

第十六章 動態網頁的開發框架

早期的網頁只是單純的顯示資料，因此網站的建置、經營全部依賴站長的用心程度，接著如蕃薯藤、奇摩等搜尋引擎的興起，其實那時候的搜尋引擎就像電話簿一般，提供目錄供人檢索，找尋所需的資料。

這樣的網頁被稱之為**靜態網頁**，也由於網路技術的蓬勃發展，AJAX、RSS、Flash與部落格等等新穎技術匯集而成Web 2.0，而靜態網頁的相關技術，HTML的制定及發展則被歸類為Web 1.0。

Web 2.0的精髓在於製作**動態網頁**，利用資料庫組織檔案，檔案總數不會每增加一頁就增加一個，而是除了建構網站所需的基本檔案外，另外加入所需的資料庫檔案個數。動態網頁同時與使用者有立即直接的互動，如同線上討論版、網誌或是訂閱RSS新聞等，使用者透過自己的電腦就可以連結到其他的電腦，進行一些公開或非公開的活動。

許多程式語言都能搭配開發框架設計動態網頁，而且通常都有好幾種熱門選擇，Python亦同，我們所要介紹的**Django**正是其中熱門的一種，除了遵守**MVC**的原則外，架構上的完整，整體的一致性，這與Python所訴求的簡單恰巧不謀而合。

什麼是網頁的開發框架呢？所謂的開發框架好比是建構一個商品展示櫥窗，商品是我們所展示的東西，就如同網頁的內容，櫥窗可以設計的美侖美奐，目的是吸引顧客的注意，櫥窗亦如網頁的外貌，若能替網頁規劃一個簡潔、清爽的版面，不僅方便讀者瀏覽內容，也提高讀者駐足或與該網頁互動的興趣。

由櫥窗內側的玻璃門，可由工作人員取出商品，或是倉庫取出新的商品，然後放到櫥窗內，工作人員的工作就像是網頁伺服器所做的事情，這部份正是我們需要寫的程式。倉庫的意義恰如資料庫，程式連結資料庫與網頁，依讀者的要求，由資料庫中擷取資料出來，經過程式的編排，就形成我們眼前的網頁了。

什麼又是MVC呢？MVC是Model-View-Controller的頭字母縮寫詞，顧名思義，Model的中文為模型，這裡的意思是說我們所定義出的資料，通常是用自訂型態的方式，View是呈現網頁外貌的樣板，Controller如同上述商品櫥窗後的工作人員，負責控制與管理。

我們會逐步說明在Django裡的MVC是如何運作的，現在，先來談談Django的設計哲學。

Django的設計哲學

Django的設計哲學包括**寬鬆的結合**、**較少的程式碼**、**快速發展**、**DRY原則**、**清晰性與一致性**。

利用Django製作網頁的各個元件屬於寬鬆的結合，譬如MVC當中的各個部份互不相干，各自獨立運作，這樣的好處是製作網頁樣板時不需考慮資料的設置，連結網址的處理也與其他元件無關等等。寬鬆的結合，發展時只需要專注於個別元件的完備周全，從而利於修改、除錯與後續發展。

發展程式的程式碼越少，寫的速度可以越快，出錯的機會也大為降低。Django是一個基於Python語言的模組庫，因而也承接所有Python語言的特性，諸如「清楚」、「簡潔」、「乾淨」這些Python的禪道，在Django裡處處可見，同時也符合快速發展的設計哲學。由於Django最初的目的就是構築經常迅速更新的新聞網站，因而不論資料的處理、替換等都需要立即有效的作用，因而快速發展為網站是否能成功的關鍵因素。

所謂的DRY是「Don't repeat yourself」的縮寫，中文意思是「不要重複做相同的事」，譬如靜態網頁需要一頁又一頁的撰寫HTML，Django建構的動態網頁只需要一個或數個樣板，然

後所有網頁都能依需求套用。總而言之，冗贅是不好的，訴諸一般化的標準才是Django的本意。

清晰性同Python的禪道，Django不需要傳統程式語言所賦予的魔法，程式的簡單、易讀才是重點。而一致性是說由Django所構築網站的結構，從下而上所有層級保持一致，假設學會了其中某個元件，也能輕鬆掌握其他元件的運作方式。

說了這麼多的設計哲學，是不是有點等不及了呢？我們就來安裝Django吧！

安裝Django

我們以釋出版本1.0，MS-Windows平台的安裝為例。首先從官網下載Django的壓縮檔，經過適當的解壓縮後會得到一個資料夾，接著，我們需要開啟命令提示字元視窗，然後移動目錄路徑到該資料夾中。

