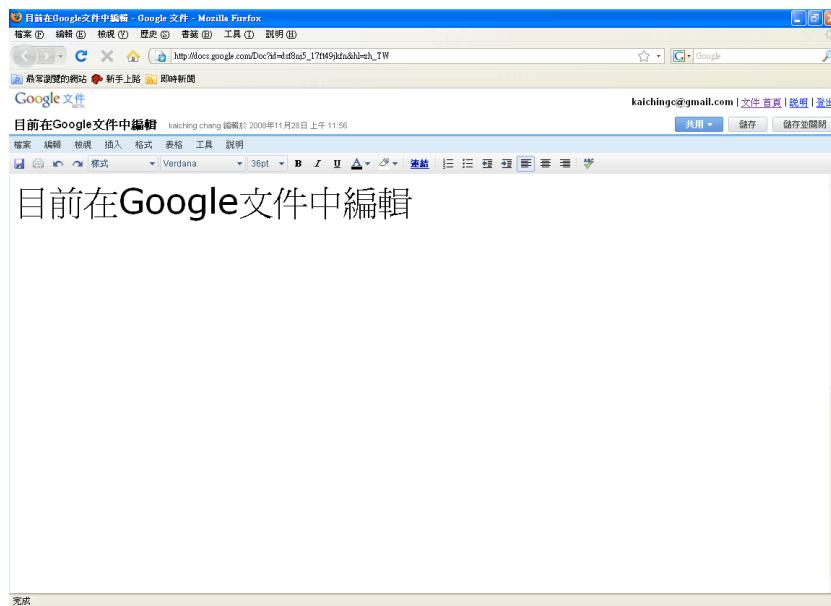


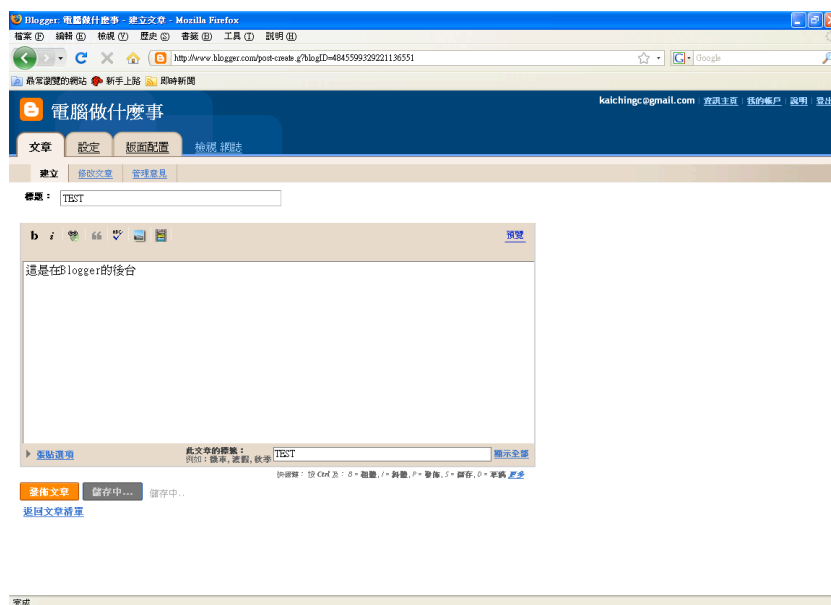
第十七章 網頁的應用程式

所謂網頁的應用程式是指利用瀏覽器執行伺服器上的應用程式，使用者不須事先安裝，典型的例子如Google文件。

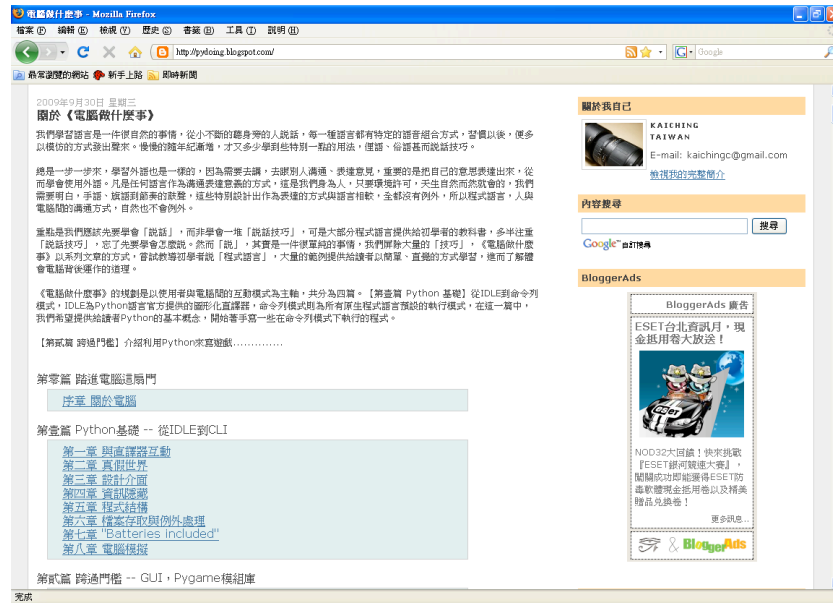


Google文件另外包含簡報、試算表、表格等，同時Google也提供如地圖、閱讀器及電子郵件等多項服務，而另一個搜尋引擎Yahoo!奇摩，或是入口網站yam天空、PChome等也多有提供一些如相簿、購物、算命、部落格等的服務，我們使用這些服務多半透過瀏覽器，因此這些都是網頁應用程式的例子。

某些網路服務，如部落格將介面分成供大眾瀏覽的前台及管理用的後台。



後台提供撰寫文章，版面配置及部落格各種設定等等的應用程式，而前台可以看到的連結項目、動畫，某些特定功能如搜尋等，其實也是屬於網頁應用程式的一種，雖然有些並沒有功能操作。

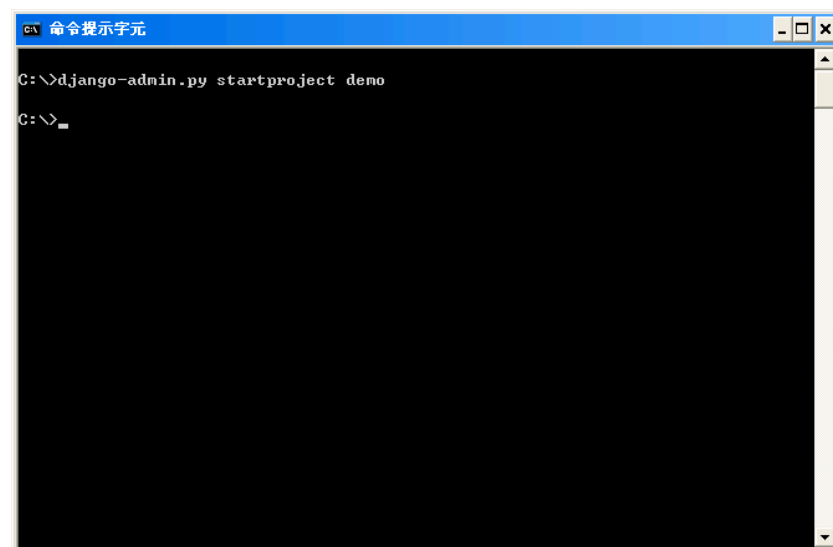


網頁應用程式是一種籠統的概稱，凡是由瀏覽器連結到伺服器，由伺服器執行伺服端的程式，或是從伺服器下載程式到客戶端，也就是在我們自己的電腦上來執行，最後由瀏覽器顯示出結果。

