

# 第十九章 簡單的討論板

這一章我們延續上兩章的留言板，使網頁可依主題進行留言，使之成為討論板，同時以CSS擴充版面風格，所採用的CSS如下。

```
body {
    background: #4F6A5F;
}

a {
    color: #FFFFFFF
}

.wrapper {
    background: #4F6A5F;
    width: 80%;
    padding: 8px;
    margin: 30 auto auto;
    border-style: solid;
    border-width: 0;
}

.heading {
    background: #142D34;
    width: 98%;
    padding: 10px;
    margin: 5 auto auto;
    color: #FFFFFFF
}

.content {
    background:#9BA987;
    width: 98%;
    padding: 10px;
    margin: 0 auto auto;
}

.text {
    font-size: 100%;
    background-color: #FFFFFFF;
    border: thin solid #545454;
    padding: 5px;
    width: 90%;
}

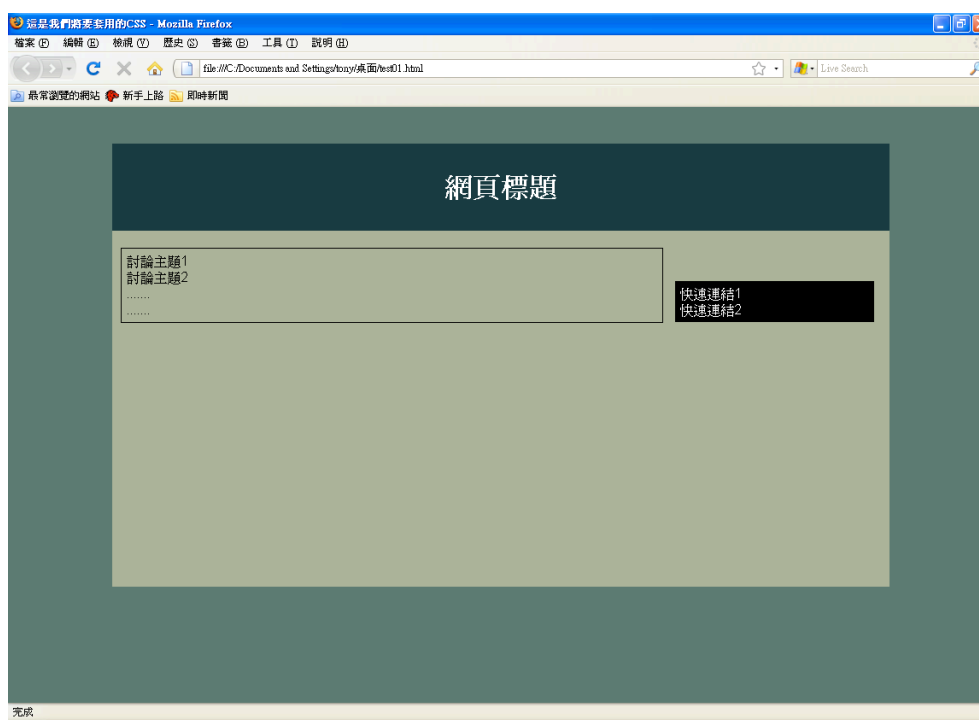
#entries {
    background: #9BA987;
    width: 70%;
    padding: 5px;
    margin: 10 5 auto auto;
    border-style: solid;
    border-width: 1;
    display: inline-block;
}

#sidebar {
    background:#000000;
```

```
width: 25%;  
padding: 5px;  
margin: 10 auto 300 5;  
border-style: solid;  
border-width: 0;  
display: inline-block;  
color: #FFFFFF;  
}
```

HTML的標籤通常帶有許多屬性設定，包括區域範圍、字型大小、背景顏色等，然而每到一個標籤就得重複設定，若有修改網頁檔的時候，過程就會非常的繁瑣、累綴。CSS就是為了減少這個缺點所帶來的影響，將HTML標籤集中管理，在同一區域（或檔案）進行設定。