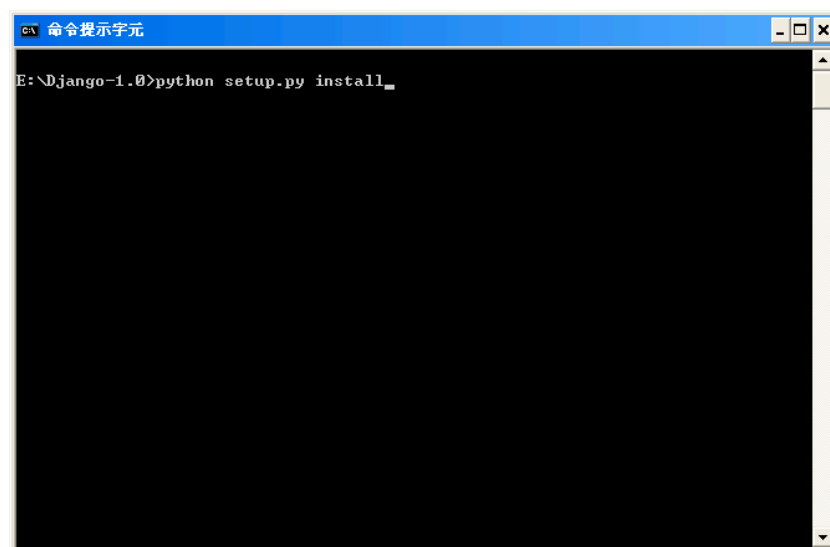
Note

官網的[下載頁](#)有Django其他平台的安裝指南。

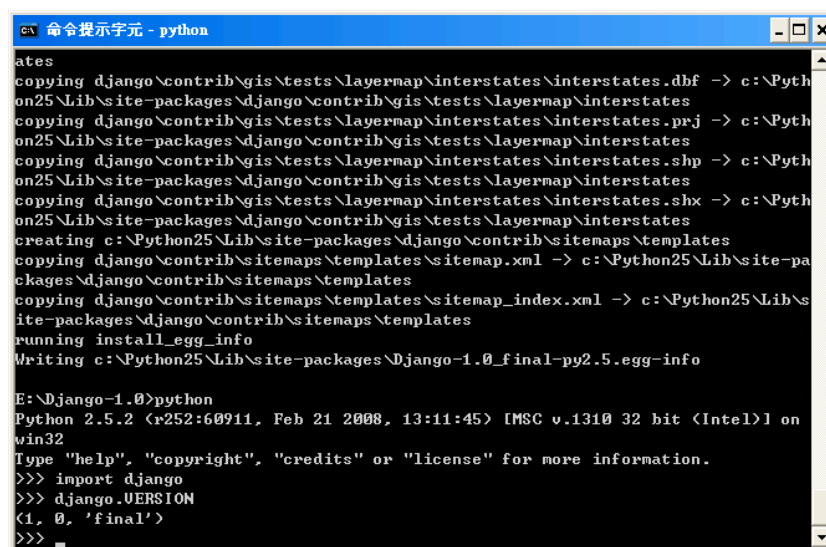
Note

要在Windows命令提示字元中移動所在目錄，請用「cd 目錄名」的指令。

然後，請打入「python setup.py install」的指令。



這樣就會進行安裝了，接著就會出現一長串的訊息，顯示安裝複製檔案的過程。當程序跑完，如果要確認安裝是否成功，我們在下一行提示符號打入「python」指令，進入直譯器之中。我們在第一個「>>>」提示符號後打入「import django」，假如沒有任何事情發生，那就表示安裝是成功的了。



```
ates
copying django\contrib\gis\tests\layermap\interstates\interstates.dbf -> c:\Python25\Lib\site-packages\django\contrib\gis\tests\layermap\interstates
copying django\contrib\gis\tests\layermap\interstates\interstates.prj -> c:\Python25\Lib\site-packages\django\contrib\gis\tests\layermap\interstates
copying django\contrib\gis\tests\layermap\interstates\interstates.shp -> c:\Python25\Lib\site-packages\django\contrib\gis\tests\layermap\interstates
copying django\contrib\gis\tests\layermap\interstates\interstates.shx -> c:\Python25\Lib\site-packages\django\contrib\gis\tests\layermap\interstates
creating c:\Python25\Lib\site-packages\django\contrib\sitemaps\templates
