files' section with a search bar 'Enter new file name, eg hello.py' and a 'New file' button. A list of files is shown, including .bashrc, .gitconfig, .profile, .pythonstartup.py, .vimrc, and README.txt. At the bottom, there's a red-bordered 'Upload a file' button.

選擇剛剛壓縮好的 `mysite.zip`。當畫面上出現這個檔案時，便表示已經上傳完成。

The screenshot shows the PythonAnywhere 'Files' interface after the upload. The 'Directories' and 'Files' sections are identical to the previous screenshot, but the file list now includes 'mysite.zip' at the bottom. The file details show it was uploaded on 2016-07-02 at 09:33 with a size of 18.7 KB. The 'Upload a file' button is still present at the bottom.

Step 2: 開啓 Bash Console

切換到 **Consoles** 分頁，點選 **Bash** 開啓一個新的 Bash console。我們可以透過它下指令，以建置部署環境。

Start a new console:

Python: 3.5 / 3.4 / 3.3 / 2.7 / 2.6 IPython: 3.5 / 3.4 / 3.3 / 2.7 / 2.6 PyPy: 2.7
Other: **Bash** | MySQL
Custom:

3% used: 3.18s of your 100 second CPU allowance ([more info](#))
Allowance resets in 4 hours, 21 minutes

Your consoles:
You have no consoles. Click a link above to start one.

Start a **console** using the links above. If you're not sure which one you want, we recommend Python 2.7. These are real consoles running on PythonAnywhere servers, with lots of **batteries included**. Bash shells give access to a full GNU/Linux environment including vim and emacs. Consoles can be shared with other users, have Internet access (filtered for free users, full access for **paying customers**) and do not lose state if you close your browser window.

The **Files** tab provides basic file management, scripts can be run and text files can be edited.

Inside the **Web** tab you can configure web apps, which will be served at <http://djangogirlstaipei.pythonanywhere.com/> — or, if you have a paid plan, at any other domain you own. Try using one of the built in web frameworks like Django, Flask, or web2py to get started quickly. If you're more adventurous, you can use any web framework that supports the WSGI protocol.

The **Schedule** tab configures scripts, like scrapers or notifications, that need to be run periodically.

The **Databases** tab lets you set up access to MySQL or Postgres databases. MySQL is available for all accounts, free or paid, while Postgres is a paid feature.

首先利用 `unzip` 指令將專案解壓縮：

```
~ $ unzip mysite.zip
```

Bash console 3115854

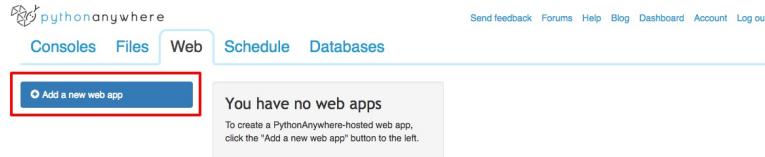
```
09:46 ~ $ ls
README.txt  mysite.zip
09:47 ~ $ unzip mysite.zip
Archive: mysite.zip
  creating: mysite/
  inflating: mysite/db.sqlite3
  inflating: mysite/manage.py
  creating: mysite/mysite/
extracting: mysite/mysite/__init__.py
  creating: mysite/mysite/__pycache__/
  inflating: mysite/mysite/__pycache__/__init__.cpython-35.pyc
  inflating: mysite/mysite/__pycache__/settings.cpython-35.pyc
  inflating: mysite/mysite/__pycache__/urls.cpython-35.pyc
  inflating: mysite/mysite/__pycache__/wsgi.cpython-35.pyc
  inflating: mysite/mysite/settings.py
  inflating: mysite/mysite/urls.py
  inflating: mysite/mysite/wsgi.py
  creating: mysite/templates/
  inflating: mysite/templates/hello_world.html
  inflating: mysite/templates/home.html
  inflating: mysite/templates/post.html
  creating: mysite/trips/
extracting: mysite/trips/__init__.py
  creating: mysite/trips/__pycache__/
  inflating: mysite/trips/__pycache__/__init__.cpython-35.pyc
  inflating: mysite/trips/__pycache__/admin.cpython-35.pyc
  inflating: mysite/trips/__pycache__/models.cpython-35.pyc
  inflating: mysite/trips/__pycache__/views.cpython-35.pyc
  inflating: mysite/trips/admin.py
  creating: mysite/trips/migrations/
  inflating: mysite/trips/migrations/0001_initial.py
extracting: mysite/trips/migrations/__init__.py
```