上述的CSS會有如下的風貌。

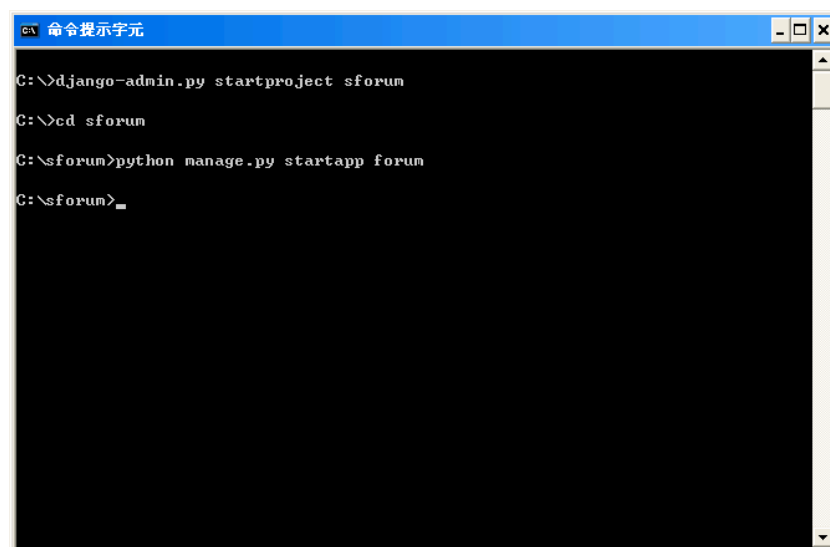


網頁以綠色系為主體，上方標題區為最深的綠色，標題文字為白色，下方分成兩欄，左邊為較寬的主要內容區，右邊為較窄的黑色背景，放置如「回首頁」之類的快速連結。連結文字的顏色設定為白色，主要內容區放置主題或回覆文章，該文章內容區域的背景會改為白色，文字則為黑色。

網頁主要內容，預計分為討論主題索引、張貼新的主題、瀏覽主體內容、回覆文章等。

## 討論版的專案

我們重新建立一個名為「sforum」的專案，然後在專案內建立「forum」的app。



```
C:\>django-admin.py startproject sforum
C:\>cd sforum
C:\sforum>python manage.py startapp forum
C:\sforum>_
```

別忘了要先作一些設定，也就是調整setting.py的內容。先是資料庫的部份

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

時區的部份

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

語系編碼的部份

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

樣版目錄的部份

```
TEMPLATE_DIRS = (
    '..',
)
```

app的部份

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

接下來，我們依M、T、V的順序分別討論如何完成建置這「簡單的討論版」。

## M - 物件模型

一般討論版常見所需的欄位有主題、發表者暱稱、時間與內容等，因此物件模型需要這四個欄位，我們在models.py放置如下的內容。

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

class Subject(models.Model):
    title = models.CharField(max_length=128)
    name = models.CharField(max_length=64)
    date = models.DateTimeField('date')
```

```

        content = models.TextField()

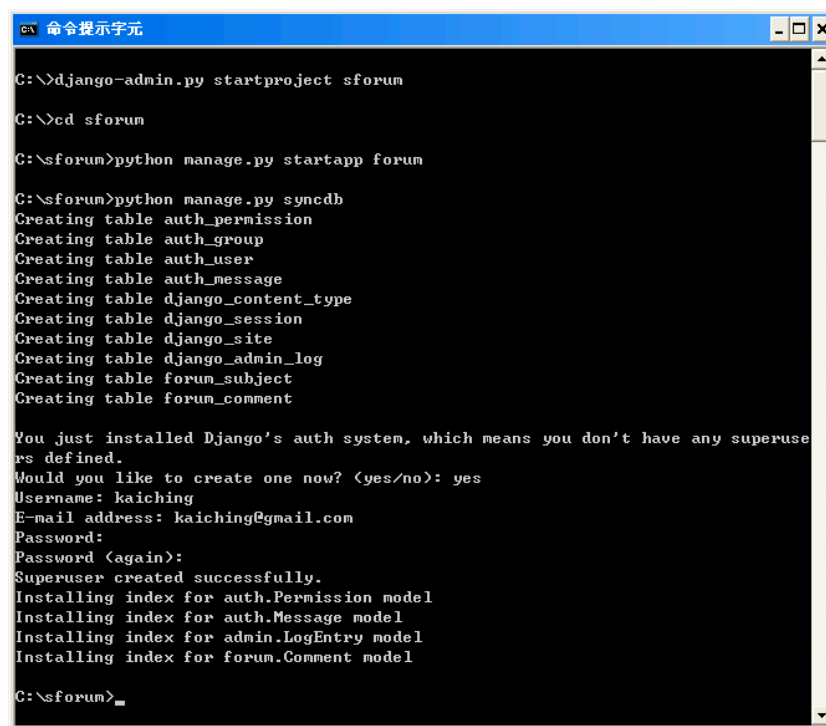
class Comment(models.Model):
    reference = models.ForeignKey(Subject)
    name = models.CharField(max_length=64)
    date = models.DateTimeField('date')
    content = models.TextField()

