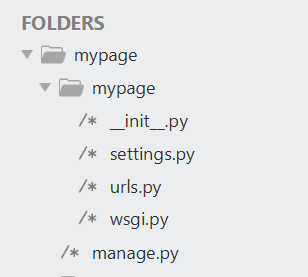
#Cmd django-admin startproject mysite(想建立的資料夾名字)



\_\_setting\_\_.py : 項目設置配置/包含一些初始化的設置

\_\_urls\_\_.py : url模式存放的地方，每一個url都map到一個視圖(view)

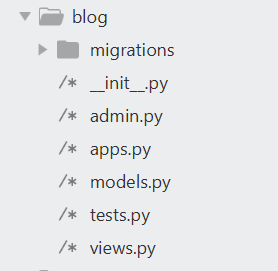
Wsgi.py : 配置項目運行，如同一個wsgi應用

#Cd mysite python manage.py migrate //先初始化

#settings 的 TIME\_ZONE改成 'Asia/Taipei'

創建一個應用

#cmd python manage.py startapp blog



admin.py : 註冊模型，把他們包含到django的管理介面

migrations : 包含這個應用裡面的數據遷移，允許django跟蹤你的model變化,同步數據庫

models.py : 應用數據模型

views.py : 應用邏輯，每個視圖都會接收一個http請求，處理這個請求，最後返回一個response

添加程式碼到blog下models



title : 帖子的標題

slug : 這個字段在url中使用，是一個短標籤，包含字母數字下畫線及連接線

author : Foreignkey，定義一個一對多的關係

body : TextField，在SQL數據庫是TEXT

created :表明帖子什麼時間創建，auto\_now\_add自動保存當前日期

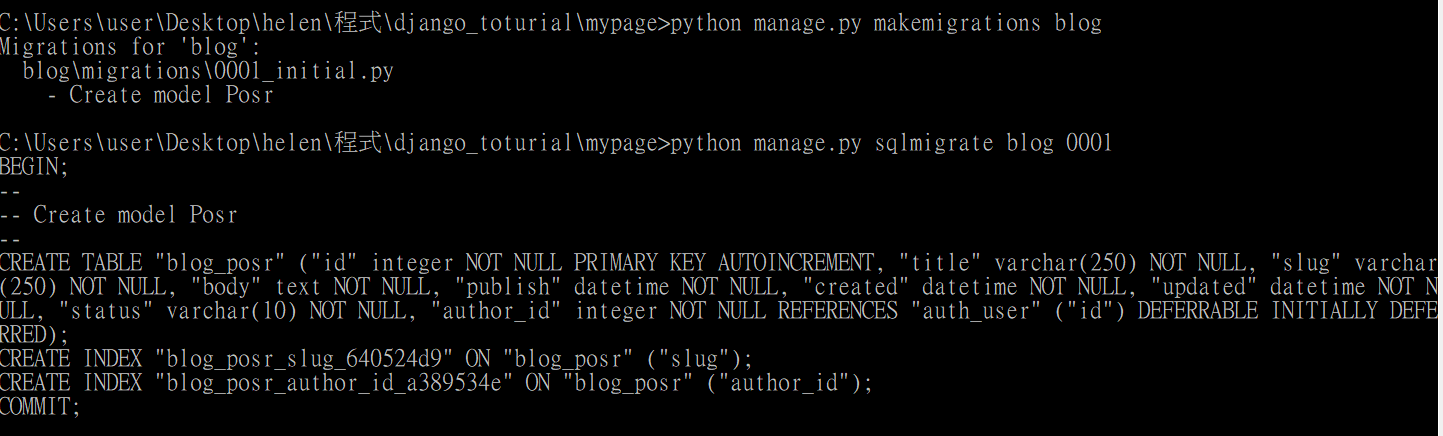
updated:更新時間

Meta:返回依照publish字段降序(負號)排列的結果

#settings下的INSTALLED\_APPS末尾添加 ’blog’,激活這個應用(順序有差，放在最後，不能亂放)

#python manage.py makemigrations blog //數據庫遷移，根據當前模型同步數據庫

#python manage.py sqlmigrate blog 0001 //返回它的sql，根據目前使用的數據庫進行輸出，這裡語句是SQLite的



#python manage.py migrate

***添加/刪除/改變存在的modles字段，或添加新的models而編輯models.py，都要先makemigrations進行數據庫遷移，允許django保持對model改變的跟蹤，再用migrate保持數據庫跟模型同步。***

創建超級用戶來管理管理站點

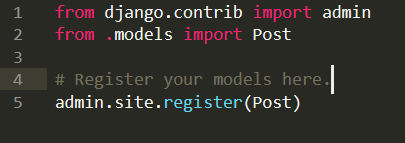
#python manage.py createsuperuser

(輸入密碼時不會有字，鼠標也沒動不用緊張)