copying django\contrib\sitemaps\templates\sitemap.xml -> c:\Python25\Lib\site-packages\django\contrib\sitemaps\templates
copying django\contrib\sitemaps\templates\sitemap_index.xml -> c:\Python25\Lib\site-packages\django\contrib\sitemaps\templates
running install_egg_info
Writing c:\Python25\Lib\site-packages\Django-1.0_final-py2.5.egg-info

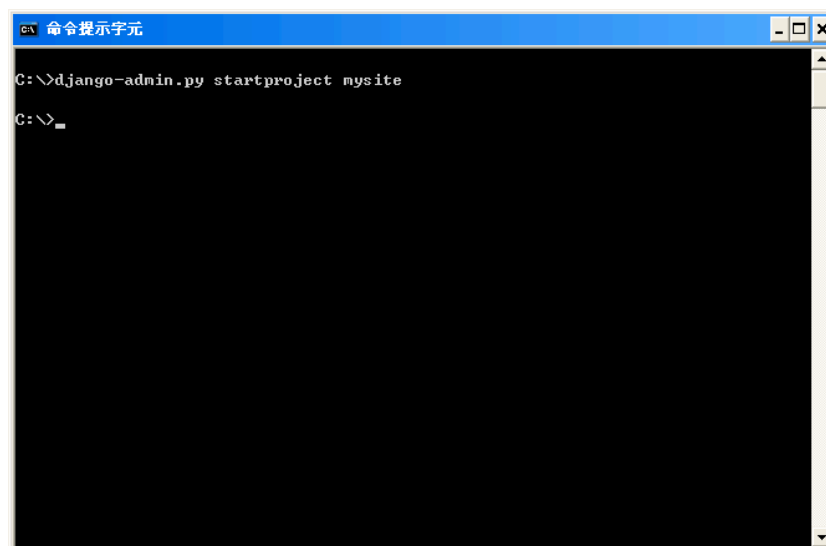
E:\Django-1.0>python
Python 2.5.2 (r252:60911, Feb 21 2008, 13:11:45) [MSC v.1310 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> django.VERSION
(1, 0, 'final')
>>>
```

雖然實際的網站架設都需要伺服器，硬體可以不用專屬的電腦，我們自己的電腦也能當做伺服器，然而軟體方面比較麻煩，可能需要些專屬的軟體，我們卻可以暫時不須煩惱這些，因為Django有一個簡易型的伺服器，純粹提供發展網站過程之用。

伺服器部份稍後再提，我們需要先建立專案。

建立專案

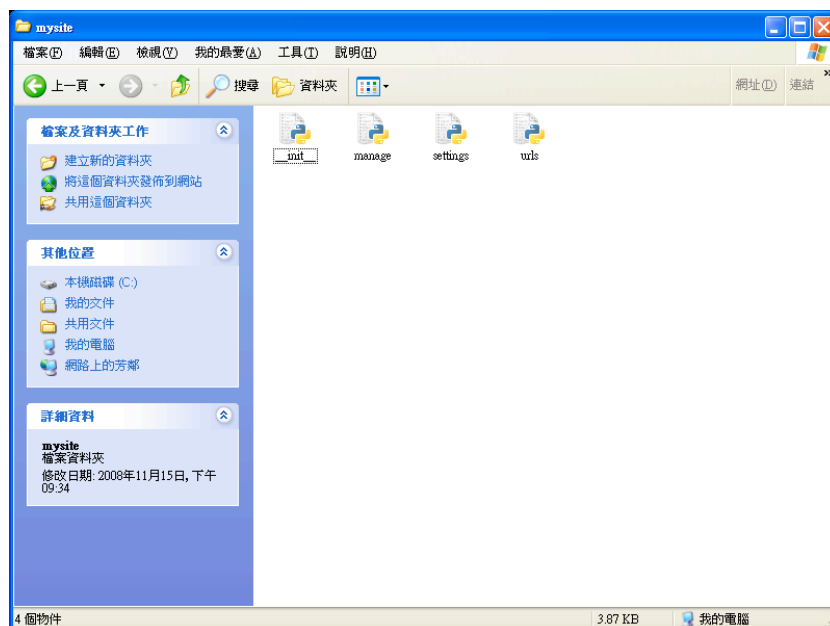
專案是網站所有檔案的集合，我們要在打算放置專案的資料夾中鍵入「django-admin.py startproject mysite」的指令。然後按下Enter，如果沒有其他訊息就跳到下一個提示符號，就表示成功建立一個名為mysite的專案了。



```
C:\>django-admin.py startproject mysite
C:\>
```

mysite是自訂的專案名稱，當然，我們可以自由選擇專案名稱，但是建議不要跟Django或Python的內建名稱衝突，譬如取名為「django」，這樣一來專案內的檔案若引入Django模組庫或是其他標準模組庫中的模組時，會發生模組無法作用的情況。