我們繼續利用Django發展動態網頁，下一步，也要開始架構應用程式。

settings.py的調整

我們以常見的留言板做例子，來架構我們自己的網頁應用程式。開始之前，我們重新建立一個名為「demo」的專案。



然後打開setting.py，有一些設定需要先做調整。先看到以下的部份，註解的文字我們略去不提。

```
DATABASE_ENGINE = ''
DATABASE_NAME = ''
DATABASE_USER = ''
DATABASE_PASSWORD = ''
```

```
DATABASE_HOST = ''  
DATABASE_PORT = ''
```

這部份是屬於資料庫的設定，有關資料庫的觀念我們稍後再提，這裡我們只須做兩個調整，分別將變數DATABASE_ENGINE及DATABASE_NAME後的空字串填上設定值。

```
DATABASE_ENGINE = 'sqlite3'  
DATABASE_NAME = 'data.db'
```

DATABASE_ENGINE為所用的資料庫名稱，而DATABASE_NAME為建立資料庫的檔案名稱，我們這裡所選用的資料庫sqlite3，在Python 2.5版以後已經原生支援，所以不需要另外安裝。使用sqlite3的另一個好處是不必額外的開通其他如MySQL等的資料庫系統，發展網頁的過程僅需要我們自己的電腦，也可以不用維持網路連線狀態。

接下來我們看到時區

```
TIME_ZONE = 'America/Chicago'
```

及語系編碼兩個設定。

```
LANGUAGE_CODE = 'en-us'
```

時區預設在美洲的芝加哥，我們將其改成亞洲的臺北

```
TIME_ZONE = 'Asia/Taipei'
```

語系編碼預設為美國英文，這裡要改成臺灣用的中文。

```
LANGUAGE_CODE = 'zh-tw'
```

接著看到最底下

```
INSTALLED_APPS = (  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.sites',  
)
```

我們替序對INSTALLED_APPS加入兩個值，如下。

```
'django.contrib.admin',  
'demo.guestbook',
```

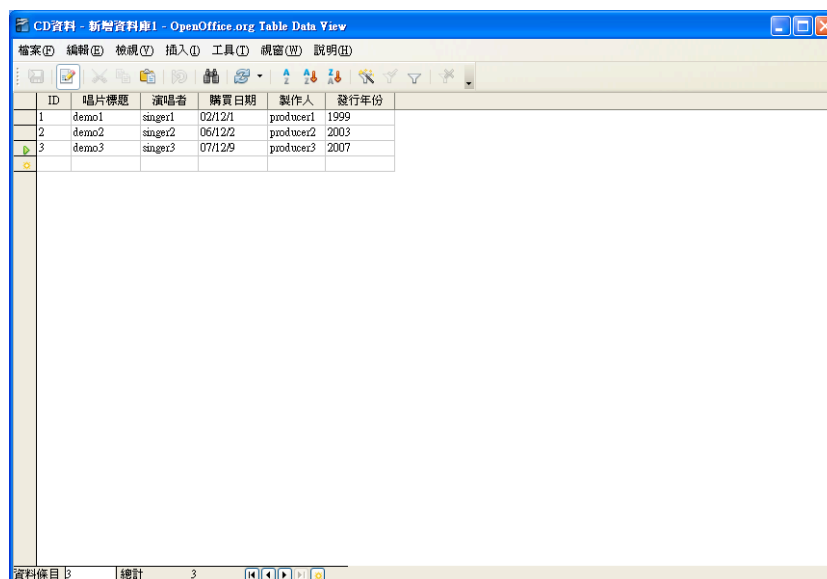
django.contrib.admin是後台管理介面，加入的目的是為了讓我們有一個後台，方便直接管理之用，而demo.guestbook是待會要建立的留言板應用程式。

然後我們來談談資料庫吧！

資料庫的觀念

上一章就提過動態網頁由資料庫來組織檔案，但是我們還沒正式介紹什麼是資料庫。什麼是資料庫呢？簡單來說，資料庫是一群資料有系統的整理方式，方便查詢及修改資料，常見的商業用資料庫系統如Microsoft的Access、甲骨文的Oracle等等。

開放原始碼的資料庫系統的例子則有MySQL、SQLite與OpenOffice的Base等，我們以Base為例，說明資料如何整理。



ID	唱片標題	演唱者	購買日期	製作人	發行年份
1	demo1	singer1	02/12/1	producer1	1999
2	demo2	singer2	06/12/2	producer2	2003
3	demo3	singer3	07/12/9	producer3	2007

如圖，這是利用內建的表格精靈做出來的，每列都是個別的一筆資料，每筆資料都有各自的欄位（行），欄位中填入相對應的資料，Base提供給使用者的是視覺化的方式，讓使用者能夠實際看到資料、儲存，以及後續的組織管理。

這樣的組織方式是不是清楚多了呢？尤其對大量的資料而言，假如學校師生共有三千人，如果要做出通訊錄來，可能就需要姓名、學號、年級、班別、電話、地址等等的資料，表格的列與行就讓我們非常容易理解。這是視覺化所見即所得的方法，然而其他的資料庫系統，如SQLite，卻有極大的不同。

```
÷÷root
ûû
±îÜÄ´im[F.çî¼Sq_L6 ÿíÛÆ±)
delete_comment)
change_comment
#
```