由於雲端的環境與我們本機端不同，我們需要為它建立一個新的虛擬環境，並在裡面安裝 Django：

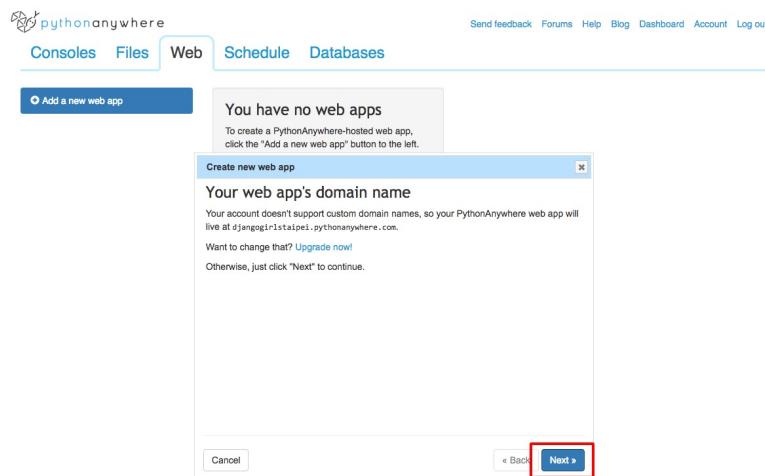
```
~ $ virtualenv --python=python3.5 djangogirls_venv
~ $ source djangogirls_venv/bin/activate
(djangogirls_venv) ~ $ pip install "django<1.9"
```