#python manage.py runserver

到<http://127.0.0.1:8000/admin/>會看到Group和User(可以看到剛剛創建的用戶訊息)，blog應用中的Post模型跟User模型因為author是連結在一起的。

在管理站點添加模型



這時候會看到管理站點出現BLOG，下面有Post



接下來要訂製models展示形式，使用ModelAdmin的定制類

list\_display : 可以設置在管理對象列表介面想要顯示的模型

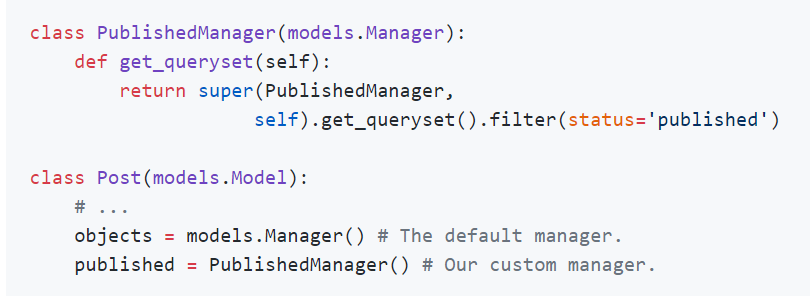
list\_filter : 右側過濾欄

search\_fields : 上面的搜索框

prepopulated\_fields : 告訴django透過title自動填充slug(不要打錯字!!!!!!不會有錯誤訊息可是會無法自動填充)

記得最後一句中把PostAdmin放入

創建model manager



定義我們制訂的管理器，接著添加到模型中。

Django的Object-relational mapping(ORM)是基於查詢集(QuerySet)。查詢集是從你的數據庫中根據一些過濾條件跟範圍取回來的結果對象進行的採集。

構建列和views

一個views接收一個web請求，處理過後返回一個web response。

為每個views定義URL模式，創建html模板templates渲染這些視圖生成的數據。

###帖子列###

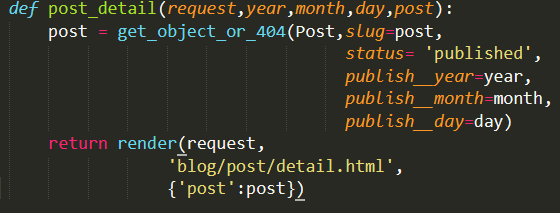


request這個參數所有視圖都需要有!!!

第二句代表我們獲取到所有狀態為已經發布的帖子。

最後，使用render給予的模板渲染帖子列。

###展示單獨帖子###



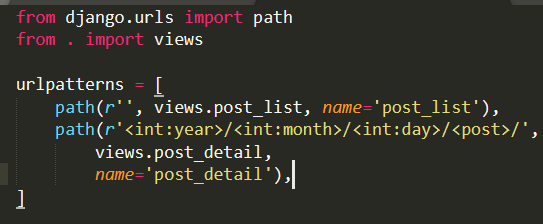
通過給予日期/slug獲取到一篇已經發布的帖子，slug字段中的unique\_for\_date確保在給予的日期只有一個帖子帶有一個slug。

為views添加URL模式

django運行中會遍歷所有URL模式直到第一個匹配的請求URL，之後導入匹配的URL的視圖並且執行。

#blog目錄下創建一個urls.py

最好為每個應用創建單獨的urls.py文件，保證應用可以給別的項目使用



把blog中的url包含到mypage的主URL，編輯mypage的urls.py



需要including another urls時，import include

models的標準URLS

之前使用了post\_detail給Post建構url，django的慣例是給模型添加get\_absolute\_url()方法來返回一個對象的標準url。

使用reverse()允許通過他們名

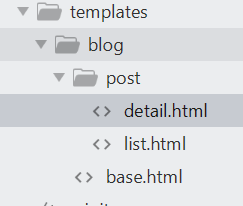


透過strftime保證個位數的月分和日期要戴上0

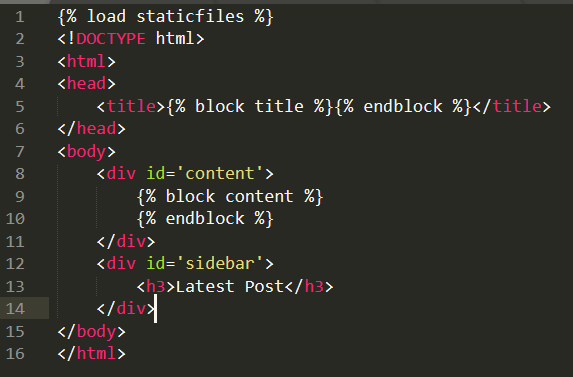
第一句改from django.urls import reverse，第五行裡面去掉blog:

