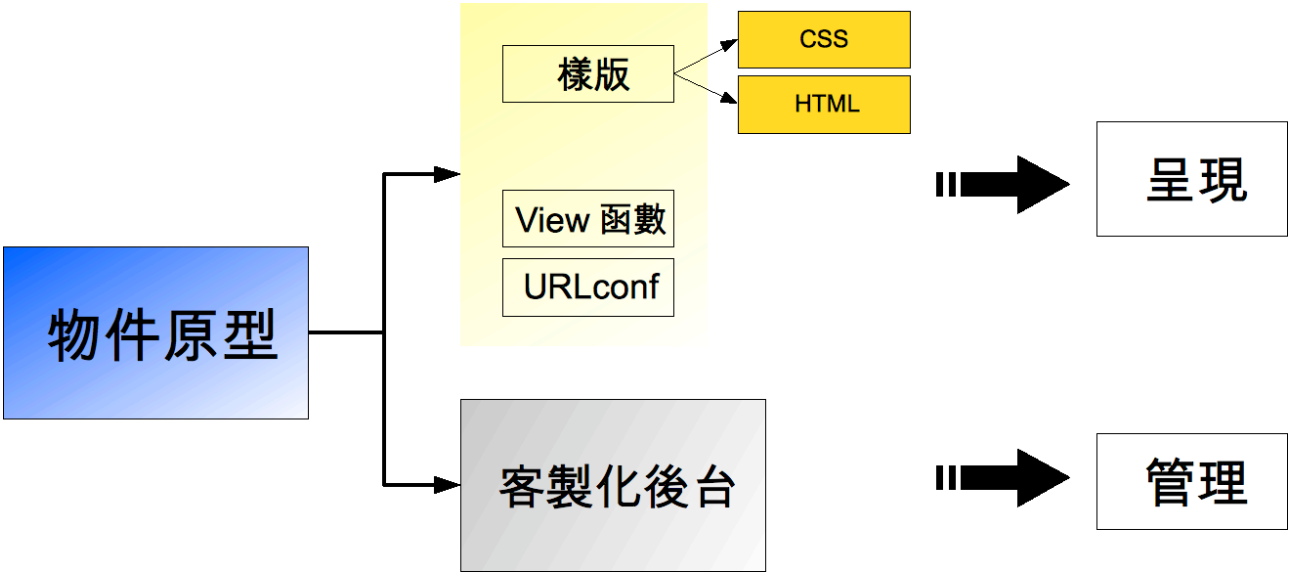
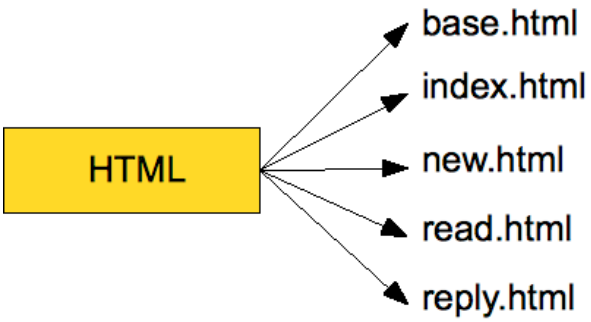
第二十章 上線測試

我們概略複習一下利用Django製作網站的開發流程。



第一步便是規劃物件原型，也就是MTV中的M，其為我們希望透過網頁所要處理的表單資料，在討論板的例子便是models.py裡所定義的Subject與Comment型態，其中的標題、名字、日期與內容等欄位。

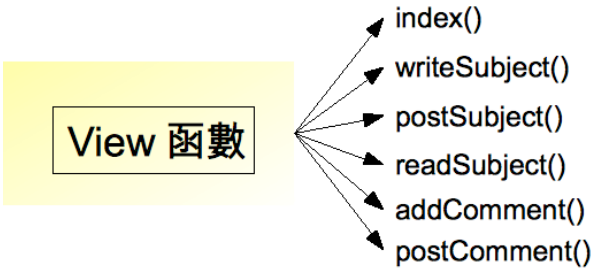
接下來大體分成兩個部份，其一用來呈現網站內容，另一個部份則是利用客製化後台讓我們便於管理網站。前者又可分成三項工作，分別為MTV中作為T的樣版、作為V的View函數，以及設定連結網址的URLconf，三者分別儲存在template資料夾、views.py與urls.py之中。其中，樣版以CSS裝飾，HTML為骨架，樣版語言處理顯示呈現的結果，而CSS儲存在style.css之中，樣版分成五個HTML檔案，如下。