Step 3: 新增一個新的 web app

點擊左上角的 logo 回到主頁面，接著切換到 Web 分頁，點選 Add a new web app 按鈕：

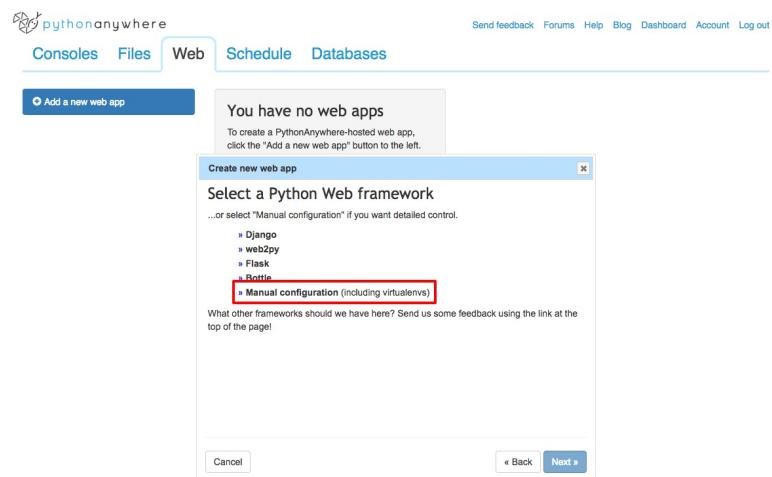


在出現的視窗中首先會詢問你想使用的網址。由於免費方案無法設定，我們直接按下 Next 使用預設值。

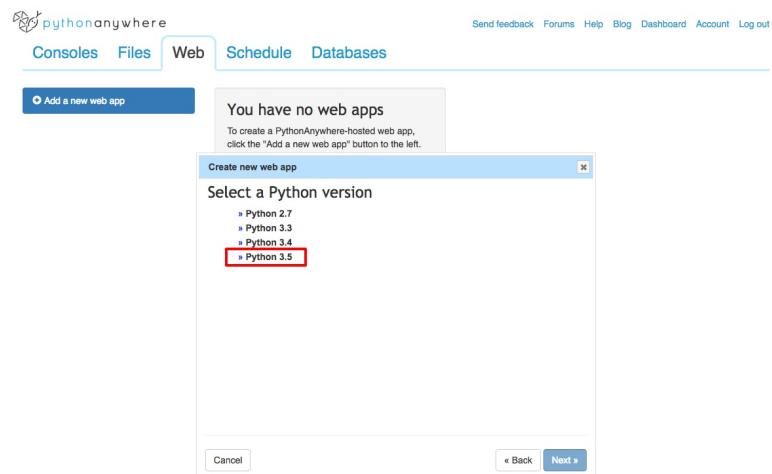


接著需要選擇 Python Web framework。請特別注意不要選擇 **Django**。我們將手動設定虛擬環境，因此使用 *Manual configuration*。

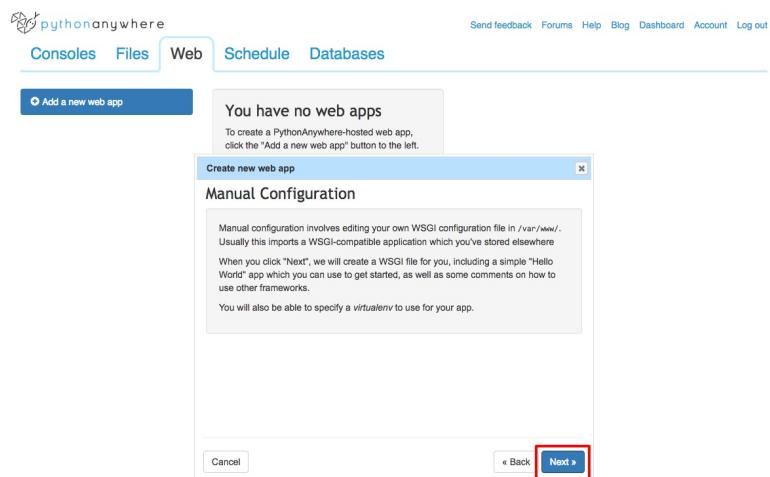
Deploy



在 Python 版本選擇畫面中使用 *Python 3.5*。



最後按下 *Next*，即完成建立 web app。



Step 4: 修改 web app 設定

當你完成上一個步驟後，Web 分頁會顯示 web app 的設定介面，讓你可以修改設定、重載 web app、或是查看錯誤訊息等等。

The screenshot shows the PythonAnywhere web interface for managing a web application named 'djangogirlstaipei.pythonlywhere.com'. The interface includes sections for Configuration, Best before date, Traffic, Code, Virtualenv, Log files, and Static files. A red box highlights the 'Enter path to a virtualenv, if desired' input field.

Configuration for djangogirlstaipei.pythonlywhere.com

Best before date:
Free sites have a limited lifespan, but you can renew that here up to a maximum of three months from today's date. You can always extend it later too! We'll send you an email a week before it expires. [See here for more details.](#)

This site will be disabled on **Sunday 02 October 2016**

Traffic:
How busy is your site?
This month (previous month) 16 (31)
Today (yesterday) 16 (0)
Hour (previous hour) 0 (0)

Code:
What your site is running.
Source code: [Enter the path to your web app source code](#)
Working directory: [/home/djangogirlstaipei/](#) [Go to directory](#)
WSGI configuration file: [/var/www/djangogirlstaipei_pythonanywhere_com_wsgi.py](#)
Python version: 3.5

Virtualenv:
Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here.](#) You need to Reload your web app to activate it; NB - will do nothing if the virtualenv does not exist.
Enter path to a virtualenv, if desired

Log files:
The first place to look if something goes wrong.
Access log: [/djangogirlstaipei.pythonlywhere.com.access.log](#)
Error log: [/djangogirlstaipei.pythonlywhere.com.error.log](#)
Server log: [/djangogirlstaipei.pythonlywhere.com.server.log](#)

Static files:
Files that aren't dynamically generated by your code, like CSS, JavaScript or uploaded files, can be served much faster straight off the disk if you specify them here. You need to Reload your web app to activate any changes you make to the mappings below.