try:
    admin.site.register(Subject)
    admin.site.register(Comment)
except admin.site.AlreadyRegistered:
    pass

```

我們定義了兩個型態，Subject處理主題，Comment則是處理回覆文章，注意到Comment定義中包含了models.ForeignKey(Subject)，這使每一篇回覆文章都會與所屬主題，也就是Subject產生連結。models.DateTimeField()為時間的欄位，待會我們會以datetime模組中的now()函數得到現在時間，然後儲存到這個欄位之中。

然後，別忘了我們還要設置資料庫檔案。



```

C:\>django-admin.py startproject sforum

C:\>cd sforum

C:\sforum>python manage.py startapp forum

C:\sforum>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 forum_subject
Creating table forum_comment

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: kaiching@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
Installing index for forum.Comment model

C:\sforum>_

```

## T - 樣版

我們分別需要顯示主題索引、張貼新的主題、瀏覽主體內容、回覆文章等，因此總共需要四個樣版。我們將這四個樣版會重複的HTML部份挑出來，寫進如下的base.html，提供給以上樣版繼承之用。

```

<html>

<head>
    <title>{% block title %} 簡單的討論板 {% endblock %}</title>
    <link rel=StyleSheet href="/scripts/style.css" TYPE="text/css">
</head>

<body>

```

```

<div class="wrapper">
  <div class="heading"><center><h1>簡單的討論板</h1></center></div>

  <div class="content">
    <div id="entries">
      {% block content %}{% endblock %}
    </div>
    <div id="sidebar">
      {% block sidebar %}{% endblock %}
    </div>
  </div>
</div>
</body>

</html>

```

我們將稍早的CSS寫到style.css檔案之中，然後在專案資料中建立新的「scripts」資料夾，把style.css儲存到這個資料夾裡，這就是網頁所要套用CSS的路徑「href="/scripts/style.css"」，由於這是屬於伺服器網址的路徑，所以我們待會修改urls.py時，也要將這個路徑加進去。

這裡定義了三個「block」標籤，{% block title %}為網頁標題，出現在瀏覽器上方的標題列，{% block content %}設置主要內容區，{% block sidebar %}則設置側邊欄的快速連結。

以下的樣版作為討論版的首頁，網址預備設置為http://127.0.0.1:8000/，提供所有主體的索引及連結，儲存在index.html之中。

```

{% extends "templates/base.html" %}

{% block title %}
簡單的討論版 - 首頁
{% endblock %}

{% block content %}
  {% if subject_list %}
    <ul>
      {% for subject in subject_list %}
        <li> <a href="{% subject.id %}">{{ subject.title }}</a>
        作者：{{ subject.name }}；時間：{{ subject.date|date:"Y F j f" }}<p/>
        </li>
      {% endfor %}
    </ul>
  {% else %}
    目前尚無主題.....<br />
  {% endif %}
{% endblock %}

{% block sidebar %}
  <a href="writeSubject">增加新的主題</a><br />
{% endblock %}

```

這裡用了個{% if %}標籤，判斷是否有主題在，若有，<ul>與<li>標籤便以列表的方式，在網頁中顯示主題的目錄，若是沒有也會顯示「目前尚無主題.....」。其中時間用了過濾器「date」，Y、F、j、f依序表示年、月、日、時。

側邊欄則是提供「增加新的主題」的連結，該連結的網址為http://127.0.0.1:8000/writeSubject/，待會這個網址要放置發表主題的樣版。

以下的樣版作為發表主題之用，儲存在new.html之中。