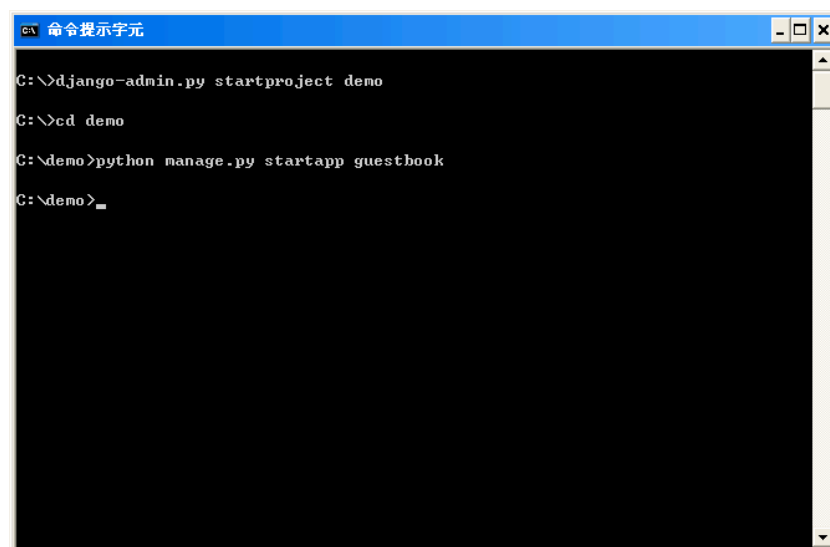
這是用SQLite，也就是上述的sqlite3，所建立的一個資料庫檔案，然後用文字編輯器開啟所看到的一個片段。其中有一些是看得懂的英文字，其他多是將許多符號錯雜，形同亂碼般的排列，這正是SQLite所定義組織資料的方式，他的目的不是給人閱讀，而是方便結構化查詢語言（SQL）的運作。

表格為資料庫運作的其中一種模式。然而由於要加快查詢與修改的速度，所以很多資料庫系統都不是視覺化所見即所得的介面，反倒是利用專屬的模式，如SQL之類的語言來進行操作，很多是特別標記的方式以及簡單的編碼。

實際資料庫背後的道理，甚至操作的方式都相當複雜，我們需要了解到什麼程度呢？別怕，就目前而言，其實知道資料庫是為我們組織資料就夠了，讓Django提供的資料庫介面為我們處理資料。

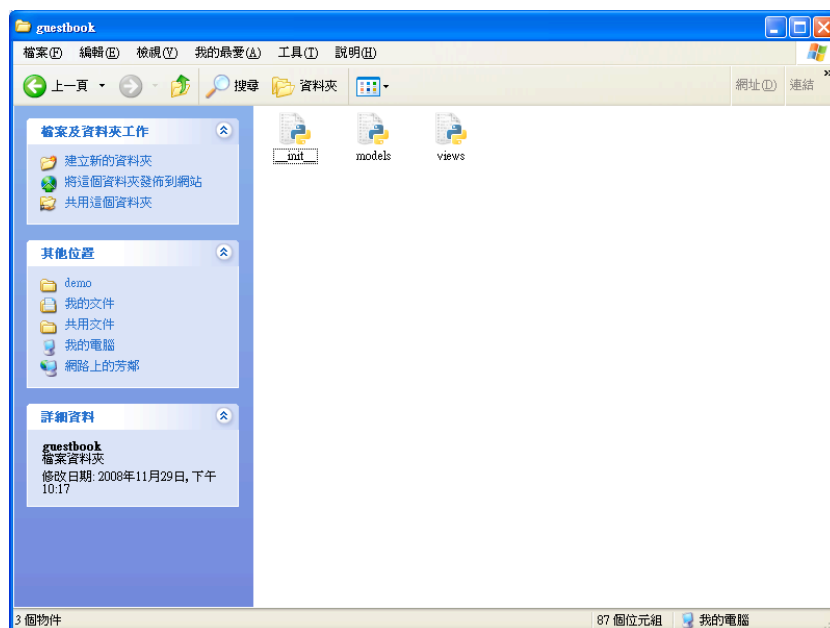
建立app

應用程式簡稱app，我們現在開始來建立網頁的留言板應用程式。首先，移動到demo目錄下，然後打入「python manage.py startapp guestbook」的指令。



```
C:\>django-admin.py startproject demo
C:\>cd demo
C:\demo>python manage.py startapp guestbook
C:\demo>
```

如果無聲無息的跳到下一個提示字元，那就表示我們第一個名為guestbook的app建立成功了，這與建立專案雷同，實際上也是建立一個名為guestbook的資料夾。



其內有三個檔案，說明如下。

- __init__.py：形成套件所需的檔案。
- models.py：放置物件模型的檔案。
- views.py：顯示網頁的檔案。

規劃物件模型

物件模型基本上就是自訂型態，藉由型態定義中的各個屬性賦予資料的各個欄位，程式內容需要放在models.py之中。我們將model.py原有的內容刪去，然後放入下面的程式碼。

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

class Entry(models.Model):
    title = models.CharField(max_length=150) #留言的標題
```

```
name = models.CharField(max_length=50)    #留言者的名字
url = models.URLField(blank=True)         #留言者的網址
text = models.TextField()                 #留言內容