mysite專案實際上的動作是建立一個mysite的資料夾。



注意一點，這也是利用套件的方式組織的，以便在每個模組檔案中相互引入利用。

專案內有四個檔案，說明如下。

__init__.py：形成套件所需的檔案。

manage.py：命令列工具，用來管理專案，詳細說明可參考[django-admin.py and manage.py](#)。

settings.py：提供設定專案的檔案，詳細說明可參考[Django settings](#)。

urls.py：提供網頁連結方式的檔案，詳細說明可參考[URL dispatcher](#)。

後續會依需要分別說明這些檔案內容，我們接著來啟動Django提供的伺服器。

啟動伺服器

我們在命令提示字元中移動到mysite資料夾，然後下達「python manage.py runserver」指令。

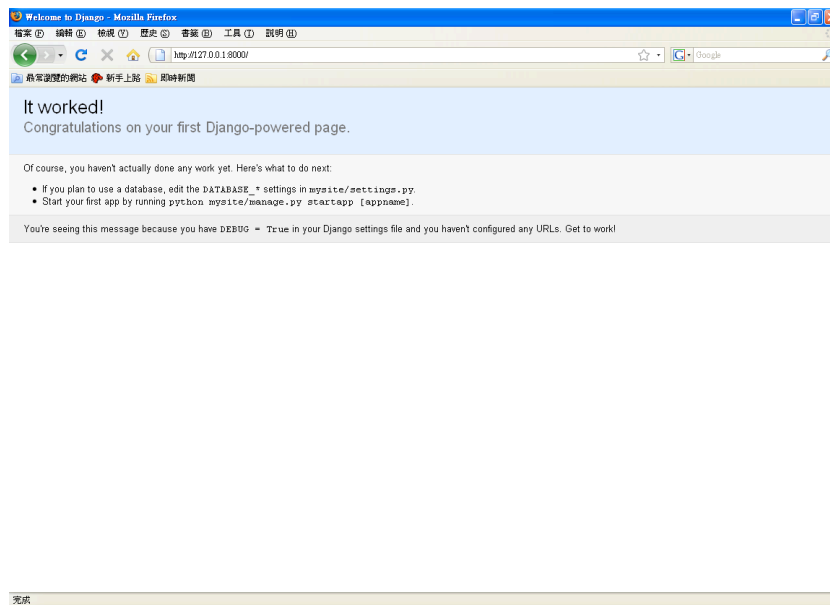
```
命令提示字元 - python manage.py runserver

C:\mysite>python manage.py runserver
Validating models...
0 errors found

Django version 1.0-final-SUN-unknown, using settings 'mysite.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

程式先驗證有無錯誤，確認沒有錯誤後變開始啟動伺服器，同時告訴我們使用的是mysite中setting.py中的設定值，利用網頁瀏覽器在http://127.0.0.1:8000/的位置可以看到網頁結果。若要離開伺服器，同時按住Ctrl及Break兩鍵即可（或是Ctrl及c）。

我們打開Firefox，然後輸入http://127.0.0.1:8000/的網址。



「It worked!」這個訊息表示伺服器架設成功了。

NOW!

接下來，我們建立一個顯示現在時間的簡單例子，先要做的是在mysite資料夾中建立一個view.py檔案，並且存入以下的內容。

```
#-*- coding: UTF-8 -*-
```

```
from django.http import HttpResponse
import datetime
```

```
def current_datetime(request):
    now = datetime.datetime.now()
    html = u"<html><body>現在時間是 %s.</body></html>" % now
    return HttpResponse(html)
```

檔案開始是UTF-8的編碼宣告，接著從django模組庫的http模組引入HttpResponse型態，再引入標準模組庫中的datetime模組。裡頭僅僅定義了一個current_time()函數，所需一個參數為HttpResponse型態的物件，習慣上命名為request。

程式碼共三行，第一行由datetime模組取得電腦的現在時間，儲存到變數now之中，然後建立一個u前綴字串，包含HTML的基本語法，最後當成HttpResponse型態的參數，同時current_time()直接回傳該型態所建立的物件。

view.py把MVC原則中的V與M融合在一起，這裡是一個簡單例子，所以實際的V（樣板）放在變數html之中，利用變數now記錄的現在時間則屬於M，我們在下一章會把V與M分開存放，V放在views.py，而M放在models.py。至於C沒有專屬的檔案，各種控制邏輯都屬於C的範疇。

例子很單純，我們利用HttpResponse型態建立所要顯示的網頁，但若要讓網頁實際可見，我們還須修改url.py。

Note

關於HttpResponse型態的詳細說明，可參考官網中的[HttpResponse objects](http://pydo.org/HttpResponse-objects/)。

URLconf

URLconf就是設定網頁連結的方式，也就是連結網址的設定，mysite專案之中利用url.py儲存其設定。我們開啟url.py，發現檔案中已經有如下的內容。