```

{% extends "templates/base.html" %}

{% block title %}

```

## 簡單的討論版 - 發表新的主題

```
{% endblock %}

{% block content %}
    <form action="/postSubject/" method="post">
        主題：
        <input type="text" name="title"> <br/>
        作者：
        <input type="text" name="name"> <br/>
        內容：
        <textarea rows="10" cols="70" name="content"></textarea>
        <p/>
        <input type="submit" value="張貼">
    </form>
{% endblock %}

{% block sidebar %}
    <a href="/">回到首頁</a><br />
{% endblock %}
```

<form>為表單的HTML標籤，利用表單的方式，我們可以直接在前台傳輸資料給伺服器，然後顯示網頁內容。<input>標籤接受使用者的輸入，type屬性text為文字，submit是按鈕。<textarea>則是接受多行文字輸入的標籤。

當使用者輸入完成，點擊按鈕「張貼」，表單中輸入的資料就會回傳給伺服器，網址會連結到<form>的屬性action指定的位置，這會是網址http://127.0.0.1:8000/postSubject/。另外需要一個函數處理這些由前端來的資料，至於是什麼樣的函數稍待一會。

這裡，側邊欄提供「回到首頁」的連結，如果在發表主題時突然打算放棄，其連結可以離開發表頁，所輸入的資料也就不會儲存到資料庫之中。

以下是瀏覽單獨主題的樣版，儲存在read.html之中。

```
{% extends "templates/base.html" %}

{% block title %}
簡單的討論版 - {{ subject.title }}
{% endblock %}

{% block content %}
    <h1>主題名稱：{{ subject.title }}</h1><br />
    作者：{{ subject.name }}；時間：{{ subject.date|date:"F j, Y" }}<br />
    <p><div class="text">{{subject.content}}</div></p>
    <br />

    {% if comments %}
        <p>** 回覆文章 **</p>
        {% for comment in comments %}
            作者：{{ comment.name }}；時間：{{ comment.date|date:"F j, Y" }}<br />
            <p><div class="text">{{ comment.content }}</div></p>
        {% endfor %}
    {% else %}
        <p>目前沒有回覆文章喔！</p>
    {% endif %}
{% endblock %}

{% block sidebar %}
    <a href="/">回到首頁</a><br />
    <a href="add">回覆文章</a><br />
{% endblock %}
```

最後會把回覆文章都顯示出來，同樣的，這裡也用了{% if %}標籤，如果沒有回覆文章，就單純的顯示「目前沒有回覆文章喔！」。側邊欄則有「回到首頁」與「回覆文章」兩個連結，若在第一個主題中點擊「回覆文章」的話，網址會連到http://127.0.0.1:8000/1/add/。

以下為第四個樣版，發表回覆文章之用，儲存在reply.html之中。

```
{% extends "templates/base.html" %}

{% block title %}
簡單的討論版 - 發表回覆文章
{% endblock %}

{% block content %}
<form action="{% subject.id %}/postComment/" method="post">
  <h1>主題名稱：{% subject.title %}</h1>
  回覆作者：
  <input type="text" name="name"> <br/>
  回覆內容：
  <textarea rows="10" cols="70" name="content"></textarea>
  <input type="submit" value="張貼" />
</form>
{% endblock %}

{% block sidebar %}
  <a href="/">回到首頁</a><br />
{% endblock %}
```

回覆文章與發表主題的樣版類似，差別是前者直接顯示主題名稱，另外連結網址也要稍作注意，如果在第一個主題發表回覆，點擊最後「張貼」的按鈕，網址會連到http://127.0.0.1:8000/1/postComment/，同樣的，這也需要一個函數處理回覆的資料，我們待會再提。

## V - View函數與URLconfs

由於網頁的顯示是在view.py加入控制的函數，實際要顯示又得在urls.py中替urlpatterns加入新的連結設定，這部份有點複雜，別擔心，我們一步一步來完成。

view.py引入的部份如下。

```
from django.shortcuts import render_to_response, get_object_or_404
from sforum.forum.models import Subject, Comment
from django.http import HttpResponseRedirect
from datetime import datetime
```

有些新的東西，待會再來解釋，我們先在urls.py加入以下的連結設定。

