編寫你的第一個 Django 應用，第 1 部分

讓我們透過範例來學習。

透過這個教學，我們將帶著你建立一個基本的投票應用程式。

它將由兩部分組成：

- 一個讓人們查看和投票的公開網站。
- 一個讓你能增加、修改和刪除投票的管理網站。

我們假定你已經閱讀了 安裝 Django。你能知道 Django 已被安裝，且安裝的是哪個版本，透過在命令提示欄輸入命令（由 $ 前置符號）。

```
$ python -m django --version
```

如果這行命令輸出了一個版本號碼，證明你已經安裝了此版本的 Django；如果你得到的是一個“No module named django”的錯誤提示，則表示你尚未安裝。

這個教學是為了 Django 3.0 寫的，它支援 Python 3.6 和後續版本。如果 Django 的版本不比對，你可以透過頁面右下角的版本切換器切換到對應你版本的教學，或更新至最新版本。如果你正在使用一個較舊版本的 Python，在 我應該使用哪個版本的 Python 來配合 Django? 查找一個合適的 Django 版本。

你可以查看文件 如何安裝 Django 來取得關於移除舊版本，安裝新版本的流程和建議。

從哪裡取得協助：

如果你在閱讀本教學的過程中有任何疑問，可以前往FAQ的:doc:Getting Help</faq/help> 的版塊。

建立專案

如果這是你第一次使用 Django 的話，你需要一些初始化設置。也就是說，你需要用一些自動產生的程式設定一個 Django 專案 —— 即一個 Django 專案實例需要的設定項集合，套件括資料庫設定、Django 設定和應用程式設定。

打開命令列，**cd** 到一個你想放置你程式的目錄，然後執行以下命令：

```
$ django-admin startproject mysite
```

這行程式將會在當前目錄下建立一個 **mysite** 目錄。如果命令失敗了，查看 執行 django-admin 時遇到的問題，可能能給你提供協助。

注解

你得避免使用 Python 或 Django 的內部保留字來命名你的專案。具體地說，你得避免使用像 **`django`** (會和 Django 自己產生衝突)或 **`test`** (會和 Python 的內置組件產生衝突)這樣的名字。

我的程式該放在哪?

如果你曾經是原生 PHP 程式設計師（沒有使用過現代框架），你可能會習慣於把程式放在 Web 伺服器的文件根目錄(諸如 **`/var/www`**)。當使用 Django 時不需要這樣做。把所有 Python 程式放在 Web 伺服器的根目錄不是個好主意，因為這樣會有風險。比如會提高人們在網站上看到你的程式的可能性。這不利於網站的安全。

把你的程式放在文件根目錄 **以外** 的某些地方吧，比如 /home/mycode。

讓我們看看 **`startproject`** 建立了些什麼:

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

這些目錄和文件的用處是:

- 最外層的 **`mysite/`** 根目錄只是你專案的容器， 根目錄名稱對Django沒有影響，你可以將它重命名為任何你喜歡的名稱。
- **`manage.py`**: 一個讓你用各種方式管理 Django 專案的命令列工具。你可以閱讀 django-admin and manage.py 獲取所有 **`manage.py`** 的細節。
- 裡面一層的 **`mysite/`** 目錄套件含你的專案，它是一個純 Python 套件。它的名字就是當你引用它內部任何東西時需要用到的 Python 套件名。 (比如 **`mysite.urls`**).
- **`mysite/__init__.py`**：一個空文件，告訴 Python 這個目錄應該被認為是一個 Python 套件。如果你是 Python 初學者，閱讀官方文件中的 更多關於套件的知識。
- **`mysite/settings.py`**：Django 專案的設定文件。如果你想知道這個文件是如何運作的，請查看 Django 設定 了解細節。
- **`mysite/urls.py`**：Django 專案的 URL 聲明，就像你網站的“目錄”。閱讀 URL調度器 文件來獲取更多關於 URL 的內容。
- **`mysite/asgi.py`**：作為你的專案的執行在 ASGI 相容的Web伺服器上的入口。閱讀 如何使用 WSGI 進行部署 了解更多細節。

- **mysite/wsgi.py**：作為你的專案的執行在 WSGI 相容的Web伺服器上的入口。閱讀 如何使用 WSGI 進行部署 了解更多細節。

用於開發的簡易伺服器

讓我們來確認一下你的 Django 專案是否真的建立成功了。如果你的當前目錄不是外層的 **mysite** 目錄的話，請切換到此目錄，然後執行下面的命令：

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

你應該會看到如下輸出：

```
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.

七月 23, 2020 - 15:50:53
Django version 3.0, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

注解

忽略有關未套用最新資料庫遷移的警告，稍後我們會處理資料庫。

你剛剛啟動的是 Django 自帶的用於開發的簡易伺服器，它是一個用純 Python 寫的輕量級的 Web 伺服器。我們將這個伺服器內置在 Django 中是為了讓你能快速的開發出想要的東西，因為你不需要進行設定生產級別的伺服器（比如 Apache）方面的工作，除非你已經準備好投入生產環境了。

現在是個提醒你的好時機：**千萬不要** 將這個伺服器用於和生產環境相關的任何地方。這個伺服器只是為了開發而設計的。(我們在 Web 框架方面是專家，在 Web 伺服器方面並不是。)

現在，伺服器正在執行，瀏覽器開啟 https://127.0.0.1:8000/。你將會看到一個“恭喜”頁面，隨著一只火箭發射，伺服器已經執行了。

更換連接埠

預設情況下，**runserver** 命令會將伺服器設置為監聽本機內部 IP 的 8000 連接埠。

如果你想更換伺服器的監聽連接埠，請使用命令列參數。舉個例子，下面的命令會使伺服器監聽 8080 連接埠：