```
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'^mysite/', include('mysite.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),
)
```

這個檔案是執行django-admin.py後產生的，有許多行已經存在，並且加上註解的井字號，某些確實是程式的一部分，透過註解化使其成為註解，而不會被執行。註解化的程式行，如

```
# from django.contrib import admin
# admin.autodiscover()
```

與

```
# (r'^mysite/', include('mysite.foo.urls')),
# (r'^admin/doc/', include('django.contrib.admindocs.urls')),
# (r'^admin/(.*)', admin.site.root),
```

都是屬於程式預設的一部分，若是我們去除註解的井字號，伺服器有在執行的情況底下，連結到首頁（http://127.0.0.1:8000/），便可瀏覽我們所建置的網頁內容。

其中，後者便是有關連結網址的設定部份，「r'^mysite/'」、「r'^admin/doc/'」、「r'^admin/(.*)'」是用**r前綴字串**，而字串內屬於**正規表示法**，「include('mysite.foo.urls')」、「include('django.contrib.admindocs.urls')」、「admin.site.root」則是連結到該位置，伺服器相對應執行的程式。

r前綴字串把引號裡的內容以原始方式儲存，因為有些正規表示法可能會如同跳脫字元一般，所以r前綴字串避免了這些問題。我們暫時不去討論這些註解化的程式碼，先將url.py修改為所需要的內容。

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

urlpatterns = patterns('',
    (r'^time/$', 'mysite.view.current_datetime'),
)
```

首先從django.conf.urls.defaults模組引入所有的物件，包括patterns()函數，然後呼叫patterns()函數，將其回傳值儲存到變數urlpatterns之中。函數patterns()的第一個參數為空字串，其後為數個設定連結的序對值。序對中需要兩個數值，其內第一個是r前綴字串，第二個也是字串，利用Python路徑表示網頁內容是用mysite資料夾中view模組的current_datetime()函數。

正規表示法表示出的字串當中，第一個「^」是脫字符號，接著是路徑名「time」加上斜線「/」，最後是金錢符號「\$」。脫字符號在正規表示法中代表字串的開始，金錢符號則代表字串的結束。

因此，網址<http://127.0.0.1:8000/time/>就會是例子結果顯示的網頁。

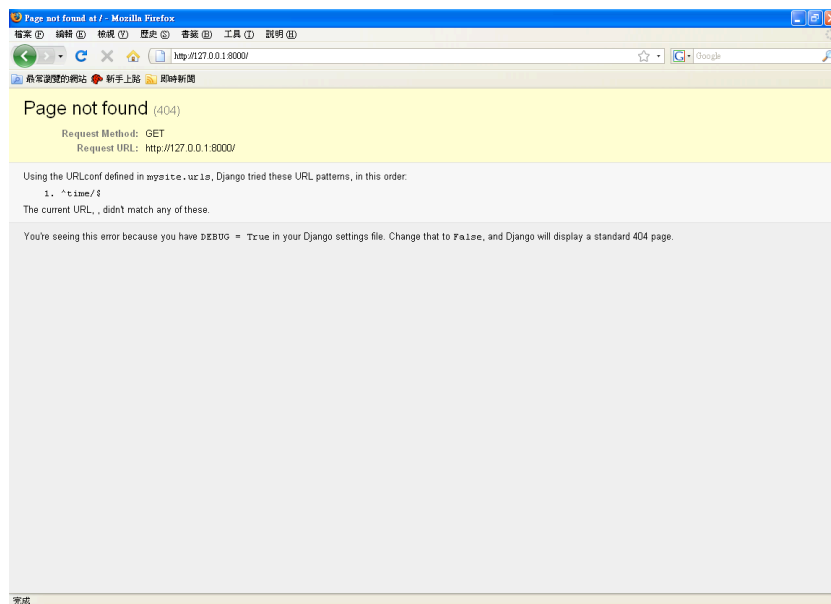
Note

關於正規表示法的詳細說明，可參考Python Library Reference中的[re](#)。

錯誤代碼404

若修改檔案，網頁重新連結就可以看到修改後的效果，但是在mysite加入新的檔案，就必須要重新啟動伺服器。因為我們加入了新的url.py，所以先以Ctrl+c結束稍早的伺服器，再以「python manage.py runserver」的指令重新啟動伺服器。

