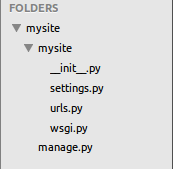
Django 自称是“最适合开发有限期的完美WEB框架”。本文参考《Django web开发指南》，快速搭建一个blog 出来，在中间涉及诸多知识点，这里不会详细说明，如果你是第一次接触Django ，本文会让你在感性上对Django有个认识，完成本文操作后会让你有兴趣阅读的相关书籍和文档。  
  
废话少说，come on!!  
  
本操作的环境：  
  
===================  
  
deepin linux 2013（基于ubuntu）  
  
python 2.7  
  
Django 1.6.2  
  
===================  
  
**创建工程**  
  
创建mysite工程项目：

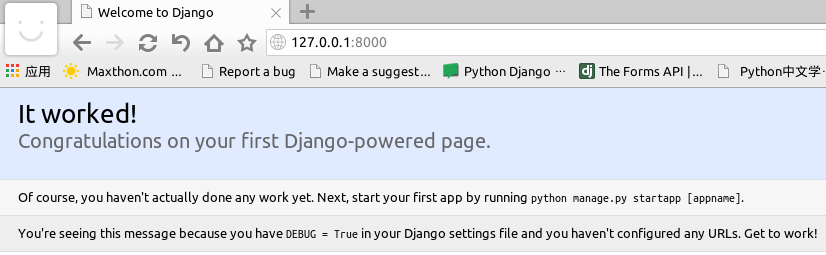
1. fnngj@fnngj-H24X:~/djpy$ django-admin.py startproject mysite

*复制代码*

工程目录结构：  
  
  
manage.py ----- Django项目里面的工具，通过它可以调用django shell和数据库等。  
  
settings.py ---- 包含了项目的默认设置，包括数据库信息，调试标志以及其他一些工作的变量。  
  
urls.py ----- 负责把URL模式映射到应用程序。  
  
  
运行服务：

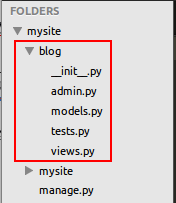
1. fnngj@fnngj-H24X:~/djpy$ cd mysite/
2. fnngj@fnngj-H24X:~/djpy/mysite$ python manage.py runserver
3. Validating models...
4. errors found
5. May 19, 2014 - 13:16:51
6. Django version 1.6.2, using settings 'mysite.settings'
7. Starting development server at http://127.0.0.1:8000/
8. Quit the server with CONTROL-C.

*复制代码*

浏览器访问<http://127.0.0.1:8000/>  
  
  
  
**创建blog应用**                                                                                                     
  
在mysite目录下创建blog应用

1. fnngj@fnngj-H24X:~/djpy/mysite$ python manage.py startapp blog

*复制代码*

目录结构：  
  
  
设计你的Model  
  
现在我们打开blog目录下的models.py文件，这是我们定义blog数据结构的地方。

1. from django.db import models
2. # Create your models here.
3. class BlogsPost(models.Model):
4. title = models.CharField(max\_length = 150)
5. body = models.TextField()
6. timestamp = models.DateTimeField()

*复制代码*

创建BlogsPost类，继承django.db.models.Model父类，定义三个变量，title （博客标题），body（博客正文），timestamp（博客创建时间）  
  
  
**设置数据库**                                                                                                  
  
　　python 自带SQLite数据库，Django支持各种主流的数据库，这里为了方便推荐使用SQLite，如果使用其它数据库请在settings.py文件中设置。  
  
切换到mysite创建数据库：

1. fnngj@fnngj-H24X:~/djpy/mysite$ python manage.py syncdb
2. Creating tables ...
3. Creating table django\_admin\_log
4. Creating table auth\_permission
5. Creating table auth\_group\_permissions
6. Creating table auth\_group
7. Creating table auth\_user\_groups
8. Creating table auth\_user\_user\_permissions
9. Creating table auth\_user
10. Creating table django\_content\_type
11. Creating table django\_session
12. You just installed Django's auth system, which means you don't have any superusers defined.
13. Would you like to create one now? (yes/no): yes    输入yes/no
14. Username (leave blank to use 'fnngj'):    用户名（默认当前系统用户名）
15. Email address: fnngj@126.com    邮箱地址
16. Password:      密码
17. Password (again):      确认密码
18. Superuser created successfully.
19. Installing custom SQL ...
20. Installing indexes ...
21. Installed 0 object(s) from 0 fixture(s)

*复制代码*

**设置admin应用**                                                                                      
  
　　admin 是Django 自带的一个后台管理系统。  
  
1、添加blog应用，打开mysite/mysite/settings.py 文件：

1. # Application definition
2. INSTALLED\_APPS = (
3. 'django.contrib.admin',
4. 'django.contrib.auth',
5. 'django.contrib.contenttypes',
6. 'django.contrib.sessions',
7. 'django.contrib.messages',
8. 'django.contrib.staticfiles',
9. 'blog',
10. )