#將物件註冊到資料庫之中
try:
    admin.site.register(Entry)
except admin.site.AlreadyRegistered:
    pass
```

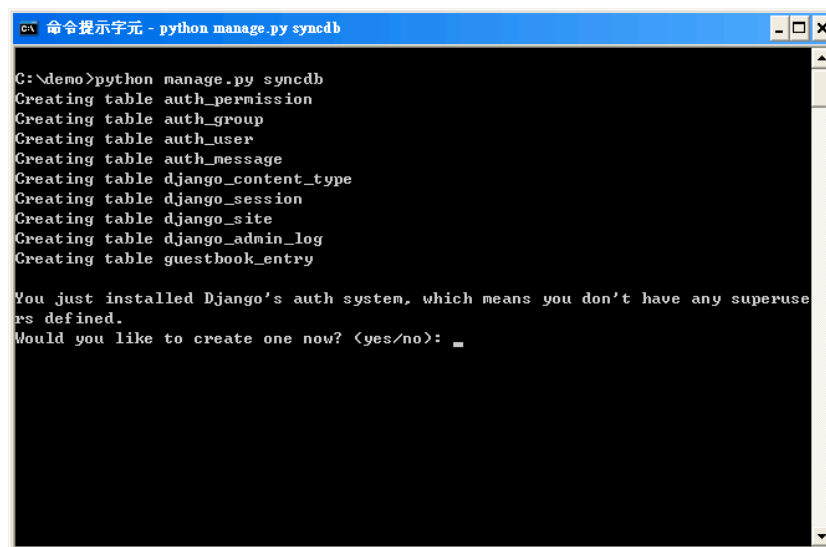
model.py一開始從django.db及django.contrib分別引入models及admin兩個模組，前者用為定義資料欄位，後者則是將物件註冊到資料庫之中。

然後我們規劃留言板當中需要四個輸入欄位，分別是留言的標題、內容，以及留言者的名字、網址，這些欄位以定義Entry型態來達成。Entry型態繼承自Model物件，其內定義title、name、url、text等四個Field物件的屬性。前兩者為CharField物件，單行的文字輸入方塊。url為URLField物件，其效果類似CharField，將參數blank設為True可使這個欄位可以輸入，也可以不用輸入。而text則為TextField物件，預設為多行的文字輸入。

最後利用try陳述，admin.site.register()將型態Entry註冊到資料庫，如果已經註冊的話，便執行except陳述底下的pass陳述，也就是不做任何事情

載入物件模型到資料庫

物件模型規劃完成，接下來我們要設置資料庫的檔案，我們回到「命令提示字元」的視窗，鍵入「python manage.py syncdb」的指令。

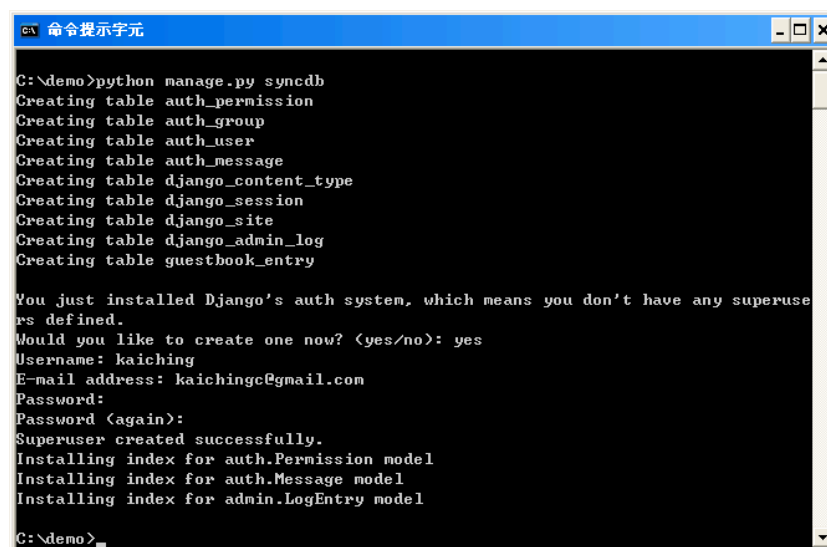


```
命令提示字元 - python manage.py syncdb

C:\demo>python manage.py syncdb
Creating table auth_permission
Creating table auth_group
Creating table auth_user
Creating table auth_message
Creating table django_content_type
Creating table django_session
Creating table django_site
Creating table django_admin_log
Creating table guestbook_entry

You just installed Django's auth system, which means you don't have any superusers defined.
Would you like to create one now? (yes/no): _
```

程式接著建立了許多資料庫所需的資料表，最後問我們要不要建立管理者的帳號，當然要填入yes。

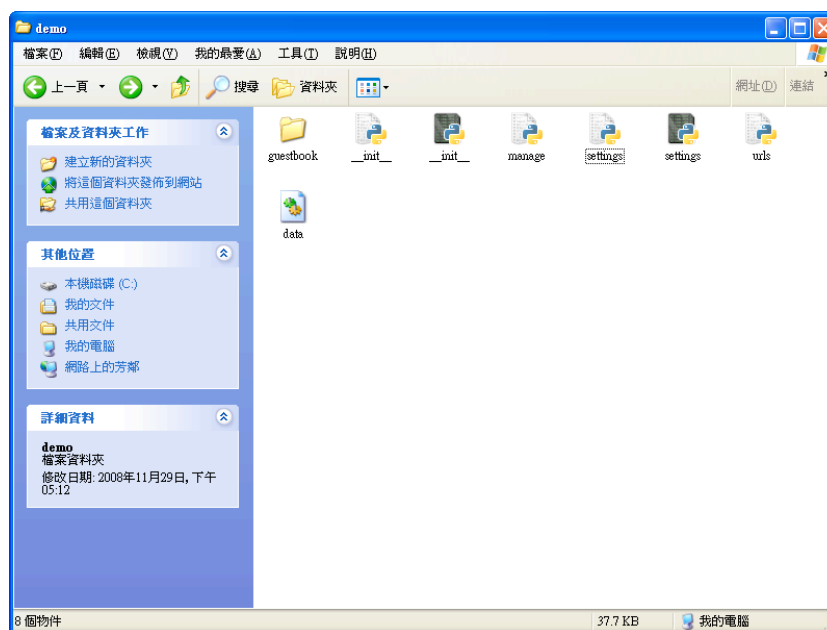


```
C:\demo>python manage.py syncdb
Creating table auth_permission
Creating table auth_group
Creating table auth_user
Creating table auth_message
Creating table django_content_type
Creating table django_session
Creating table django_site
Creating table django_admin_log
Creating table guestbook_entry

You just installed Django's auth system, which means you don't have any superusers defined.
Would you like to create one now? (yes/no): yes
Username: kaiching
E-mail address: kaichingc@gmail.com
Password:
Password (again):
Superuser created successfully.
Installing index for auth.Permission model
Installing index for auth.Message model
Installing index for admin.LogEntry model
C:\demo>
```

輸入使用者名稱、E-mail，然後連續兩次輸入密碼後，管理者的帳號就成功建立了。

我們做完這一連串的步驟，實際上就是建立資料庫檔案data.db。



urls.py的調整

如果要在網頁顯示後台的管理介面，我們還需要調整一下urls.py，只需要去掉幾個跟admin有關的註解井字號即可，最後檔案內容應該要如下所示。

```
from django.conf.urls.defaults import *

# Uncomment the next two lines to enable the admin:
from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    # Example:
    # (r'^demo/', include('demo.foo.urls'))),
```

```
# Uncomment the admin/doc line below and add 'django.contrib.admindocs'
# to INSTALLED_APPS to enable admin documentation:
# (r'^admin/doc/', include('django.contrib.admindocs.urls')),

# Uncomment the next line to enable the admin:
(r'^admin/(.*)', admin.site.root),

)
```

注意中間的

```
# Uncomment the next two lines to enable the admin:
from django.contrib import admin
admin.autodiscover()
```

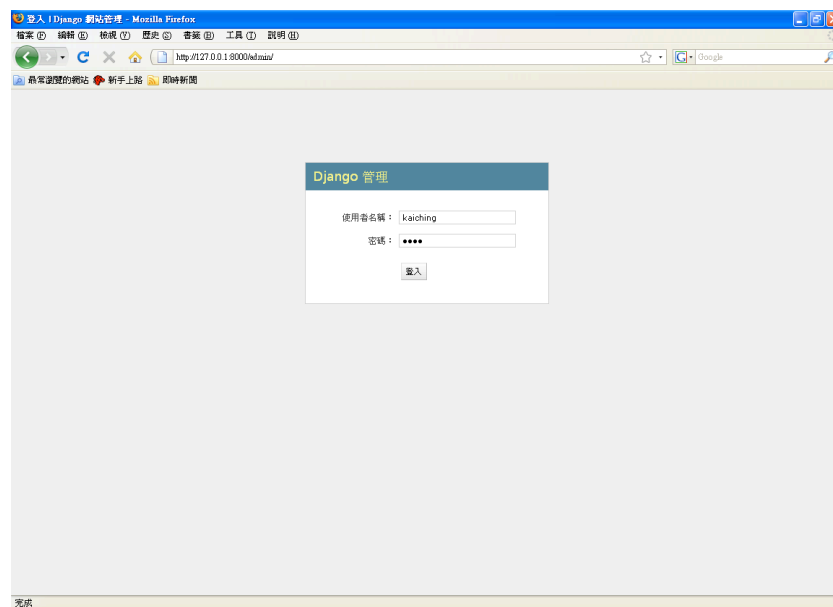
先引入admin，然後呼叫admin.autodiscover()。再來是urlpatterns中最後的