為views創建templates

在blog應用目錄底下建立下列文件結構



base.html



第一行告訴django去加載django.contrib.staticfiles模板標籤(template tags)，加載之後就可以在這個模板中使用{% static %}模板過濾器。

兩個{% block %}標籤，在這個包起來的區塊定義一個block，繼承這個模板的其他模板可以用自定義的內容填充它，這裡有content,title兩個區塊。

list.html

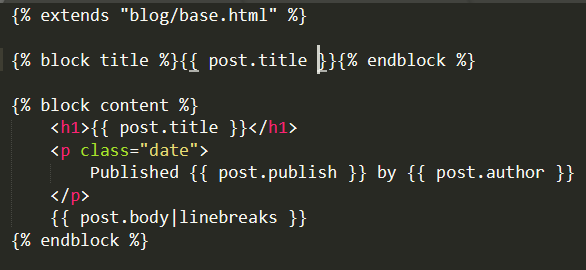


通過{% extends %}模板標籤告訴django需要繼承blog/base.html

通過for循環迭代顯示帖子

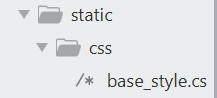
兩個模板過濾器:truncatewords縮短內容限制字數，linebreaks轉換內容的換行符號變成html的換行符號

detail.html



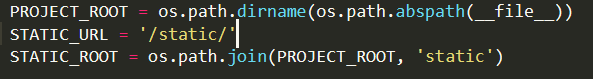
美化一下!

#blog下添加資料夾

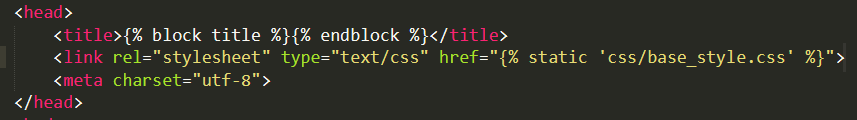


#setting添加上下兩行

(還有別的寫法，可見https://docs.djangoproject.com/en/2.1/howto/static-files/



#base.html放入css檔案

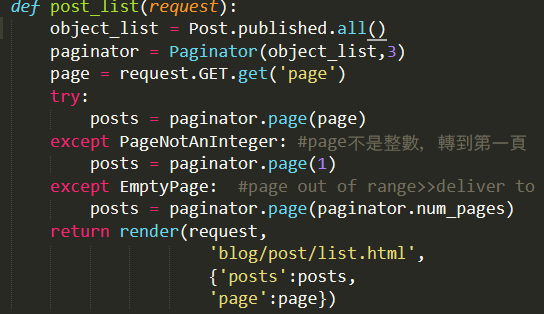


開始更改css檔案的內容!

添加頁碼

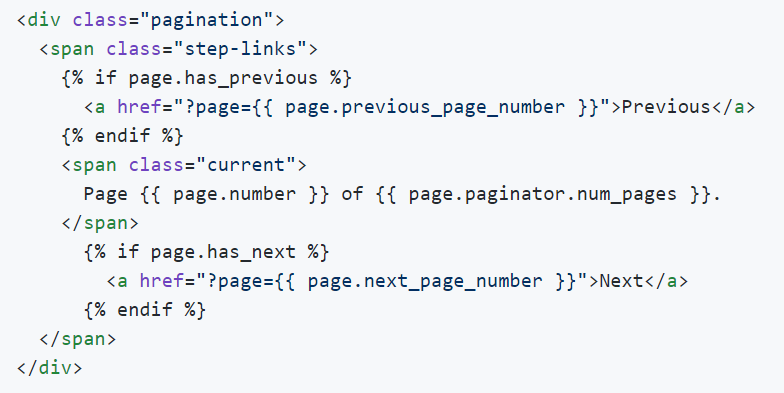
Django有一个内置的Paginator类允许你方便管理分页。

修改views的post\_list



如果page不是一個整數，返回第一頁的結果，如果參數數字超出最大頁數，顯示最後一頁。

創建pagination.html在templates文件夾



#回到list.html模板，把{% include “pagination.html”with page=post %}夾進content裡。

使用基於類的視圖(Classic-based view)

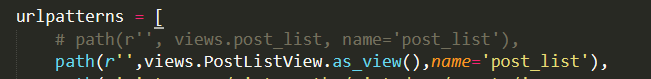


使用特定的查詢集代替取回所有對象。

還就變量posts，不指定的話context\_object\_name默認變量是object\_list。

使用制定的模板來渲染頁面。

#打開blog底下的urls.py文件，註釋掉第一個原本用的，使用新的url模使用PostListView類。



Any arguments passed to [**as\_view()**](https://docs.djangoproject.com/en/2.1/ref/class-based-views/base/#django.views.generic.base.View.as_view) will override attributes set on the class.

#編輯list.html