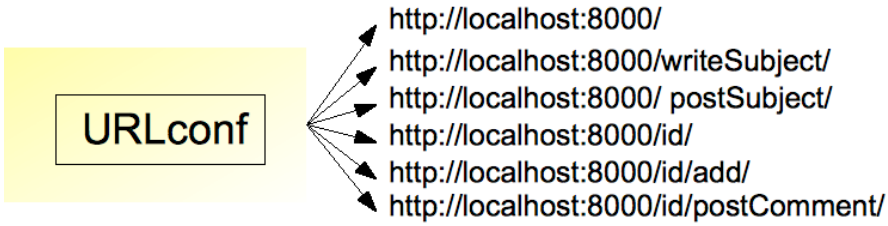
base.html為基礎樣版，所有的HTML包含CSS標籤都放在這裡面，而index.html、new.html、read.html、reply.html都繼承自base.html，使網頁顯示出一致性的外觀，然後這四個樣版使用Django的樣版語言處理所要顯示的資料。

View函數將發表主題及回覆文章的各項資料，如標題、作者、時間、內容等傳送到資料庫中儲存，需要呈現網頁時也將個別所需的資料從資料庫中擷取出來，然後送給樣版，由樣版負責呈現的方式。

我們一共寫了六個View函數，如下。



每個URLconf個別與View函數與樣版搭配，依哪個網址使用哪個View函數，接著View函數再將資料交給樣版，最後我們就可以在瀏覽器上看到結果。URLconf除了管理網站用的http://hocalhost:8000/admin/外，有如下數種。



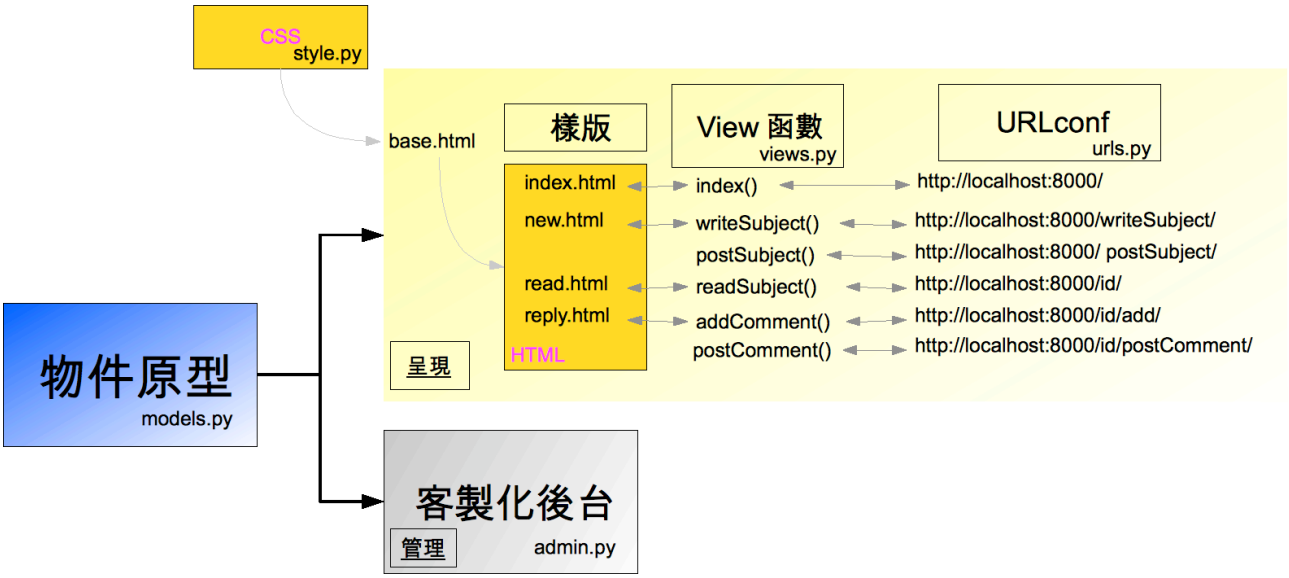
其中，http://hocalhost:8000/的網址對應到index()函數，http://hocalhost:8000/writeSubject/對應到writeSubject()函數，http://hocalhost:8000/id/對應到readSubject()函數，id則指出是哪一篇文章的，http://hocalhost:8000/id/add/則是對應到addComment()函數，如下表。