```
# Uncomment the next line to enable the admin:
(r'^admin/(.*)', admin.site.root),
```

這是讓http://127.0.0.1:8000/admin/成為管理介面的網址。

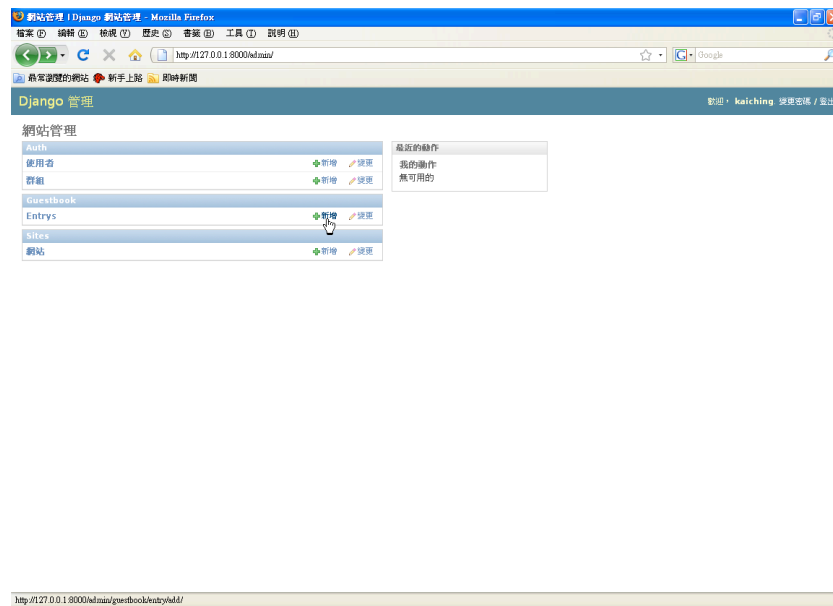
後台管理

我們啟動伺服器，然後開啟瀏覽器輸入http://127.0.0.1:8000/admin/的位置，接著就會出現「Django 管理」的畫面。

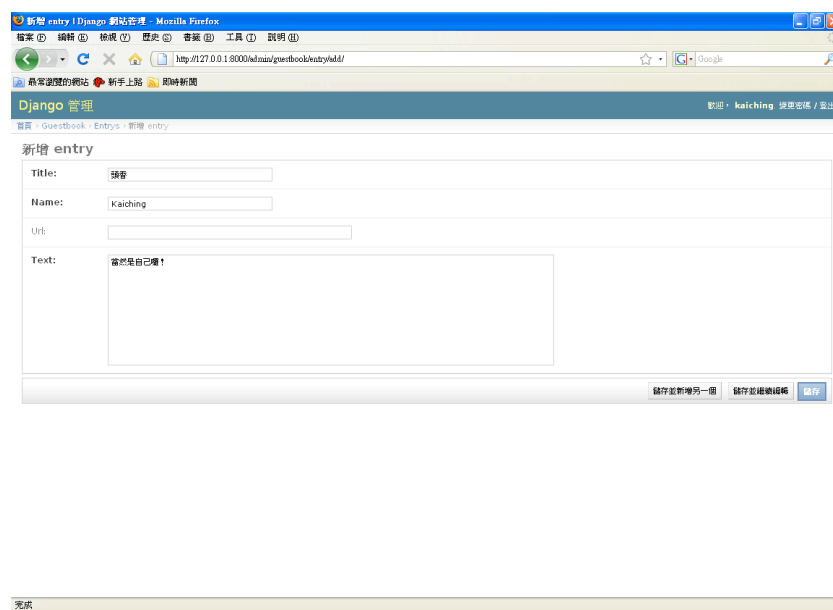


中文的，對吧！如果不是上面的中文介面出現的話，請回頭去檢查settings.py語系設定。

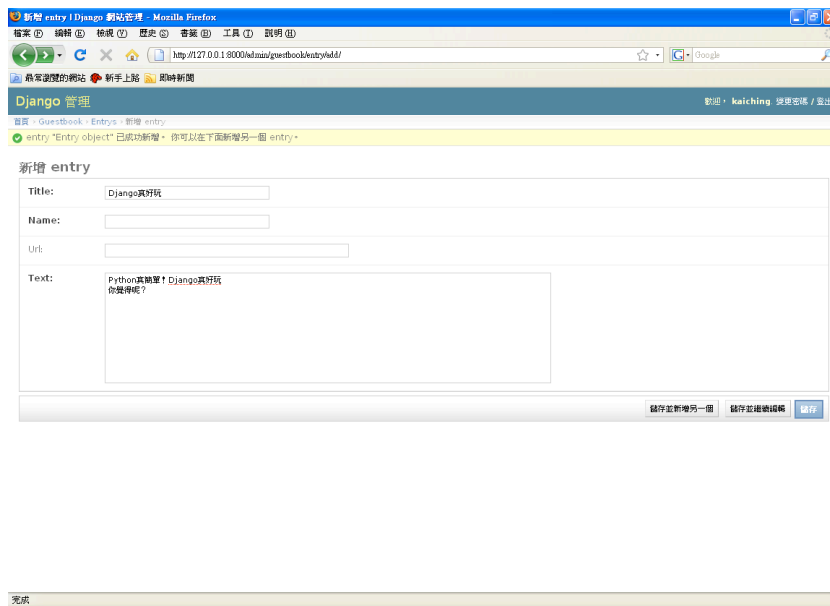
輸入帳號、密碼後登入，我們就進入了管理畫面。其中，暫時不考慮使用者、群組及網站的部份，我們的Guestbook應用程式已在裡頭，包括物件模型Entrys。然後點擊Entrys的【新增】按鈕。



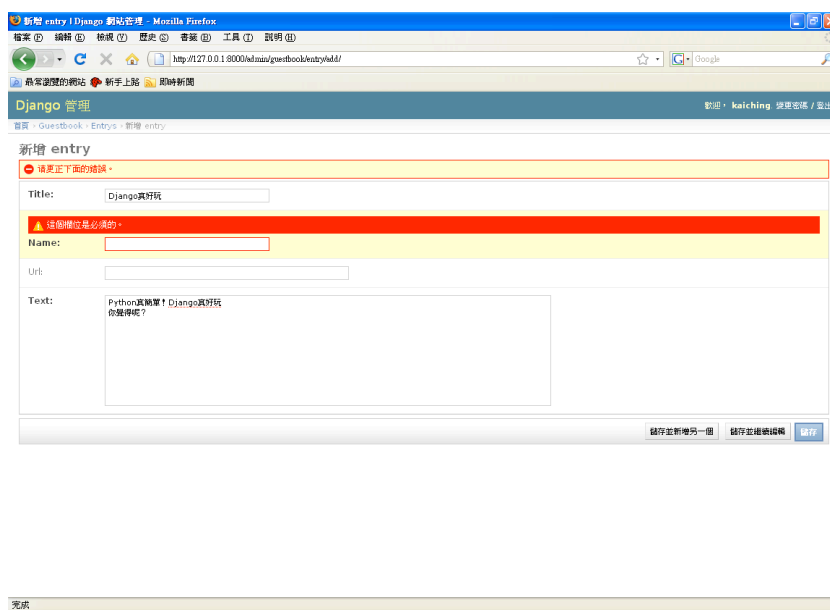
然後出現「新增entry」的畫面，這與之前我們規劃的物件模型吻合，分別有Title、Name、Url及Text四個欄位。我們在Title欄輸入「頭香」，Name欄輸入「Kaiching」，Url欄留空白，Text欄則輸入「當然是自己囉！」，然後按下【儲存並新增另一個】的按鈕。



結果不意外，沒有發生什麼事，就是簡單的出現另一個「新增entry」的畫面，同時提示我們之前的輸入已經成功新增。這次我們再輸入留言，Title欄輸入「Django真好玩」，Name及Url欄留空白，Text欄則輸入「Python真簡單！Django真好玩 你覺得呢？」。