URL	Directory	Delete
Enter URL	Enter path	

Password protection:
Ideal for sites that are under development when you don't want anyone to see them yet. You need to Reload your web app to activate any changes made here.

Protection: Disabled
Username: [Enter a username](#)
Password: [Enter a password](#)

Delete:
Deleting this web app will remove your web app from the Internet, but will not remove your code. The WSGI configuration file will be backed up.
[Delete djangogirlstaipei.pythonlywhere.com](#)

首先在中間的 **Virtualenv** 區塊填入我們前面設定的虛擬環境位置 `/home/<your-PythonAnywhere-username>/djangogirls_venv`，並點選按鈕儲存設定。

注意：請將 `<your-PythonAnywhere-username>` 改成你在 PythonAnywhere 註冊的 `username`。我們在後面還會不斷提到這個路徑，務必也要記得修改！

Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here.](#) You need to Reload your web app to activate it; NB - will do nothing if the virtualenv does not exist.

Enter path to a virtualenv, if desired

接著設定 PythonAnywhere 與我們的網站如何溝通。用 Python 撰寫的網站程式是透過 Python 制定的 **WSGI - Web Server Gateway Interface** 標準，來定義和伺服器溝通的介面。在 **Code** 區塊點選 `/var/www/<your-PythonAnywhere-username>_PythonAnywhere_com_wsgi.py`，打開編輯介面：

Code:

What your site is running.

Source code:	<i>Enter the path to your web app source code</i>
Working directory:	/home/djangogirlstaipei/
WSGI configuration file:	/var/www/djangogirlstaipei_pythonanywhere_com_wsgi.py
Python version:	3.5

↗ Go to directory

刪除原有的程式碼，並覆蓋為：

```
import os
import sys

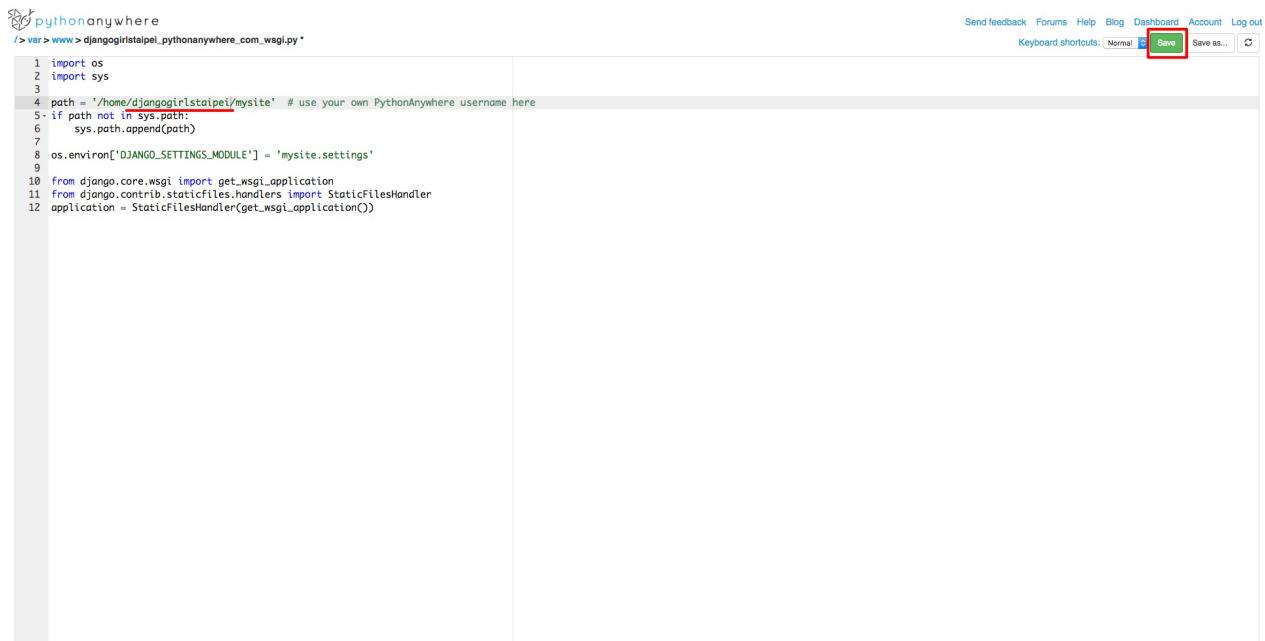
path = '/home/<your-PythonAnywhere-username>/mysite' # use your own PythonAnywhere us
ername here
if path not in sys.path:
    sys.path.append(path)

os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'

from django.core.wsgi import get_wsgi_application
from django.contrib.staticfiles.handlers import StaticFilesHandler
application = StaticFilesHandler(get_wsgi_application())
```