T	V	URLconf
index.html	index()	http://hocalhost:8000/
new.html	writeSubject()	http://hocalhost:8000/writeSubject/
read.html	readSubject()	http://hocalhost:8000/id/
	postSubject()	http://hocalhost:8000/postSubject/
reply.html	addComment()	http://hocalhost:8000/id/add/
	postComment()	http://hocalhost:8000/id/postComment/

還有兩個View函數分別是postSubject()與postComment()，他們沒有相對應的樣版，卻有連結網址，這是怎麼一回事呢？這是因為連結到某一個網址才會執行該網址所對應的View函數。我們要讓這兩個函數產生作用，例如發表了id為1的文章後，按下〔張貼〕的按鈕後，同時要讓網址更改為http://hocalhost:8000/1/postSubject/，這樣才能執行postSubject()函數，將資料送進資料庫之中，所以這個View函數有網址，卻不會在瀏覽器中顯示任何內容，而當資料處理完畢後，postSubject()回傳資料給http://hocalhost:8000/1/的網址，在瀏覽器中顯示文章內容。

這是在伺服器中進行的運算，過程很快，我們可能不會發現瀏覽器中網址列的變化，沒關係，重點是我們要了解如何把客戶端使用者輸入的資料，傳送給伺服器，然後伺服端程式的運作方式。

如果我們把檔名都加入開發流程圖之中，如下圖。



每個部份的關係都是鬆散的，這正式符合Django的設計哲學之一。

另一個部份，就是利用客製化後台，將後台依據所建立的各種表單資料分門別類，使網站便於管理，有如現在部落格的後台一般，這就讓Django具備現在熱門CMS之類的架站軟體特性。什麼是CMS呢？

CMS

CMS為Content Management System的頭字母縮寫詞，中文為「內容管理系統」，提供使用者不需程式語言的背景也能快速架設網站，廣義來說各類行的網站都有相對應的CMS軟體，諸如入口網站、論壇、電子商務、網誌、線上學習等，這些都有專屬的內容管理系統，如下表。

功能類型	內容管理系統
入口網站	Joomla!、Plone等
wiki	MediaWiki、MoinMoin等
論壇	phpBB、SMF、Discuz!等
電子商務	osCommerce、phpShop等
個人網誌	Lifetype、Wordpress等
線上學習	XOOPS等

上表中絕大多數的內容管理系統都是基於PHP語言，如Joomla!、 MediaWiki、phpBB、XOOPS等，其中有兩個是基於Python語言，分別是延伸自Zope的Plone與wiki應用的MoinMoin。

內容管理系統的特點是各部份都已經開發完善，有預設的樣版、規格，或是能外掛其他的應用程式，其中包括類似個人網誌服務商提供的管理後台。因此對於架站不需程式語言的背景，能夠快速架設功能完整的網站。

聽起來好像很不錯，內容管理系統很快就能把網站做好，然後直接放到網路上供人瀏覽。的確，這便是內容管理系統流行的原因之一，但是當我們需要多一點額外功能的時候，像是內

容管理系統本身沒有的，或是外掛程式中也找不到，那我們還是得自己開發，然後利用外掛或是其他方式與該內容管理系統結合。

這就是網頁框架工作存在的目的了，我們可以利用Django自行開發所有需要的功能，分別放在各個相關的app之中，例如討論版進行擴充，會員註冊系統是一個app，會員個人首頁管理也是一個app，還可以加上會員等級制度的玩法，這也是一個app…….

太多了，不勝枚舉，最重要的是我們都能把每項功能調校成最符合我們的需求。同時，Django也提供了一個管理後台的介面，使網站的管理與後續經營都更為便利，這一點，便是與內容管理系統相似之處。

上線之前，談談LAMP的概念

當網路日益普及，不管個人還是企業或公司行號，架設專屬網站的需求快速增加，為使架設網站的成本降低，許多功能強大的自由軟體成了最佳選擇方案之一。LAMP或是類似LAMP的概念於焉而生，什麼是LAMP呢？

LAMP是一組開放原始碼軟體的頭字母縮寫詞，分別是

- Linux - 作業系統
- Apache - 伺服器
- MySQL - 資料庫
- PHP - 伺服端程式語言

嚴格來說，Linux為作業系統核心的名稱，因其開放原始碼採用GPL授權的特性，因此大部分的版本，如Fedora Core、Ubuntu等，免費並且可自由取得使用，同時由所屬的組織、機機會等發行與維護，以其穩定、靈活與廉價因而常作為伺服器的作業環境，許多網站供應商都採用各自的Linux版本為其系統。