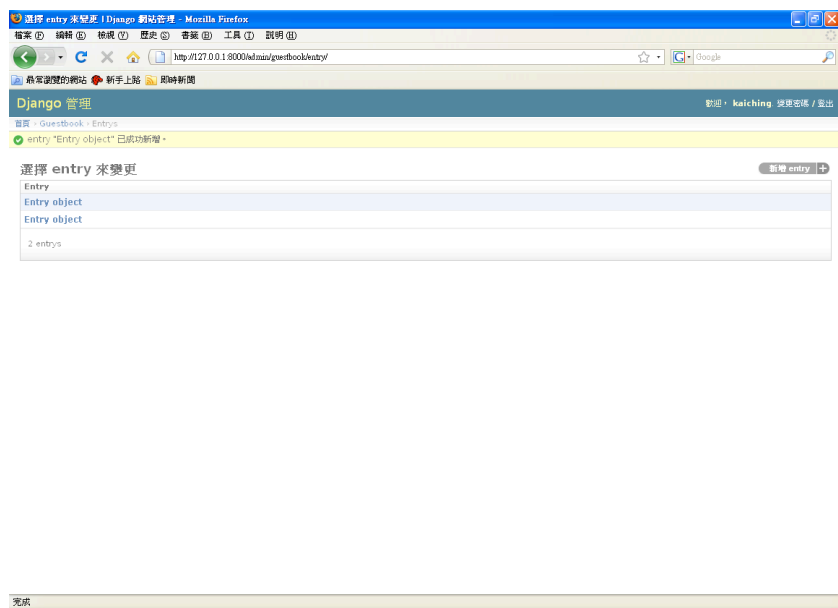


按下【儲存】後，Django管理頁給了我們紅色顯眼的警訊。

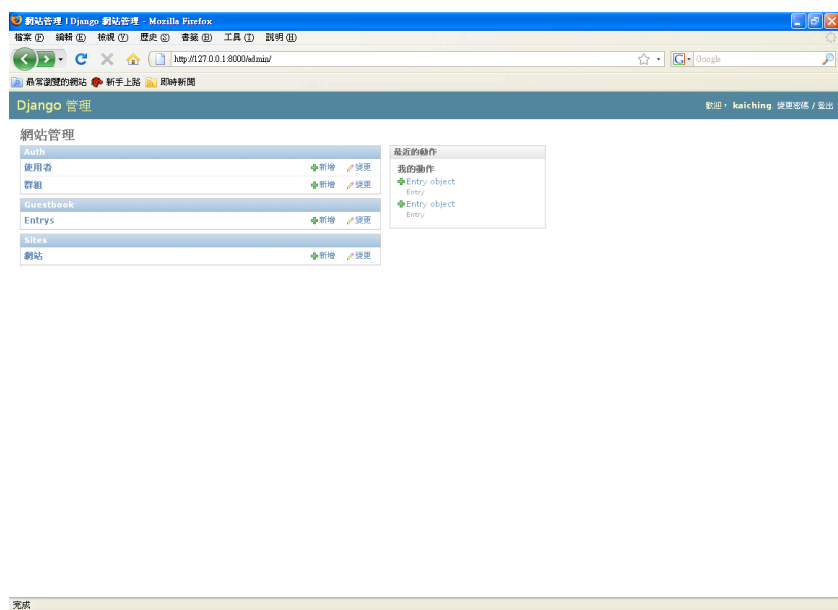


發生了什麼事呢？原因是Name欄位是必須的，Django的管理後台提供了防止錯誤發生的機制，這與Python直譯器提供的例外處理頗為類似，由於程式需要Name欄位才能正常運作，所以發生錯誤。

有沒有發現Url為灰色字樣，這同時也是Django提示給我們該欄位非必須。我們在Name欄中填入名字「Kaiching」，然後按下儲存。



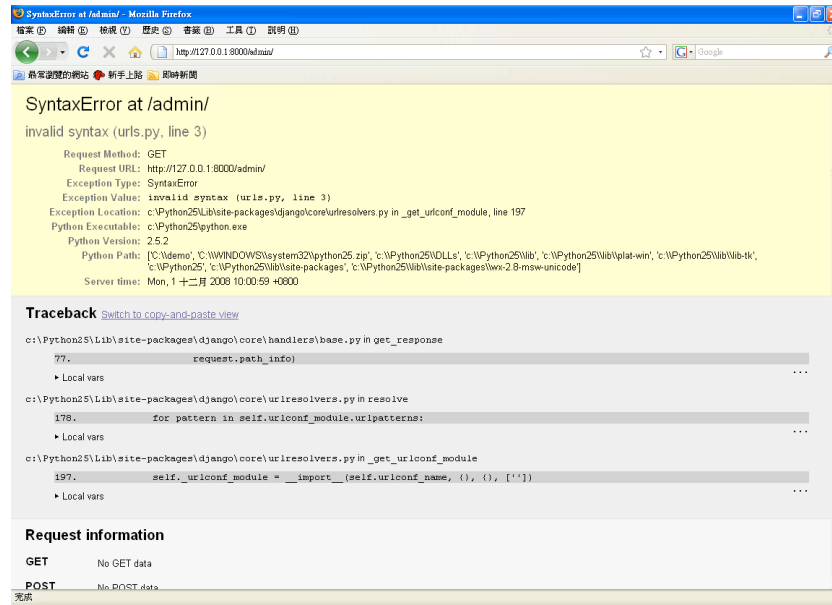
結果告訴我們成功新增，並顯示Entrys的目錄出來。我們連回管理首頁看看。



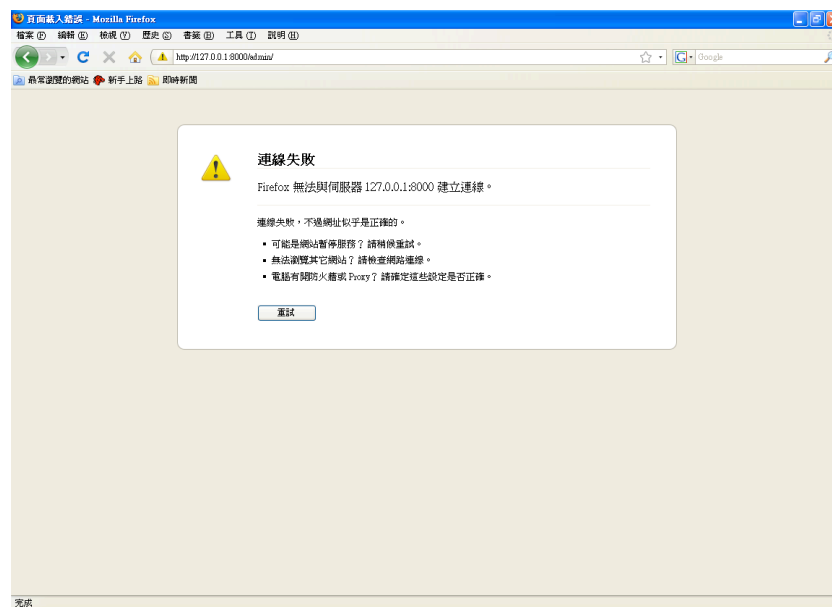
Django的管理後台會有歷史記錄，因此可以看到「最近的動作」的地方出現剛才新增留言的記錄。

錯誤處理

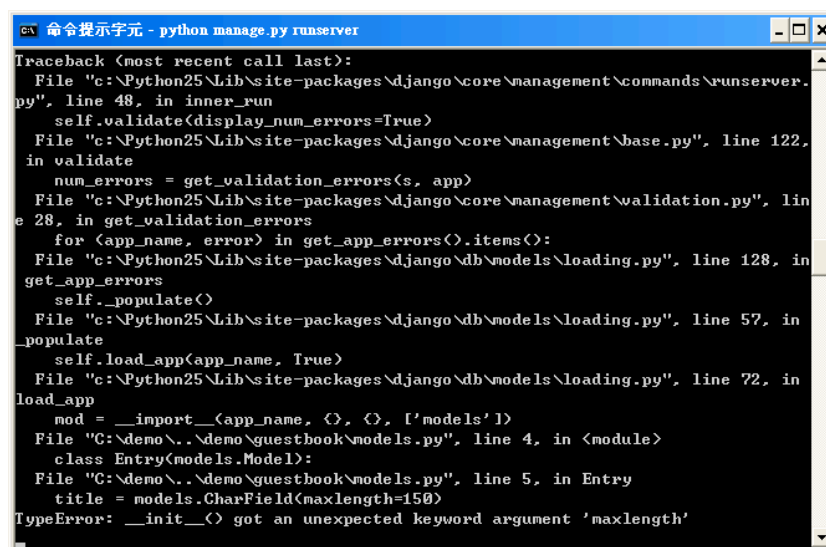
除了Django本身提供的錯誤防止機制外，發展過程中若是程式中有錯誤，譬如句型錯誤，重新連結網頁後，網頁也後自動顯示該錯誤訊息。



但如果是物件模型的錯誤，也就是在models.py之中，譬如將max_length改成maxlength，就會比較麻煩，重新連結網頁反而會連線失敗。



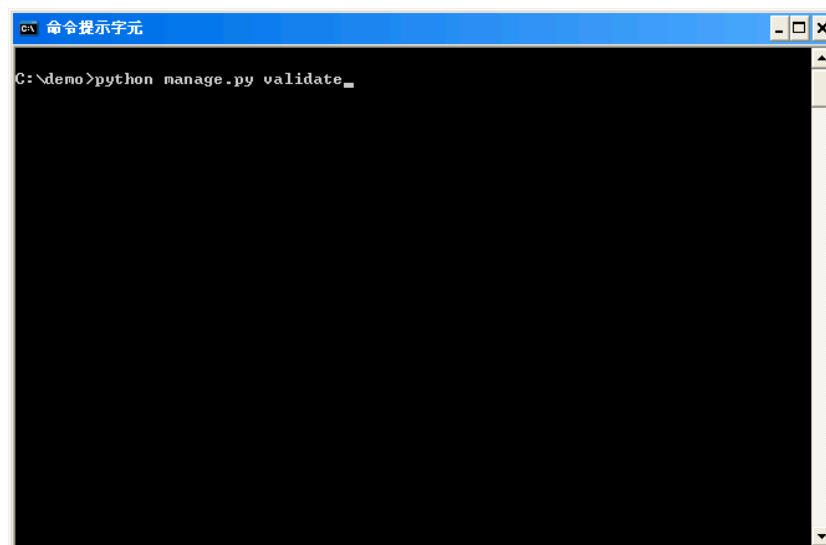
我們把目光轉移到「命令提示字元」視窗，