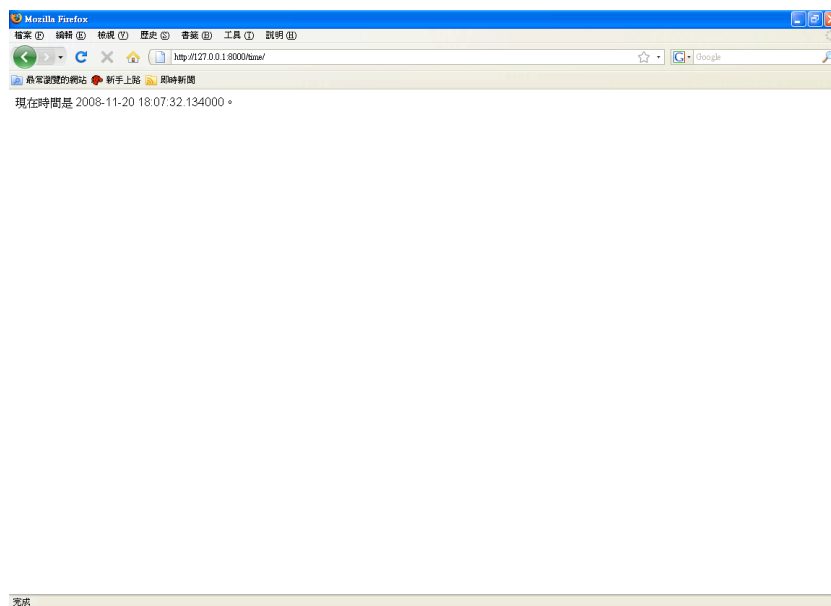
然後將網頁瀏覽器重新連結到網址<http://127.0.0.1:8000/>，結果出現以下的畫面。



錯誤訊息「Page not found (404)」，中文意思是說找不到這一頁，這是由於<http://127.0.0.1:8000/>的網址我們並沒有設定顯示的網頁，Django的伺服器產生這個訊息告訴我們。其下依URLconf設定中提供連結的建議，目前網址並不屬於URLconf之中。

凡是程式有發生例外的情況，如句型錯誤、名稱錯誤等，Django伺服器便以錯誤訊息提醒我們，就如同在IDLE中的方式一般，使我們修改錯誤可以更方便也迅速。

我們將連結網址更改為<http://127.0.0.1:8000/time/>，就會正確的顯示現在時間了。



動態的URL

前面的例子我們利用連線到特定的網址，使伺服器顯示出現在時間，伺服器也是我們自己的電腦，因為時間不會一樣，所以每一次的連線的結果都不同，網頁顯示動態的內容。事實上，連結網址也可以是動態的。

譬如，我們將url.py修改如下。

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

urlpatterns = patterns('',
    (r'^$', 'mysite.view.homepage'),
    (r'^time/$', 'mysite.view.current_datetime'),
    (r'^time/(1)/$', 'mysite.view.page_counter'),
    (r'^time/(2)/$', 'mysite.view.page_counter'),
    (r'^time/(3)/$', 'mysite.view.page_counter'),
    (r'^time/(4)/$', 'mysite.view.page_counter'),
    (r'^time/(5)/$', 'mysite.view.page_counter'),
)
```

其中，脫字符號「^」直接接上金錢符號「\$」，代表伺服器的網址http://127.0.0.1:8000/，而「^time/(1)/\$」到「^time/(5)/\$」則表示http://127.0.0.1:8000/time/1/到http://127.0.0.1:8000/time/5/的網址，小括弧的目的在於給下面定義的函數抓取數值。

同時我們在view.py中需要另外兩個函數，homepage()、page_counter()作為顯示網頁的函數，如下。

```
def homepage(request):
    html = u"<html><body>您來到了首頁.....</body></html>"
    return HttpResponse(html)

def page_counter(request, offset):
    html = u"<html><body>您來到了第 %s 頁。</body></html>" % offset
    return HttpResponse(html)
```

函數homepage()簡單的在網頁中顯示「您來到了首頁.....」的文字，page_counter()則會顯示來到的頁數，這個函數也需要多用一個參數，offset會抓取urlpatterns裡網址設定中用小括號圍起來的數值，並用字串儲存，如「^time/(1)/\$」，offset就會得到“1”。

不過這樣稍嫌麻煩，網頁多弄一頁出來，在urlpatterns就需要多一行設定連結的程式碼，其實正規表示法有簡化的方式，我們將需要page_counter()函數的程式修改如下。

```
(r'^time/(\d{1,2})/$', 'mysite.view.page_counter'),
```

當中「\d{1,2}」代表凡是個位數與兩位數的數字都是可被offset抓取的範圍，如此一來，一共可以建立112張網頁，第99頁為其最後一頁，我們將view.py的內容稍做更改，如下。