Apache為Apache HTTP Server的簡稱，由Apache軟體基金會維護的開放原始碼的伺服器軟體，原為Netscape的網頁伺服器，後來功能逐漸完備，執行速度也不比其他Unix網頁伺服器慢，又因為可以免費取得，於是被廣為採用，目前是市場佔有率最高的網頁伺服器軟體。

MySQL為資料庫軟體，也可稱為資料庫伺服器。MySQL與我們稍早使用SQLite相似，體積小、速度快，又因開放原始碼的特性，個人或非營利組織可以免費取得，同時兩者都採用SQL，結構化查詢語法進行資料檢索。

PHP與我們使用的Python相同，兩者都作為伺服端的程式語言，作為處理、運算資料之用。LAMP一個典型的例子如維基百科，利用基於PHP的MediaWiki，Linux為其開發的作業環境，伺服器軟體為Apache，資料庫系統選用MySQL。

以這樣的概念延伸，作業系統也可以選用BSD、Mac OS X或MS-Windows，Apache通常不變，或是換成其他的商業伺服器軟體，而資料庫也可選擇如PostgreSQL或是其他的商業資料庫，程式語言也可選擇Python、Perl或其他程式語言。

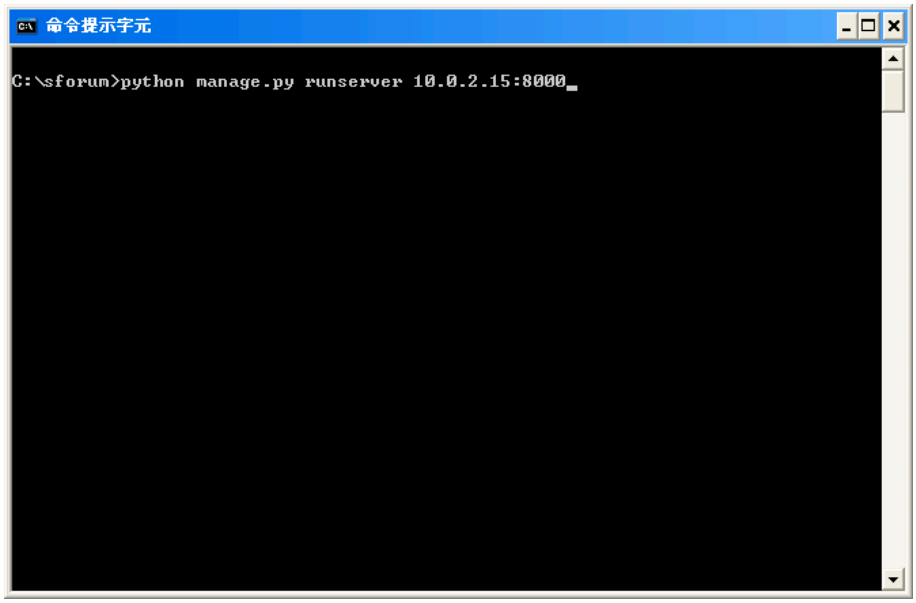
於是LAPP、WAMP或MAMP等其他的頭字母縮寫詞也如雨後春筍般的出現，架設網站當可以自由選擇作業系統、伺服器、資料庫及伺服端程式語言，我們在這裡同樣建議LAMP的組合，不過，最後一個P是由Python替代PHP。

但是我們要將Django開發出的網站直接以LAMP的方式放到網路上嗎？實際上不少國外的主機代管公司都有提供這樣的服務，我們也能自行挑選合乎需求的主機代管公司放置自己做的網站，可能要花一點錢，或著用自己的電腦用LAMP架設也可以。可是LAMP我們畢竟只講了Python的觀念，資料庫則可借用Django的API，Linux及Apache則帶有其他許多複雜的技術，這一章中我們可是講不完的。