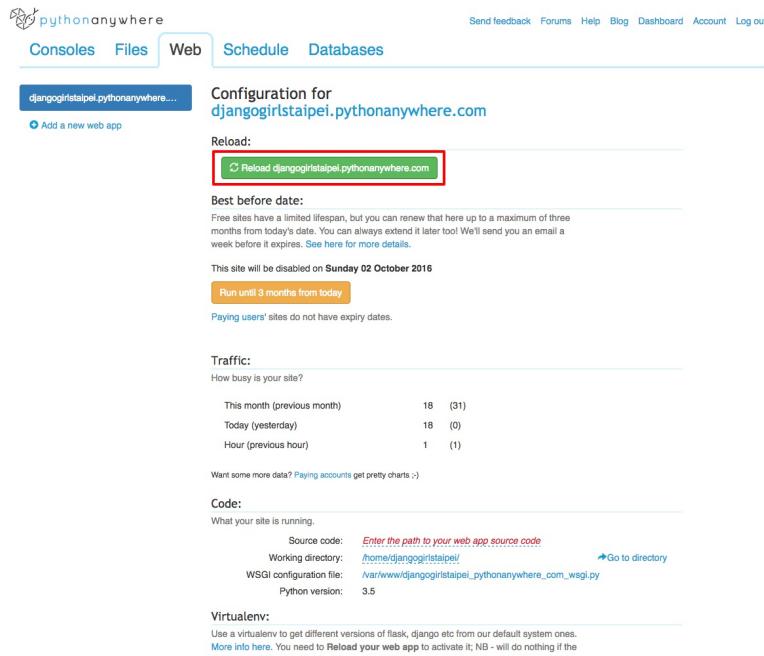
再次提醒，記得要將 `<your-PythonAnywhere-username>` 改成你在 PythonAnywhere 註冊的 `username`。

