基于django_建立的博客

本文是根据Django 博客教程书写的

在这里我用的是pycharm这个软件进行开发

搭建平台

1.创建工程

```
>>django-admin startproject blog
```

2.启动服务

```
>>python manage.py runserver
```

(在要运行的工程目录下输入此命令)

运行这条命令后 在浏览器输入后台提示的网址就可以进入自己的工程了(网址 http://127.0.0.1:8000/)

成功的页面：

3.settings.py文件中配置语言

在创建的blog文件夹下找到settings.py

>*项目中的配置参数都在这个文件中*

找到 LANGUAGE_CODE，TIME_ZONE两个参数

更改为

`LANGUAGE_CODE = 'zh-hans'` 改为汉语

`TIME_ZONE = 'Asia/shanghai'` 改为中国时区

再运行项目就会显示中文

4.建立app

```
python manage.py startapp myblog
```

在之前的settings.py 文件中找到 INSTALLED_APPS 参数 把你创建的app加进去

For example:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myblog',//add your app name
]
```

建立数据库

1.数据库设计

如下：

article

id	标题	正文	标签	分类	时间
pk			fk	fk	

Category

id	分类
pk	

label

id	标签
pk	

comments

id	评论	时间	文章**id**
pk			fk

2.编写代码

在**mylog**文件夹下查找**models.py**文件

根据所设计的数据库插入如下代码：

```
from django.db import models

# Create your models here.
import datetime
from django.db import models
from django.utils.encoding import python_2_unicode_compatible


@python_2_unicode_compatible
class Article(models.Model):
    title = models.CharField(max_length=70)
    article_text = models.TextField()
    created_time = models.DateTimeField()
    modified_time = models.DateTimeField()
    excerpt = models.CharField(max_length=200, blank=True) # 文章摘要

    category = models.ForeignKey(Category)
    label = models.ForeignKey(Label, blank=True)

    def __str__(self):
        return self.article_text


@python_2_unicode_compatible
class Category(models.Model):
    name = models.CharField(max_length=100)

@python_2_unicode_compatible
class Label(models.Model):
    name = models.CharField(max_length=100)

@python_2_unicode_compatible
class Comments(models.Model):
    name = models.CharField(max_length=100)
    article = models.ForeignKey(Article)
```

3.数据库配置

————————————————————————

问题：
django连接的是mysqldb 而我用的是 pymysql 在迁移时出现错误

```
Configured: Error loading MySQLdb module: No module named 'MySQLdb'
```

解决方法：
在工程下找到-init-.py文件，添加如下代码：

```
import pymysql
pymysql.install_as_MySQLdb()
```

————————————————————————

打开settings.py文件,找到DATABASES 参数：

在这里链接的mysql数据库

```
DATABASES = {
    'default': {
        'ENGINE':'django.db.backends.mysql',
        'NAME': 'blog',
        'USER':'root',
        'PASSWORD':'root',
        'host': '127.0.0.1',
        'port': 3306,
    }
}
```

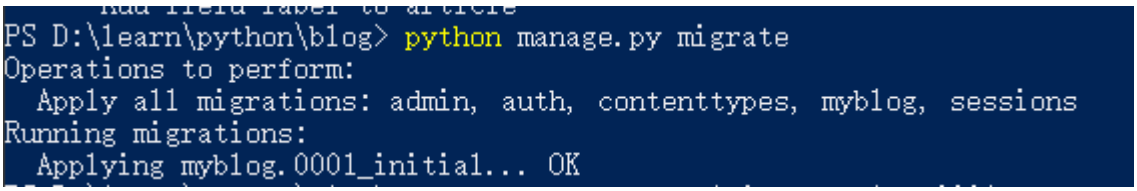
迁移数据库

`python manage.py migrate`

`python manage.py makemigrations myblog`

`python manage.py migrate`

结果：



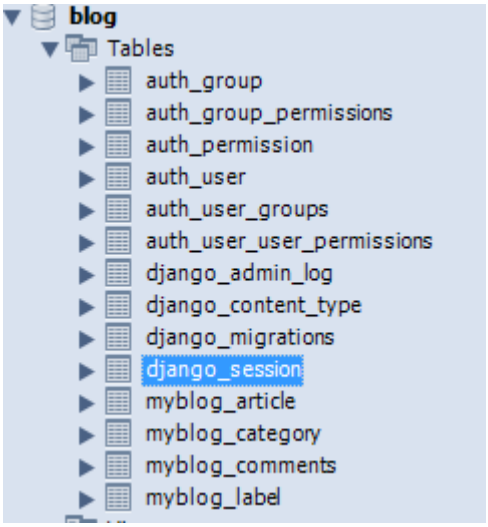
查看生成的数据库语句

`python manage.py sqlmigrate myblog 0001`

同步数据库

`.\manage.py syncdb`

运行过后在mysql中就可以看到同步的数据库了：



4.操作数据库

执行：

`python manage.py shell`

————————————————————————

在这里我用的ipython

安装： `pip install ipython`

ipython 功能很强大，这里用他在保存执行的代码

> 使用%logstart [logname [logmode]] 就可以开启IPython log，logname是日志保存的路径，logmode是日志模式，有四种模式可选：
>
> over: 如果存在那么覆盖之前的log
>
> backup: 这是默认的模式，会将之前的log备份,比如将log_name backup为log_name~
>
> append: 追加
>
> rotate: 回滚，会保持这样的文件名，比如logname.1~ logname.2~ logname.3~等等

logoff 结束日志备份

————————————————————————

向Category，label中插入数据