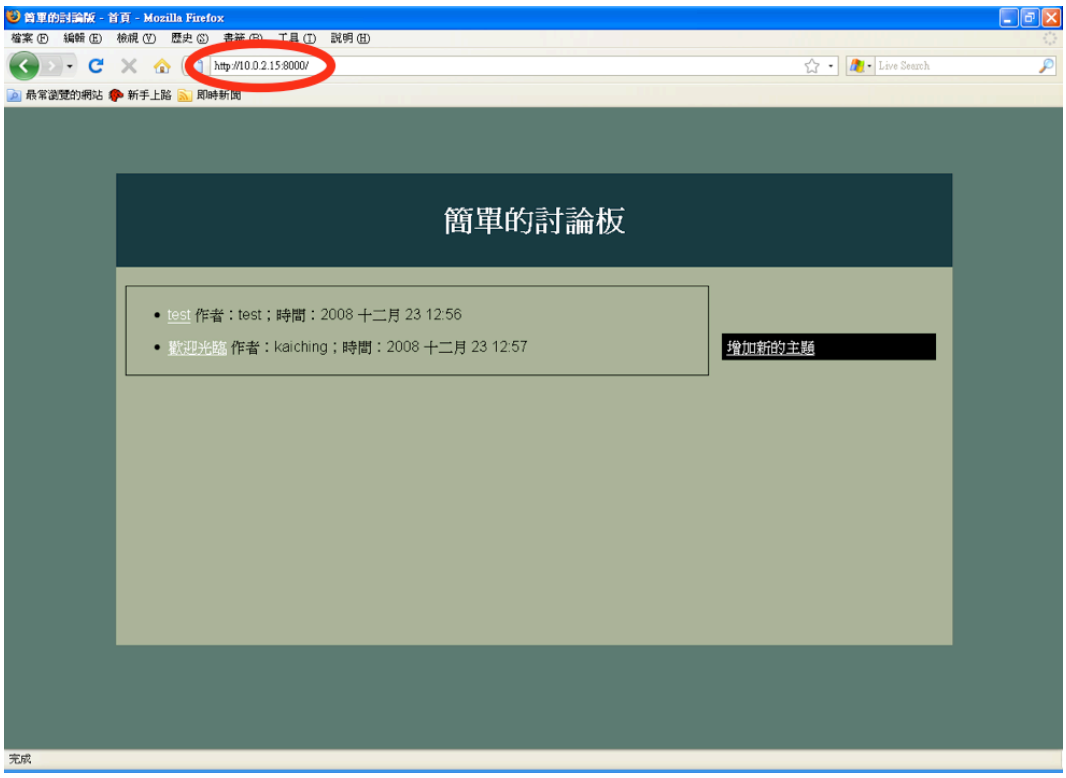
別擔心，若是純粹把剛做出來的網站放到網路上，找朋友分享一下完成自己作品的喜悅，Django所提供的發展中伺服器也辦得到呢！

上線測試

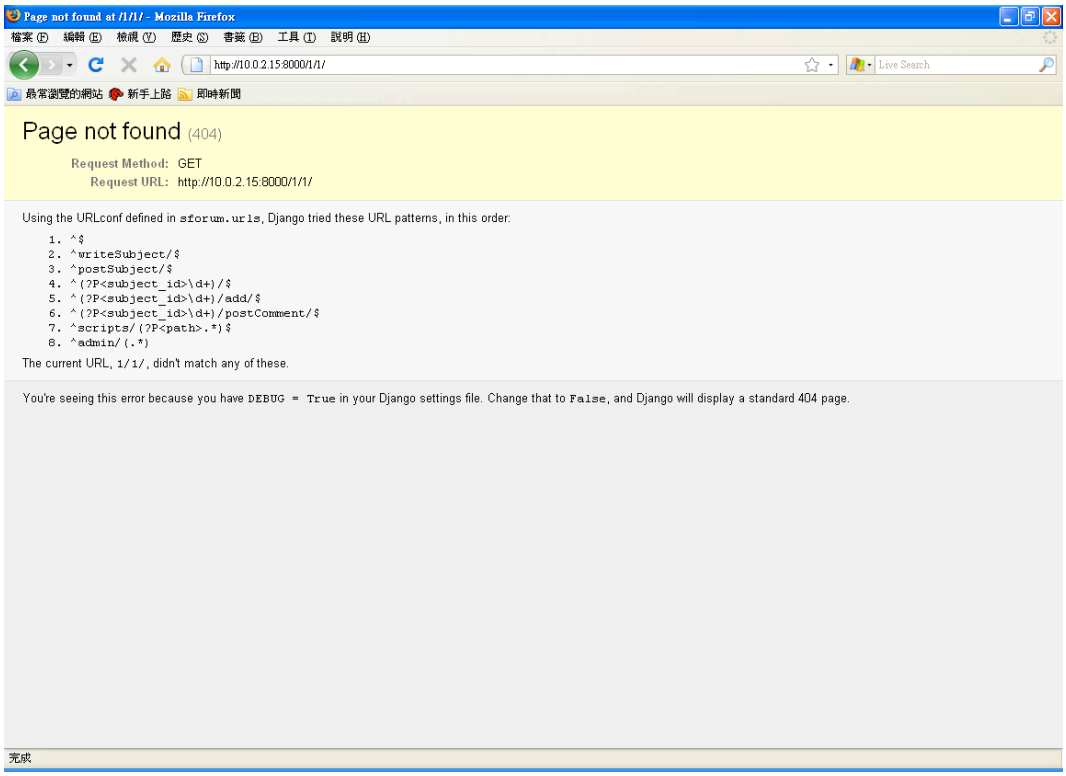
如果我們要將上一章做好的簡單討論版以發展中伺服器放到網路上，我們只需要在「python manage.py runserver」的指令後加上自己電腦的ip位址與埠號。