```
#-*- coding: UTF-8 -*-
```

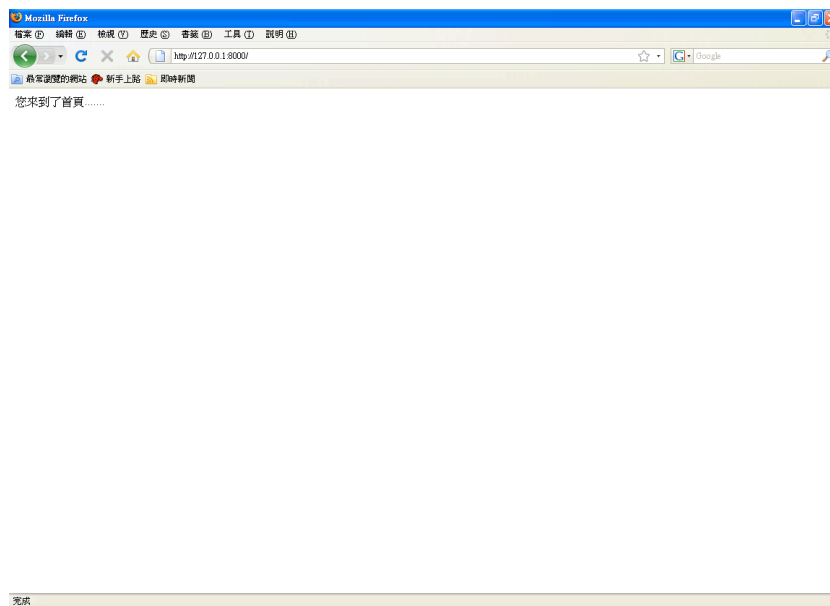
```
from django.http import HttpResponse
import datetime

def current_datetime(request):
    now = datetime.datetime.now()
    html = u"<html><body>現在時間是 %s。</body></html>" % now
    return HttpResponse(html)

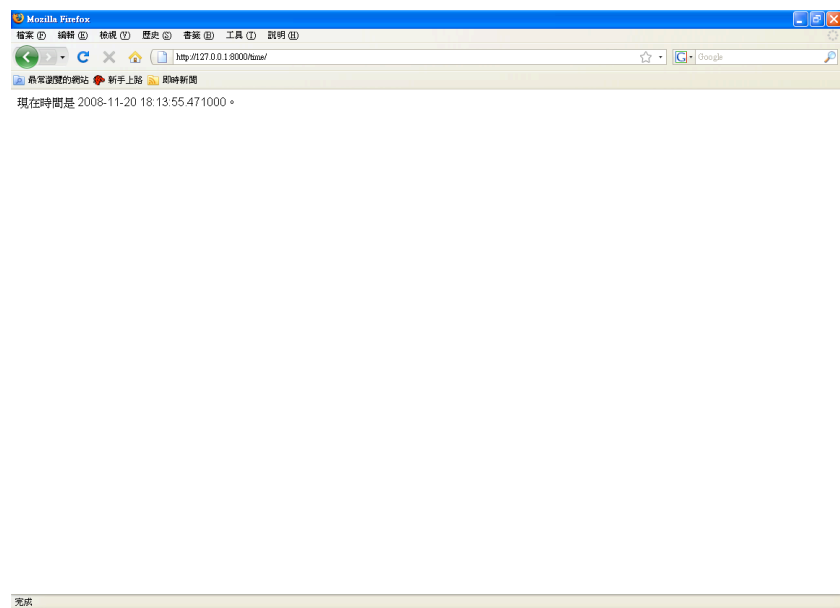
def homepage(request):
    html = u"<html><body>您來到了首頁.....</body></html>"
    return HttpResponse(html)