```
命令提示字元 - python manage.py runserver
Traceback (most recent call last):
  File "c:\Python25\Lib\site-packages\django\core\management\commands\runserver.py", line 48, in inner_run
    self.validate(display_num_errors=True)
  File "c:\Python25\Lib\site-packages\django\core\management\base.py", line 122, in validate
    num_errors = get_validation_errors(s, app)
  File "c:\Python25\Lib\site-packages\django\core\management\validation.py", line 28, in get_validation_errors
    for (app_name, error) in get_app_errors().items():
  File "c:\Python25\Lib\site-packages\django\db\models\loading.py", line 128, in get_app_errors
    self._populate()
  File "c:\Python25\Lib\site-packages\django\db\models\loading.py", line 57, in _populate
    self.load_app(app_name, True)
  File "c:\Python25\Lib\site-packages\django\db\models\loading.py", line 72, in load_app
    mod = __import__(app_name, {}, {}, ['models'])
  File "C:\demo\..\demo\guestbook\models.py", line 4, in <module>
    class Entry(models.Model):
  File "C:\demo\..\demo\guestbook\models.py", line 5, in Entry
    title = models.CharField(maxlength=150)
TypeError: __init__() got an unexpected keyword argument 'maxlength'
```

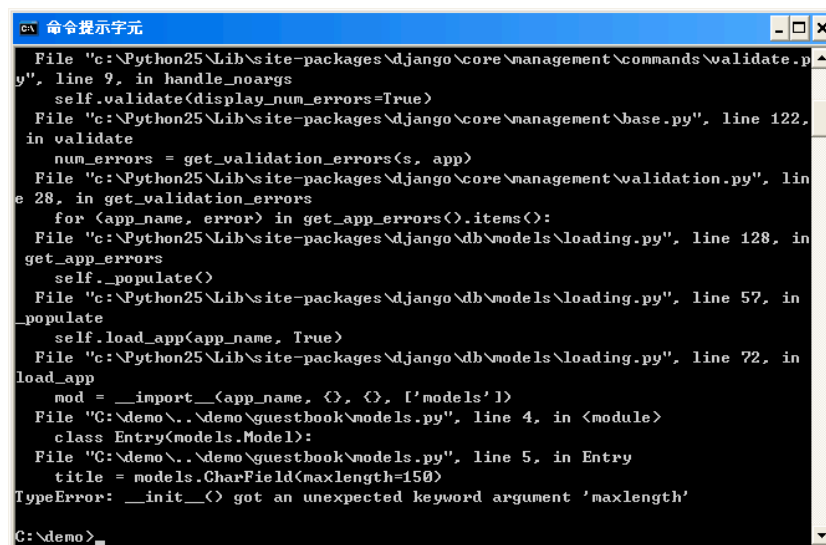
直譯器仍有告訴我們錯誤為TypeError，然而同時造成伺服器的錯誤，伺服器無法持續運作，這時必須停止伺服器，按下〔Ctrl〕+〔c〕，重新啟動伺服器，伺服器才可運作。

重要的是我們在以Django發展網頁時，必須確保物件模型部份的正確，有一個簡單的方法可以交由Django自行檢查，也就是在專案目錄下達「python manage.py validate」的指令。



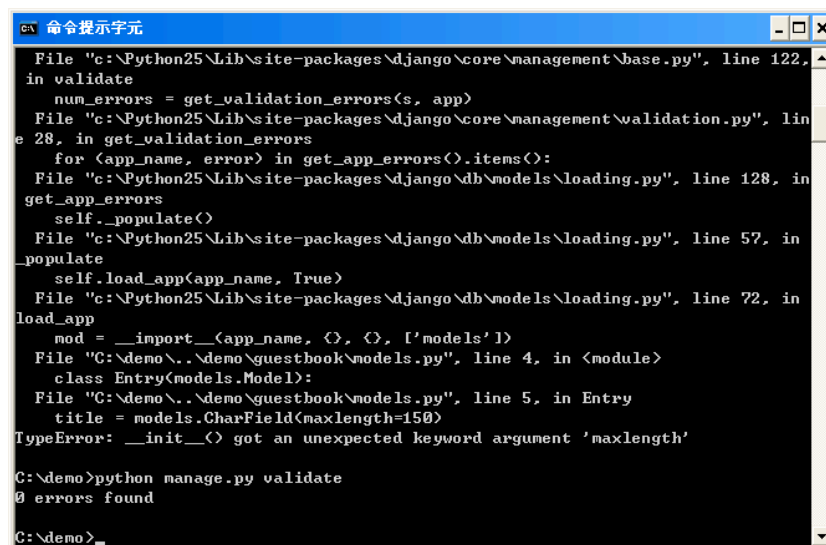
```
命令提示字元
C:\demo>python manage.py validate_
```

在我們還未將maxlength改回成max_length之前，這個指令檢查出相同的錯誤。