這樣一來，只要在瀏覽器中輸入ip位置與埠號，就可以連到簡單討論版的首頁了。



沒錯，是不是很簡單呢？但是有個問題，假如我們連錯網址，網頁同時會顯示出Django的除錯結果，如下圖。



除錯結果會把我們在伺服器的資訊透露出來，有些可能不是我們想要告訴使用者的，例如http://10.0.2.15:8000/admin/就可能不是我們想要告訴別人的位置，我們在這個網址登入帳號進入後台管理，自然不希望有入侵者直接在瀏覽器中找到入口。

自訂404錯誤頁

那我們該怎麼做呢？先在views.py加進引入Http404的陳述。

```
from django.http import Http404
```

Http404可以讓我們使用自訂的404錯誤頁，我們很簡單的在跟目錄，也就是sforum資料夾中加入一個404.html，放置如下的內容。

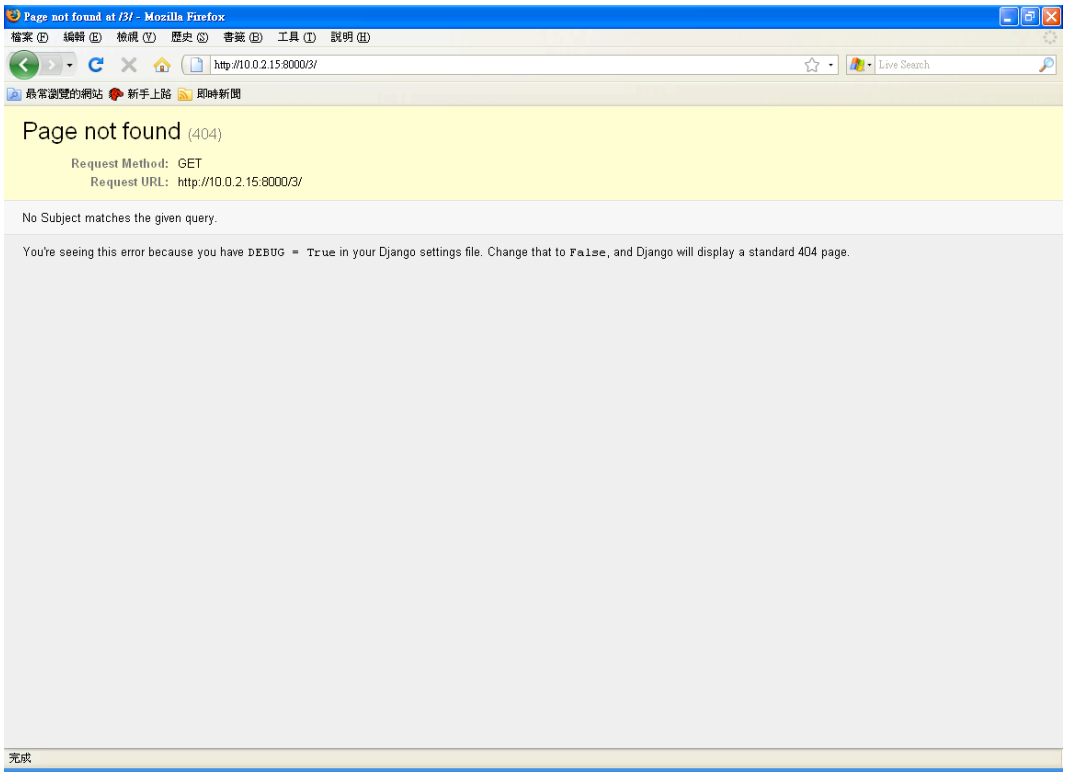
```
這是 404 錯誤頁.....
```

這就是我們自訂的404錯誤頁了，我們希望凡是找不到頁面的情況都會顯示這個錯誤訊息，然後把views.py中的readSubject()函數加入以下的try...except...陳述。

```
def readSubject(request, subject_id):
    try:
        subject = get_object_or_404(Subject, pk=subject_id)
    except Subject.DoesNotExist:
        raise Http404
    return render_to_response('templates/read.html', \
                {'subject': subject, 'comments': subject.comment_set.all()})
```