def page_counter(request, offset):
    now = datetime.datetime.now()
    if offset == "99":
        html = u"<html><body>這是最後一頁囉！<BR /> \
            現在時間是 %s。</body></html>" % now
    else:
        html = u"<html><body>您來到了第 %s 頁；<BR /> \
            現在時間是 %s。</body></html>" % (offset, now)
    return HttpResponse(html)
```

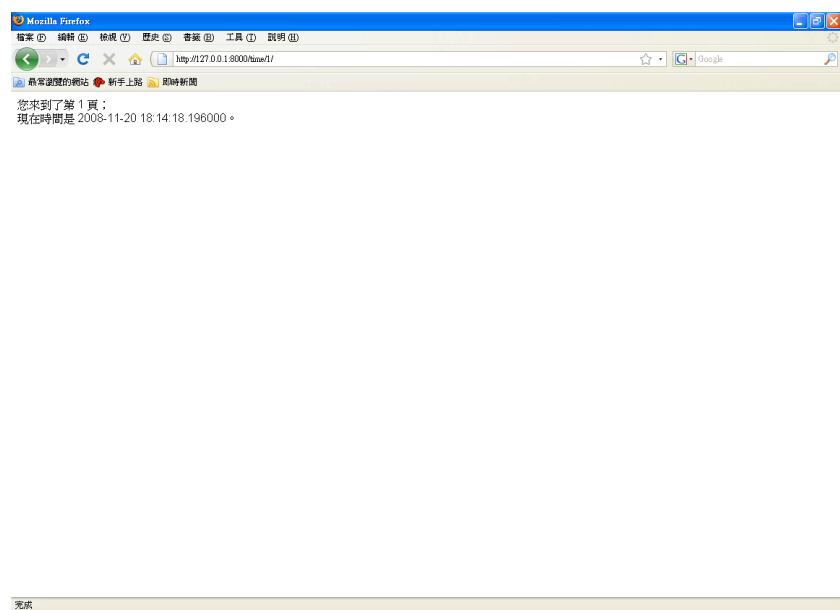
重新連結網址到http://127.0.0.1:8000/，會有如下的結果。



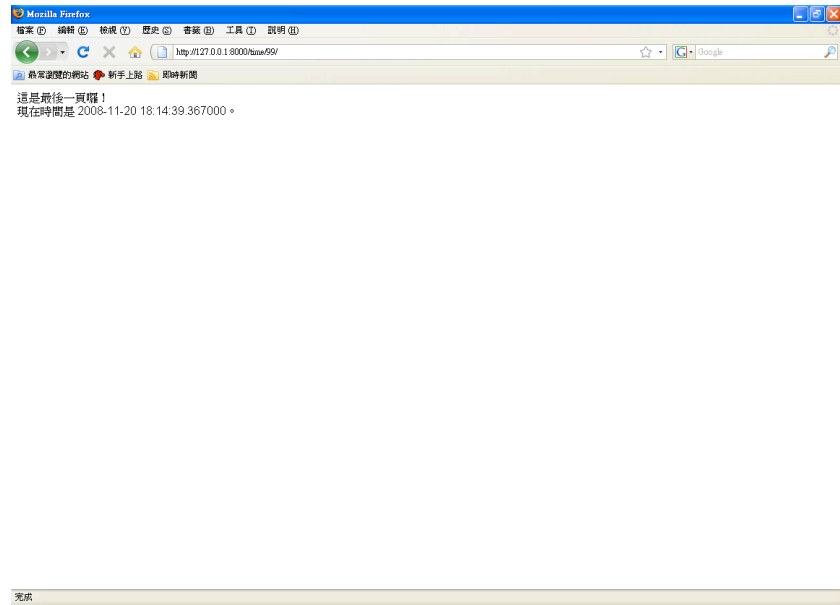
而http://127.0.0.1:8000/time/呢？



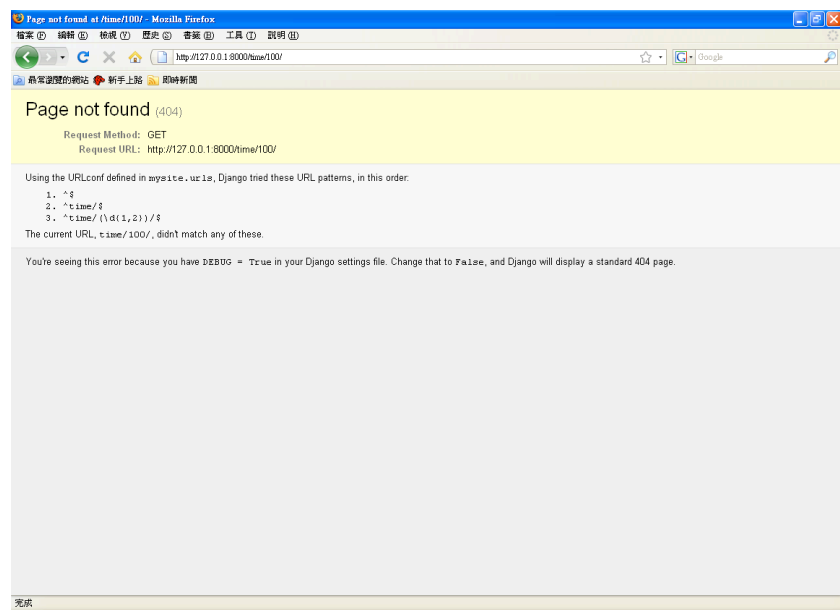
如同之前建立的一般，那第1頁呢？



符合預期，第99頁呢？



仍是與我們的設計相符，如果超出99的範圍，有沒有可能有第100頁呢？



顯然，我們熟悉的404錯誤又出現到我們的眼前。