```
命令提示字元
File "c:\Python25\Lib\site-packages\django\core\management\commands\validate.py", line 9, in handle_noargs
    self.validate(display_num_errors=True)
File "c:\Python25\Lib\site-packages\django\core\management\base.py", line 122, in validate
    num_errors = get_validation_errors(s, app)
File "c:\Python25\Lib\site-packages\django\core\management\validation.py", line 28, in get_validation_errors
    for (app_name, error) in get_app_errors().items():
File "c:\Python25\Lib\site-packages\django\db\models\loading.py", line 128, in get_app_errors
    self._populate()
File "c:\Python25\Lib\site-packages\django\db\models\loading.py", line 57, in _populate
    self.load_app(app_name, True)
File "c:\Python25\Lib\site-packages\django\db\models\loading.py", line 72, in load_app
    mod = __import__(<app_name>, <>, <>, ['models'])
File "C:\demo\..\demo\guestbook\models.py", line 4, in <module>
    class Entry(models.Model):
File "C:\demo\..\demo\guestbook\models.py", line 5, in Entry
    title = models.CharField(maxlength=150)
TypeError: __init__() got an unexpected keyword argument 'maxlength'
C:\demo>
```

若是我們修復錯誤，下達相同的指令。



```
命令提示字元
File "c:\Python25\Lib\site-packages\django\core\management\base.py", line 122, in validate
    num_errors = get_validation_errors(s, app)
File "c:\Python25\Lib\site-packages\django\core\management\validation.py", line 28, in get_validation_errors
    for (app_name, error) in get_app_errors().items():
File "c:\Python25\Lib\site-packages\django\db\models\loading.py", line 128, in get_app_errors
    self._populate()
File "c:\Python25\Lib\site-packages\django\db\models\loading.py", line 57, in _populate
    self.load_app(app_name, True)
File "c:\Python25\Lib\site-packages\django\db\models\loading.py", line 72, in load_app
    mod = __import__(<app_name>, <>, <>, ['models'])
File "C:\demo\..\demo\guestbook\models.py", line 4, in <module>
    class Entry(models.Model):
File "C:\demo\..\demo\guestbook\models.py", line 5, in Entry
    title = models.CharField(maxlength=150)
TypeError: __init__() got an unexpected keyword argument 'maxlength'

C:\demo>python manage.py validate
0 errors found
C:\demo>
```

「0 errors found」，表示物件模型的部份沒問題了。