```
(r'^scripts/(?P<path>.*)$', 'django.views.static.serve', \
    {'document_root': './scripts'}),
```

這便是載入放在scripts資料夾中CSS檔案的連結設定，少了這部份，CSS的版面風格就無法出現。緊接著來看看首頁，也就是所有討論主題的索引頁，我們在view.py加入以下的index()函數。

```
def index(request):
    subject_list = Subject.objects.all().order_by('date')[:]
    return render_to_response('templates/index.html', \
        {'subject_list': subject_list})
```

Subject.objects.all()可以取得所有的Subject物件，然後利用order\_by('date')[:]來排序，這些資料都被儲存到變數subject\_list之中，接著以render\_to\_response()與樣版結合輸出到網頁前台。

urls.py裡要加入以下的連結設定。

```
(r'^$', 'sforum.forum.views.index'),
```

然後啟動伺服器，連結到http://127.0.0.1:8000/的網址。



的確，因為還沒有主題，所以首頁的內容為「目前尚無主題.....」。我們來看看如何新增主題吧！在view.py加入以下的writeSubject()函數。

```
def writeSubject(request):  
    return render_to_response('templates/new.html')
```

這是要作為顯示new.html的樣版之用，urls.py裡要加入以下的連結設定。

```
(r'^writeSubject/$', 'sforum.forum.views.writeSubject'),
```

兩個檔案都儲存之後，我們可以點擊「增加新的主題」的快速連結，結果如下。





但是實際按下「張貼」後，網址就會連結到`http://127.0.0.1:8000/postSubject/`，我們還需要在`views.py`中定義`postSubject()`函數，伺服器才能正確的處理資料。

```
def postSubject(request):  
    s = Subject()  
    s.title = request.POST['title']  
    s.name = request.POST['name']  
    s.content = request.POST['content']  
    s.date = datetime.now()  
    s.save()  
    return HttpResponseRedirect('%s/' % s.id)
```

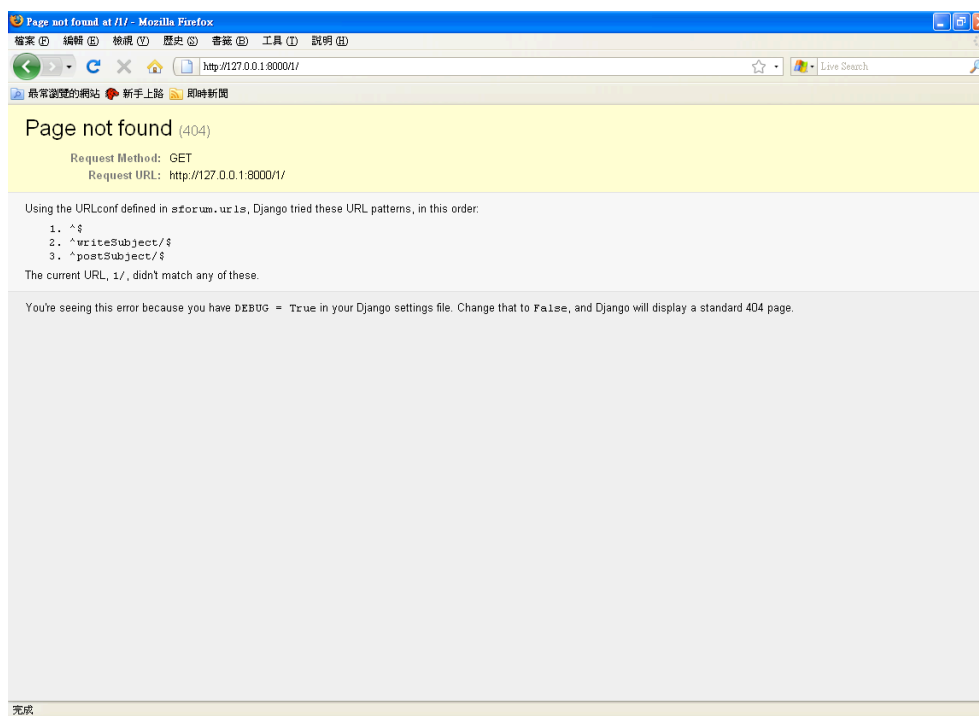
函數`postSubject()`裡先建立一個`Subject`物件，然後由該物件取得網頁前台輸入的`title`、`name`與`content`三個屬性，並以`datetime.now()`取得現在時間儲存到`date`屬性之中，這些動作完成後在以`save()`方法儲存`Subject`物件，最後以`HttpResponseRedirect()`在預設的網址顯示該網頁內容。

當然，`urls.py`裡要加入以下的連結設定，這是要讓`postSubject()`函數產生作用。