如果Subject物件不存在於連結到的頁面，readSubject()函數就引起Http404，網頁就會顯示我們自訂的404錯誤頁了。

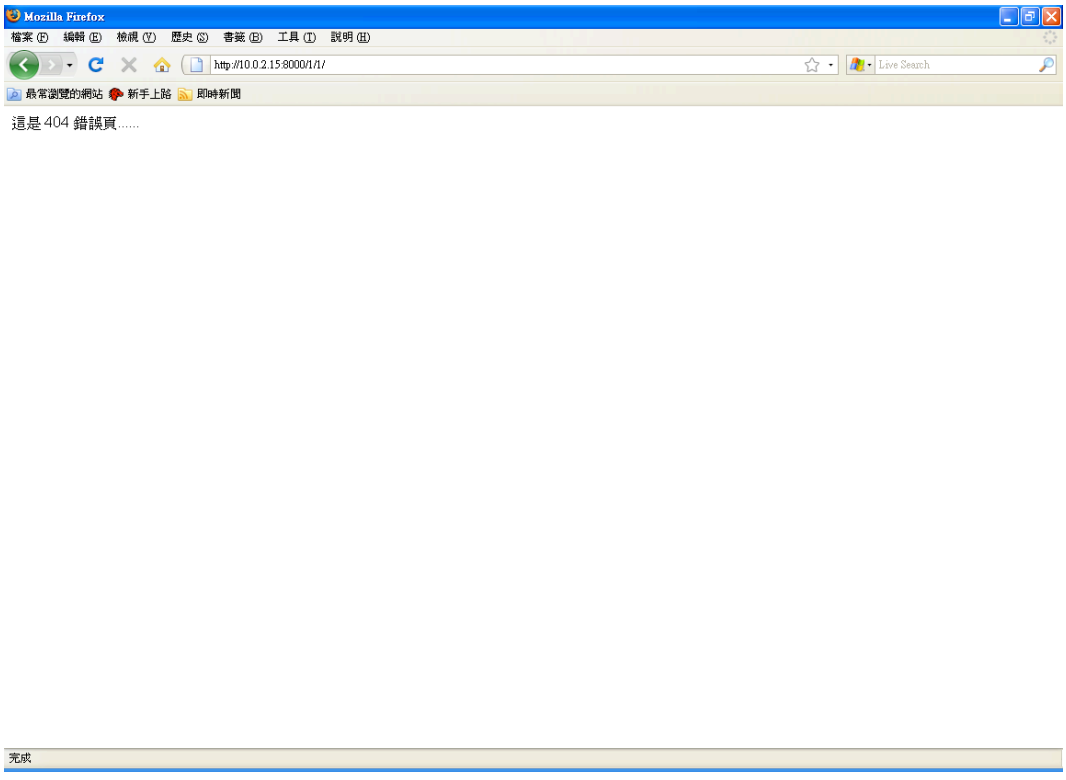
如下，我們重新啟動伺服器，然後連結到一個不存在的頁面。



咦，結果還是跟Django除錯頁一樣，只是少了除錯資訊。為什麼呢？其實這個頁面的最下方一行有告訴我們原因「You're seeing this error because you have DEBUG = True in your Django settings file.」，所以我們要將settings.py其中的一行更改如下。

```
DEBUG = False
```

我們再重新連到一個不存在的頁面試看看。



結果出爐，我們的404錯誤頁正常顯示了。

其他學習資源

我們用「簡單的討論版」作為以Django開發的例子，限於入門教學，所以提及的多是概念，未能深入探討許多主題。還有許多線上學習資源或是書籍可供參考，列舉如下：

- Django documentation：Django官網的線上文件，除了有入門教學外，對於Django各項層面都有詳盡的說明；
- The Django Book：採用GNU Free Documentation License的方式授權，其為Django開發的指南手冊，如其序所言，該書的目的是要讓讀者成為Django的專家，除了可線上閱讀外，也買的到APress的紙版，另外有簡體中文的翻譯；
- Wrox Press出版的*Professional Python Frameworks: Web 2.0 Programming with Django and Turbogears*，以實例做說明，結合各種網頁開發技術，除了 Django 外也介紹Python另一個熱門的網頁框架 - Turbogears；
- Django Step by Step：由Django愛好者limodou所寫的簡體中文線上教材，帶領讀者一步一步學習Django；
- APress出版的*Practical Django Projects*，以開發網誌為實例說明Django的各個層面；
- APress出版的 *Pro Django* ，關於Django特定主題的參考書，深入講解Python與Django相互的關係。