*复制代码*

在列表末尾，添加blog 应用  
  
2、在我们创建django项目时，admin就已经创建，打开mysite/mysite/urls.py文件：

urlpatterns = patterns('',  
    # Examples:  
    # url(r'^$', 'mysite.views.home', name='home'),  
    # url(r'^blog/', include('blog.urls')),  
  
    url(r'^admin/', include(admin.site.urls)),  
)

去掉admin 设置人注释。  
  
3、将我们创建数据添加到admin后台  
  
再次打开mysite/blog/models.py 文件进行修改：

1. from django.db import models
2. from django.contrib import admin
3. # Create your models here.
4. class BlogsPost(models.Model):
5. title = models.CharField(max\_length = 150)
6. body = models.TextField()
7. timestamp = models.DateTimeField()
8. admin.site.register(BlogsPost)

*复制代码*

4、再次初始化数据库

1. fnngj@fnngj-H24X:~/djpy/mysite$ python manage.py syncdb
2. Creating tables ...
3. Creating table blog\_blogspost
4. Installing custom SQL ...
5. Installing indexes ...
6. Installed 0 object(s) from 0 fixture(s)

*复制代码*

5、启动服务

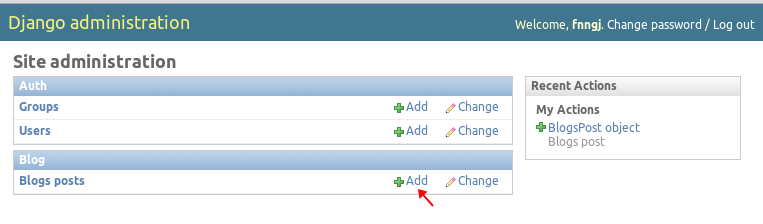
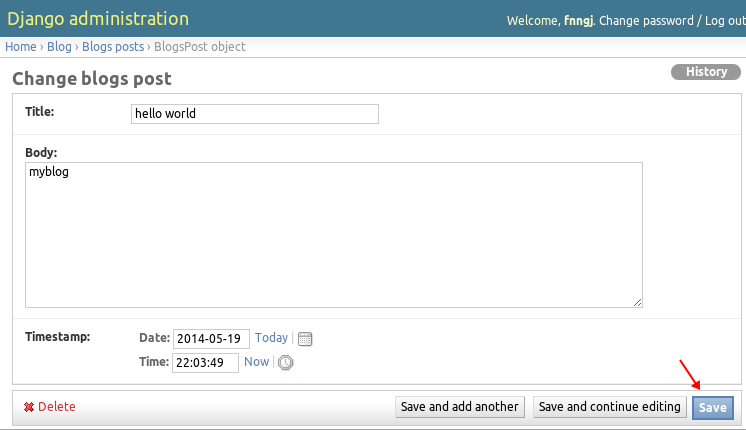
1. fnngj@fnngj-H24X:~/djpy/mysite$ python manage.py runserver
2. Validating models...
3. errors found
4. May 19, 2014 - 14:07:25
5. Django version 1.6.2, using settings 'mysite.settings'
6. Starting development server at http://127.0.0.1:8000/
7. Quit the server with CONTROL-C.

*复制代码*

6、访问admin 后台

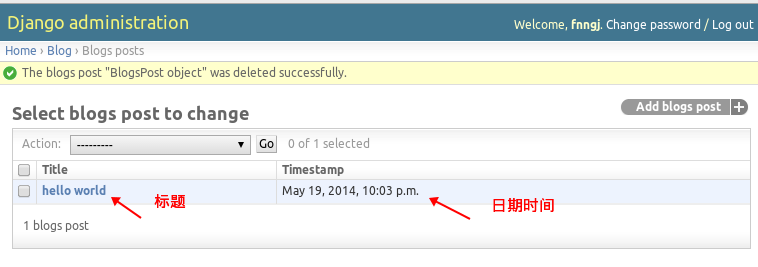
1. http://images.cnitblog.com/i/311516/201405/192322321374972.png

*复制代码*

输入用户、密码，用户名密码为第一次创建数据库时创建的。回想“设置数据库”时的设置。  
  
7、创建博客  
  
登陆成功选择add 创建博客  
  
输入博客标题，正文、日期时间、点击save 创建博客。  
  
  
**设置admin 的BlogsPost界面**                                                                
  
　　打开mysite/blog/models.py 文件，做如下修改：

1. from django.db import models
2. from django.contrib import admin
3. # Create your models here.
4. class BlogsPost(models.Model):
5. title = models.CharField(max\_length = 150)
6. body = models.TextField()
7. timestamp = models.DateTimeField()
8. class BlogPostAdmin(admin.ModelAdmin):
9. list\_display = ('title','timestamp')
11. admin.site.register(BlogsPost, BlogPostAdmin)