```
(r'^postSubject/$', 'sforum.forum.views.postSubject'),
```

別忘了改完要先存檔。

然後，我們在主題、作者、內容都輸入「test」來試看看！



找不到網頁，為什麼呢？因為我們還沒有做瀏覽主題的設定，沒關係，先連回首頁看看！



沒錯，主題test已經存在。因此，要瀏覽主題的話，我們還需要在views.py加入函數readSubject()的定義。

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

雖然還沒定義回覆文章的處理函數，不過由於我們打算把回覆文章直接放在主題文章的下方，所以回傳的資料也要包括回覆文章的部份。另外，urls.py裡也要加入以下的連結設定。

```
(r'^(?P<subject_id>\d+)/$', 'sforum.forum.views.readSubject'),
```

存檔後來看看吧！



主題正常顯示了，我們再來看看如何加入回覆文章。這與writeSubject()函數類似，稍微不同的是我們必須替回覆文章找到相對應的主題。

```
def addComment(request, subject_id):
    subject = get_object_or_404(Subject, pk=subject_id)
    return render_to_response('templates/reply.html', {'subject': subject})
```

其實不麻煩，利用get\_object\_or\_404()取得該主題Subject物件的資料，然後一起回傳給樣版就可以了。我們還需要一個類似postSubject()的函數，如下。

```
def postComment(request, subject_id):
    s = get_object_or_404(Subject, pk=subject_id)
    c = s.comment_set.create(name=request.POST['name'], \
                             content=request.POST['content'], date=datetime.now())
    c.save()
    return HttpResponseRedirect('/%s/' % s.id)
```

函數postComment()先由get\_object\_or\_404()取得該主題的Subject物件，儲存到變數s之中，然後利用create()方法取得使用者輸入的name、content、date三個屬性，放到變數c之中後用save()方法儲存該Comment物件，這裡仍是用HttpResponseRedirect()回傳，使我們可以看到回覆文章的結果。

別急著會產生什麼樣的效果，我們還要替urls.py增加兩個連結設定。

```
(r'^(?P<subject_id>\d+)/add/$', 'sforum.forum.views.addComment'),
(r'^(?P<subject_id>\d+)/postComment/$', 'sforum.forum.views.postComment'),
```

好了，來回覆文章看看吧！我們點擊回覆文章的快速連結。



然後在回覆作者與回覆內容都輸入「demo」，按下「張貼」。



這個「簡單的討論版」大體完工，最後，我們將views.py的程式內容摘錄如下。

```
from django.shortcuts import render_to_response, get_object_or_404
from sforum.forum.models import Subject, Comment
from django.http import HttpResponseRedirect
from datetime import datetime

def index(request):
    subject_list = Subject.objects.all().order_by('date')[:]
    return render_to_response('templates/index.html', \
                              {'subject_list': subject_list})

def writeSubject(request):
    return render_to_response('templates/new.html')

def postSubject(request):
    s = Subject()
    s.title = request.POST['title']
    s.name = request.POST['name']
    s.content = request.POST['content']
    s.date = datetime.now()
    s.save()
    return HttpResponseRedirect('/%s/' % s.id)

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

def addComment(request, subject_id):
    subject = get_object_or_404(Subject, pk=subject_id)
    return render_to_response('templates/reply.html', \
                              {'subject': subject})

def postComment(request, subject_id):
    s = get_object_or_404(Subject, pk=subject_id)
```

```

c = s.comment_set.create(name=request.POST['name'], \
                          content=request.POST['content'], date=datetime.now())
c.save()
return HttpResponseRedirect('/%s/' % s.id)

```

urls.py如下。