```
$ python manage.py runserver 8080
```

如果你想要修改伺服器監聽的IP，在連接埠之前輸入新的。比如，為了監聽所有伺服器的公開IP（在你執行 Vagrant 或想要向網絡上的其它電腦展示你的成果時很有用），使用：

```
$ python manage.py runserver 0:8000
```

0 是 **0.0.0.0** 的簡寫。完整的關於開發伺服器的文件可以在 :djamdin:`runserver` 參考文件中找到。

會自動重新載入的伺服器 `runserver`

用於開發的伺服器在需要的情況下會對每一次的開啟請求重新載入一遍 Python 程式。所以你不需要為了讓修改的程式生效而頻繁的重新啟動伺服器。然而，一些動作，比如增加新文件，將不會觸發自動重新載入，這時你得自己手動重啟伺服器。

建立投票應用程式

現在你的開發環境——這個“專案”——已經設定好了，你可以開始作業了。

在 Django 中，每一個應用都是一個 Python 套件，並且遵循著相同的約定。Django 自帶一個工具，可以幫你產生應用程式的基礎目錄結構，這樣你就能專心寫程式，而不是建立目錄了。

專案 VS 應用程式

專案和應用程式有什麼區別？應用程式是一個專門做某件事的網絡應用程式——比如部落格系統，或者公享記錄的資料庫，或者小型的投票程式。專案則是一個網站使用的設定和應用程式的集合。專案可以套件含很多個應用程式。應用程式可以被很多個專案使用。

你的應用程式可以存放在任何 Python path 中定義的路徑。在這個教學中，我們將在你的 **`manage.py`** 同階層目錄下建立投票應用程式。這樣它就可以作為頂層模組匯入，而不是 **`mysite`** 的子模組。

請確定你現在處於 **`manage.py`** 所在的目錄下，然後執行這行命令來建立一個應用程式：

```
$ python manage.py startapp polls
```

這將會建立一個 **polls** 目錄，它的目錄結構大致如下：

```
polls/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

這個目錄結構套件括了投票應用程式的全部內容。

編寫第一個視圖

讓我們開始編寫第一個視圖吧。打開 **polls/views.py**，把下面這些 Python 程式輸入進去：

polls/views.py

```
from django.http import HttpResponse


def index(request):
    return HttpResponse("Hello, world. You're at the polls index.")
```

這是 Django 中最簡單的視圖。如果想看見效果，我們需要將一個 URL 對映到它——這就是我們需要 URLconf 的原因了。

為了建立 URLconf，請在 polls 目錄裡新建一個 **urls.py** 文件。你的應用程式目錄現在看起來應該是這樣：

```
polls/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    urls.py
    views.py
```

在 **polls/urls.py** 中，輸入如下程式碼：

polls/urls.py

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

下一步是要在根 URLconf 文件中指定我們建立的 **polls.urls** 模組。在 **mysite/urls.py** 文件的 **urlpatterns** 欄表裡插入一個 **include()**， 如下：

mysite/urls.py

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

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

函數 **include()** 允許引用其它 URLconfs。每當 Django 遇到 **include()** 時，它會將 URL 與函式參數指定字串比對符合的部分截斷，並將剩餘的字串傳送到 URLconf 以供進一步處理。

我們設計 **include()** 的理念是使其可以即插即用。因為投票應用程式有它自己的 URLconf(**polls/urls.py**)，他們能夠被放在 "/polls/"， "/fun_polls/"，"/content/polls/"，或者其他任何路徑下，這個應用程式都能夠正常工作。

何時使用 include()

當套件括其它 URL 模式時你應該總是使用 **include()**， **admin.site.urls** 是唯一例外。

你現在把 **index** 視圖增加進了 URLconf。透過以下命令驗證是否正常工作：

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

用你的瀏覽器開啟 http://localhost:8000/polls/，你應該能夠看見 "*Hello, world. You're at the polls index.*"， 這是你在 **index** 視圖中定義的。

無法找到頁面 Page Not Found?

如果你在這裡得到了一個錯誤頁面，檢查一下你是不是開啟 http://localhost:8000/polls/ 頁面而不是 http://localhost:8000/。

函數 **path()** 具有四個參數，兩個必要參數：**route** 和 **view**，兩個可選參數：**kwargs** 和 **name**。現在，是時候來研究這些參數的含義了。

path() 參數：route

route 是一個包含 URL 模式的字串（類似正規表達式）。當 Django 回應一個請求時，它會從 **urlpatterns** 清單的第一項開始，依序比對清單中的每一個項目，直到找到比對符合的項目為止。

這些模式不會比對 GET 和 POST 參數或網域名稱。例如，URLconf 在處理請求 **https://www.example.com/myapp/** 時，它會嘗試比對 **myapp/** 。在處理 **https://www.example.com/myapp/?page=3** 請求時，也只會嘗試比對 **myapp/**。

---

path() 參數：view

當 Django 找到了一個比對符合的模式，就會呼叫這個特定的視圖函數，並傳入一個 **HttpRequest** 物件作為第一個參數，被 “捕獲” 的參數以關鍵字參數的形式傳入。稍後，我們會提供一個範例。

---

path() 參數：kwargs

任意個關鍵字參數可以作為一個字典傳遞給目標視圖函數。本教學中不會使用這一特性。

---

path() 參數：name

為你的 URL 取名能使你在 Django 的任意地方唯一地引用它，尤其是在範本中。這個有用的特性允許你只改一個文件就能全域地修改某個 URL 模式。

當你了解了基本的請求和回應流程後，請閱讀 教學的第 2 部分 開始使用資料庫.

‹ 快速安裝指南 編寫你的第一個 Django 應用，第 2 部分 ›