*复制代码*

创建BlogPostAdmin类，继承admin.ModelAdmin父类，以列表的形式显示BlogPost的标题和时间。  
  
  
**创建blog的公共部分**  
  
从Django的角度看，一个页面具有三个典型的组件：  
  
一个模板（template）：模板负责把传递进来的信息显示出来。  
  
一个视图（viw）：视图负责从数据库获取需要显示的信息。  
  
一个URL模式：它负责把收到的请求和你的试图函数匹配，有时候也会向视图传递一些参数。  
  
**创建模板**  
  
在blog项目下创建templates目录（mysite/blog/templates/）,在目录下创建模板文件archive.html，内容如下：

1. {% for post in posts %}
2. <h2>{{ post.title }}</h2>
3. <p>{{ post.timestamp }}</p>
4. <p>{{ post.body }}</p>
5. {% endfor%}

*复制代码*

设置模板路径，打开mysite/mysite/settings.py文件，在文件底部添加模板路径：

1. #template
2. TEMPLATE\_DIRS=(
3. '/home/fnngj/djpy/mysite/blog/templates'
4. )

*复制代码*

**创建视图函数**                                                                                            
  
打开mysite/blog/views.py文件：

1. from django.shortcuts import render
2. from django.template import loader,Context
3. from django.http import HttpResponse
4. from blog.models import BlogsPost
5. # Create your views here.
6. def archive(request):
7. posts = BlogsPost.objects.all()
8. t = loader.get\_template("archive.html")
9. c = Context({'posts':posts})
10. return HttpResponse(t.render(c))

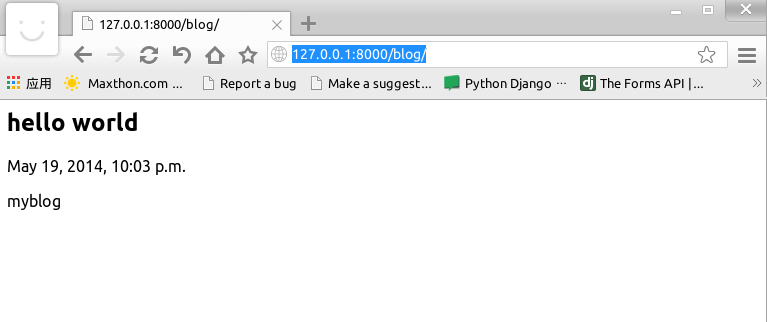
*复制代码*

posts = BlogPost.objects.all() ：获取数据库里面所拥有BlogPost对象  
  
t = loader.get\_template("archive.html")：加载模板  
  
c = Context({'posts':posts})：模板的渲染的数据是有一个字典类的对象Context提供，这里的是一对键值对。  
  
**创建blog的URL模式**                                                                               
  
在mysite/urls.py文件里添加blog的url：

urlpatterns = patterns('',  
    # Examples:  
    # url(r'^$', 'mysite.views.home', name='home'),  
      
    url(r'^blog/', include('blog.urls')),  
    url(r'^admin/', include(admin.site.urls)),  
)

在mysite/blog/目录下创建urls.py文件：

from django.conf.urls import \*  
from blog.views import archive  
  
urlpatterns = patterns('',  
                      url(r'^$',archive),  
                      )

之所以在blog应用下面又创建urls.py文件，是为了降低耦合度。这样mysite/urls.py文件针对的是每个项目的url。  
  
再次启动服务（$ python manage.py runserver），访问blog应用（[http://127.0.0.1:8000/blog/](http://127.0.0.1:8000/blog/" \t "_blank)）。  
  
页面如下：  
  
  
**添加样式**                                                                                                      
  
创建基础模板  
  
在mysite/blog/templates目录里创建base.html的模板：

1. <html>
2. <style type="text/css">
3. body{color:#efd;background:#453;padding:0 5em;margin:0}
4. h1{padding:2em 1em;background:#675}
5. h2{color:#bf8;border-top:1px dotted #fff;margin-top:2em}
6. p{margin:1em 0}
7. </style>
9. <body>
10. <h1>虫师blog</h1>
11. <h3>大人不华，君子务实</h3>
12. {% block content %}
13. {% endblock %}
14. </body>
15. </html>

*复制代码*

修改archive.html模板，让它引用base.html模板和它的“content”块。

1. {% extends "base.html" %}
2. {% block content %}
3. {% for post in posts %}
4. <h2>{{  post.title }}</h2>
5. <p>{{ post.timestamp | date:"1,F jS"}}</p>
6. <p>{{ post.body }}</p>
7. {% endfor %}
8. {% endblock %}

*复制代码*

再次刷新博客页面：  
  
  
请系统的学习django web框架，然后在此基础上做更多的扩展，开发自己真正的blog 。