```

from django.conf.urls.defaults import *

urlpatterns = patterns('',
    (r'^$', 'sforum.forum.views.index'),
    (r'^writeSubject/$', 'sforum.forum.views.writeSubject'),
    (r'^postSubject/$', 'sforum.forum.views.postSubject'),
    (r'^(?P<subject_id>\d+)/$', 'sforum.forum.views.readSubject'),
    (r'^(?P<subject_id>\d+)/add/$', 'sforum.forum.views.addComment'),
    (r'^(?P<subject_id>\d+)/postComment/$', \
     'sforum.forum.views.postComment'),
    (r'^scripts/(?P<path>.*)$', 'django.views.static.serve',
     {'document_root': './scripts'}),
)

```

## 網站的管理

我們這個「簡單的留言板」允許任何人瀏覽、發表主題以及回覆文章，那網站要怎麼進行管理呢？當然，我們可以在urls.py中放置以下的程式碼

```

from django.contrib import admin
admin.autodiscover()

```

然後在urlpatterns中加入以下的連結設定

```

(r'^admin/(.*)', admin.site.root),

```

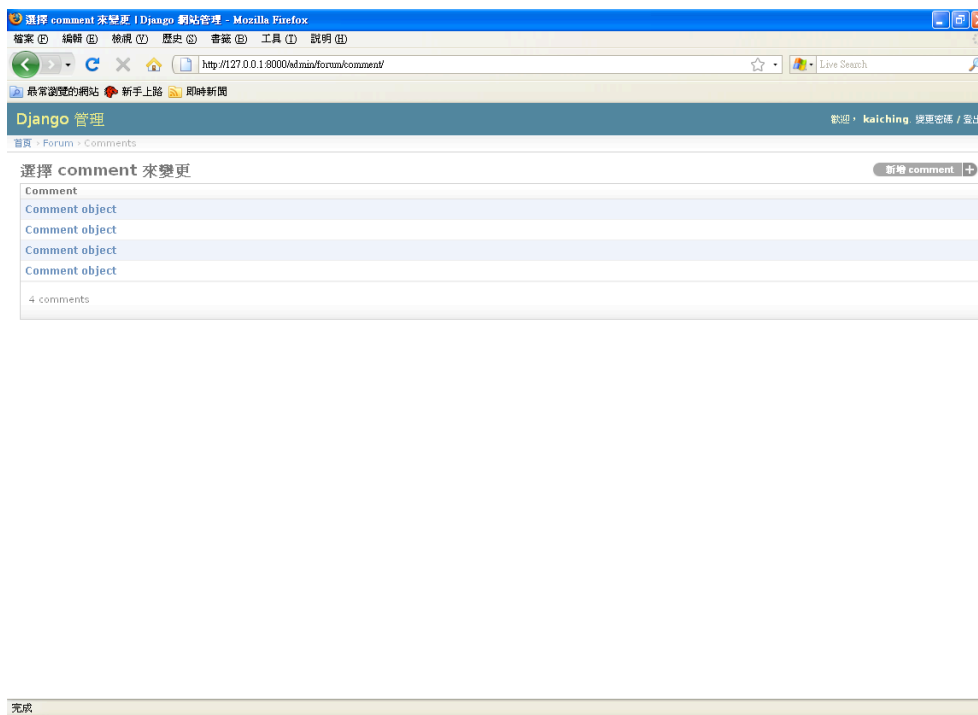
如此一來，我們就可以到後台去進行網站的管理了。

進入後台之前，我們先增加一個新的主題「歡迎光臨」，作者為「kaiching」，內容為「請大家多多發表新的主題喔！」。然後加入三篇回覆文章，第一篇作者、內容分別為「Python愛好者」、「恭喜開版！」，第二篇作者為「John」，內容留白，第三篇作者、內容則是「初學者」、「恭喜開版啊！咦，上一篇怎麼沒有內容？」。結果如下。



目前沒有設置任何的機制防止空白的發言，所以會造成這樣的問題。雖然身為網站管理者擁有絕對的權力片面修改或移除任何發言，但是基於討論板是一個公開的園地，每個人都有平等的立場，設立板規會是一個理想的作法。

在設立板規之前，假設我們可以跟發表者取得聯絡，作者John同意刪除他的發表內容，另一個作者初學者也願意刪除「咦，上一篇怎麼沒有內容？」的文字，我們就進入後台進行修改。實際進入後台後，我們看到Comment物件如下的列表。



雖然可以一個一個的點進去看Comment物件實際的內容，不過這樣子顯然有點麻煩。其實，我們可以客製化後台，直接從列表看到物件內容。要怎麼做呢？首先，在app的地方，也就是forum資料夾內建立admin.py檔案，然後加入以下的內容。