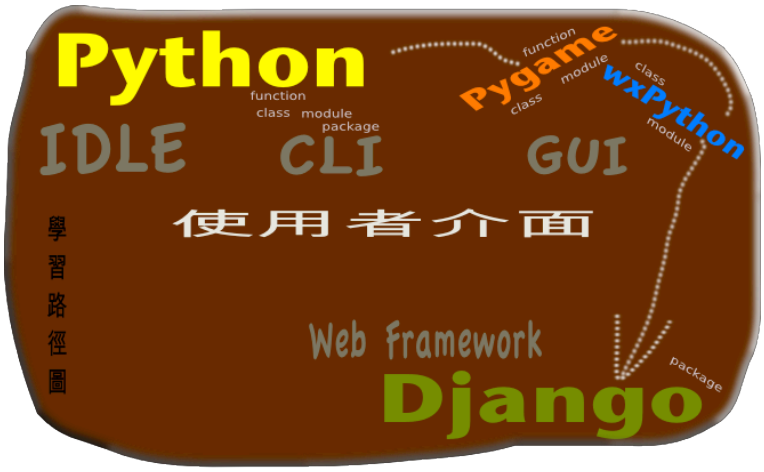
除了學習資源外，還有下列幾個網站可以拜訪一下：

- Django search：可搜尋多種語言版本的語言文章；
- Django People：可看看你的附近哪裡有Django人，截至筆者完稿為止，臺灣地區共有11個Django人；
- Django resources：列出所有Django的資源，包括各種開放原始碼的計畫等；
- django snippets：類似食譜，具有非常多的程式範例；
- pinax；將已經開發完成可重複使用的app提供下載。

仍有許多資源是我們沒有列出來的，譬如還有許多單篇的中文教學，散見在許多個人的部落格裡。雖然我們沒有蒐集整理出列表，這些資源仍是可以藉由Django search搜尋得到。

結語

我們從電腦的基本概念開始，然後進入IDLE學習與Python的直譯器互動，接著直接在命令列（命令提示字元視窗）執行程式，然後又學了圖形使用者介面的Pygame與wxPython，最後學到了網頁框架工作的Django，如下的學習路徑圖：



我們設計此分教材的概念是以「使用者介面」為核心，希望初學者該開始接觸程式語言都能很快的寫出與使用者互動的程式。由於Python簡單、易學的特性，使我們能夠在【第壹篇

Python基礎】中，很快的就把整個語言，寫程式的組織方式**函數→型態→模組→套件**走過一次，雖然細節難免忽略，重點是建立整體的觀念。

【第貳篇 跨過門檻】，我們除了介紹Pygame模組庫外，也全函數或是全型態分別發展不同的程式，使讀者可以比較兩種方式的異同。不意外的，兩種都是物件導向程式設計，因為Python內的所有東西都是物件，差別是全函數的方式利用Pygame所提供的物件，全型態的方式則引導我們做抽象化的思考，使物件的設計近於我們日常生活的想法。

我們在第貳篇，程式的組織方式則為**函數→型態→模組**。【第參篇 窗外的天空】，我們介紹了wxPython模組庫，由於wxPython本身嚴謹的繼承結構，便使應用程式的發展運用全型態的方法，因此在這一篇中程式的組織方式為**型態→模組**。

【第肆篇 打開門，迎向世界】，我們最後介紹Python的網路框架 -- Django的運用，基本上開發一個網站就是一個**套件**的組織，物件原型的部份我們需要自行定義型態，而處理顯示則要規劃View函數，等於是把**函數**、**型態**、**模組**、**套件**都做了複習，同時也學習如何運用Pyhon及Django建構網站。

我們提供了大量的範例，希望藉由大量的範例使讀者更容易體會程式的運作，進而了解電腦背後運作的道理。然而如果自己要能發展精善的程式，唯有練習，從除錯中去找出正確的模式，嘗試各種可能的方法，藉此不斷的練習，逐漸累積自己的能量，才能日益精進。