點選 Save 儲存修改後，即可點選 logo 離開編輯器。



Step 5: 重新載入（Reload）web app

回到 Web 分頁點選 *Reload* 按鈕，讓 PythonAnywhere 重新載入我們更新的設定：

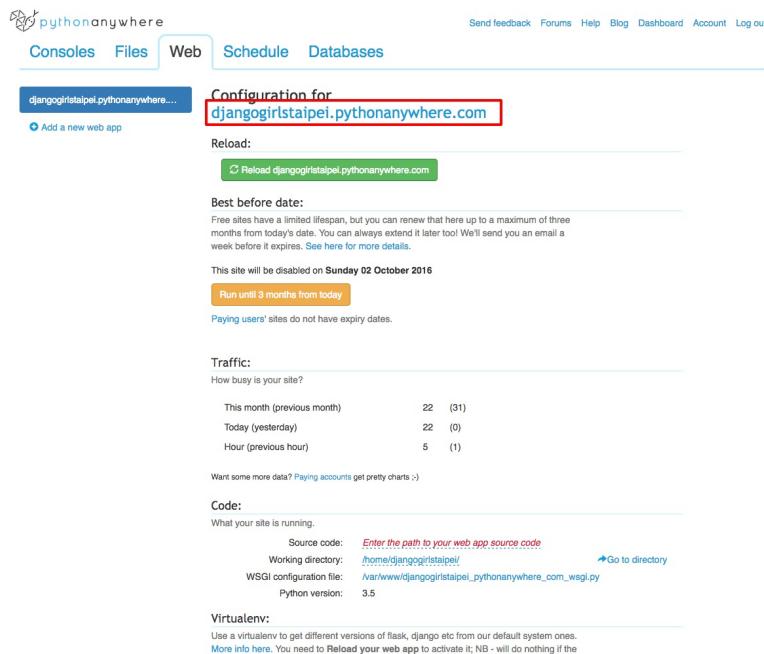


The screenshot shows the PythonAnywhere web configuration interface for the site `djangogirlstaipei.pythonanywhere.com`. At the top, there are tabs for Consoles, Files, Web, Schedule, and Databases. The Web tab is selected. Below the tabs, there's a section titled "Configuration for `djangogirlstaipei.pythonanywhere.com`". A green button labeled "Reload" is highlighted with a red box. Other buttons include "Run until 3 months from today" and "Paying users' sites do not have expiry dates". There are sections for "Best before date", "Traffic" (with a table showing data for the month, yesterday, and hour), "Code" (with source code path, working directory, WSGI config file, and Python version), and "Virtualenv" (with a note about using a virtualenv). The "Traffic" section table data is as follows:

Time Period	Visitors	Unique Visitors
This month (previous month)	18	(31)
Today (yesterday)	18	(0)
Hour (previous hour)	1	(1)

Step 6: 開啓瀏覽器觀看你的網站

直接點選，或可以複製網址到瀏覽器打開網站：



The screenshot shows the PythonAnywhere web configuration interface for the site `djangogirlstaipei.pythonanywhere.com`. The layout is identical to the previous screenshot, with tabs for Consoles, Files, Web, Schedule, and Databases, and the Web tab selected. The "Configuration for `djangogirlstaipei.pythonanywhere.com`" section includes a green "Reload" button highlighted with a red box. The "Traffic" section table data is as follows:

Time Period	Visitors	Unique Visitors
This month (previous month)	22	(31)
Today (yesterday)	22	(0)
Hour (previous hour)	5	(1)

大功告成，恭喜你成功發佈了自己的網站！這個網址可以直接分享給任何人：<https://djangogirlstaipei.PythonAnywhere.com/>——記得前面要替換成你自己的 PythonAnywhere username !

未來如果對網站進行任何修改，可以使用以下的步驟更新：

1. 壓縮專案、上傳
2. 用 Bash console 解壓縮
3. 重新載入（Reload）web app

就可以在 PythonAnywhere 上看到新的程式。

注意：在上傳、解壓縮新版程式時，也會讓雲端的所有資料庫被本機的版本覆蓋！

What's next?

恭喜! 你已經懂得如何使用 Django 寫出自己的網站，並發佈到網路上了。

接下來，我們希望你能試著：

- 修改 HTML 與 CSS，調整成你喜歡的樣子
- 為旅遊日記添加新的欄位（例如旅遊日期），並使用 `makemigrations` 和 `migrate` 更新資料庫。
- 為旅遊日記加入作者（提示：你可能會需要修改 Model，並與 Django 使用者認證功能整合）
- 將 HTML 重複的部分獨立出來共用（提示：使用 Template 繼承）
- 為每一篇日記加上留言板（提示：Django Forms 可以幫助你更快速地完成）

其他學習資源

- [Codecademy](#) -- 透過闖關遊戲方式學習 Python, HTML/CSS, JavaScript
- [Writing your first Django app](#) -- Django 1.8 官方學習指南
- [Getting Started With Django](#) -- 影片課程
- [The Django Book](#) -- 雖然 Django 版本不是最新，但相當適合初學者的一本書
- [Two Scoops of Django: Best Practices for Django](#) -- 非常推薦，曾經在 [Taipei.py](#) 隔週二聚會指定書籍
- [Django Packages](#) -- Django 相關套件彙整平台，提供搜尋和評比

關注我們的最新消息請至：

- Django Girls Taipei 網站：<http://djangogirls.org/taipei>
- Django Girls Taiwan 臉書社團：<https://www.facebook.com/groups/djangogirls.taiwan/>
- 如果有任何問題，歡迎來信與我們聯繫：taipei@djangogirls.org