```
from django.contrib import admin
from sforum.forum.models import Subject, Comment

class SubjectAdmin(admin.ModelAdmin):
    list_display = ('title', 'name', 'date', 'content')
    list_filter = ('date',)
    ordering = ('-date',)

class CommentAdmin(admin.ModelAdmin):
    list_display = ('name', 'date', 'content')
    list_filter = ('date',)
    ordering = ('-date',)

try:
    admin.site.register(Subject, SubjectAdmin)
    admin.site.register(Comment, CommentAdmin)
except admin.site.AlreadyRegistered:
    pass
```

SubjectAdmin繼承自admin.ModelAdmin，其為針對後台管理介面處理的物件。這裡用了三個屬性，list\_display為列表顯示的內容，list\_filter則側邊欄供過濾的選項，ordering為排列的方式，這裡是以date屬性來排列。

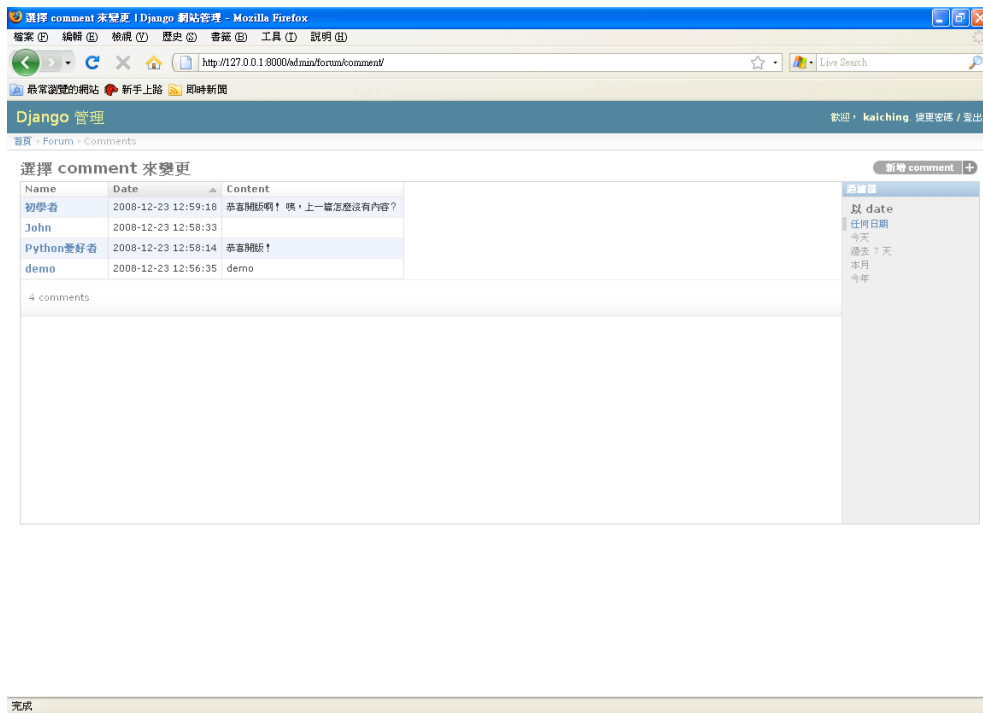
這裡要留意，由於admin.py最後利用admin.site.register()跟資料庫註冊物件，所以原先models.py中的

```
try:
    admin.site.register(Subject)
    admin.site.register(Comment)
```

```
except admin.site.AlreadyRegistered:
    pass
```

要刪除或是註解化，使之不會互相衝突。

因為新增了一個檔案，所以我們需要結束原先伺服器的運作，然後重新啟動伺服器，進入後台連結到Comment物件的列表，如下，是不是清楚多了呢？



然後修改內容，結果如下。