```
# add data to category and label
from blog.models import Category, Label, Article
from myblog.models import Category, Label, Article
c= Category(name='category test')
c.save()
t=Label(name='label test')
t.save()
```

向article中插入数据

- 首先要创建一个user

退出shell，执行：`python manage.py createsuperuser`

```
D:\learn\python\blog>python manage.py createsuperuser
Username (leave blank to use 'smurf'): Huilin
Email address: smurf28@163.com
Password:
Password (again):
This password is entirely numeric.
Password:
Password (again):
Superuser created successfully.
```

- 进入shell，添加article

```
  # add a article
  from myblog.models import Category, Label, Article
  from django.utils import timezone
  from django.contrib.auth.models import User
  user = User.objects.get(username='Huilin')
  c = Category.objects.get(name='category test')
  l=Label.objects.get(name='label test')
  a = Article(title='title test', article_text='test', created_time=timezone.now(),
modified_time=timezone.now(), category=c, author=user, label=l)
  a.save()
```

用**shell**进行数据库的增删改查

以label表为例：

增：l=Label(name='test')

　　l.save()

查：l=Label.objects.get(name='label test')

改：l.name='label test'

　　l.save()

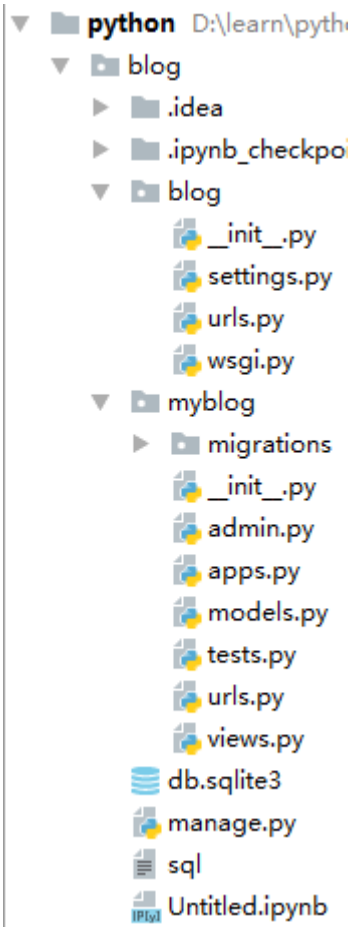
删：l.delete()

生成首页-test

1.绑定 URL 与视图函数

创建**urls.py**

在你的app下创建urls.py 文件，目录结构如下：



填写如下代码：

```
from django.conf.urls import url
from . import views

urlpatterns = [
    url(r'^$', views.index, name='index'), # r'^$' 正则表达式，匹配一个空字符
]
```

编写视图函数

找到你的app下的views.py文件，填写如下代码：

```
from django.http import HttpResponse


def index(request):
    return HttpResponse("welcome to my blog!")
```

配置项目url

在你的工程下的urls.py（和刚才创建的不是一个文件）添加如下代码：

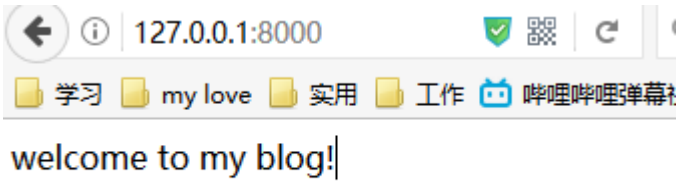
```
from django.conf.urls import url, include # add
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'', include('myblog.urls')), # add
]
```

运行程序

```
python manage.py runserver
```

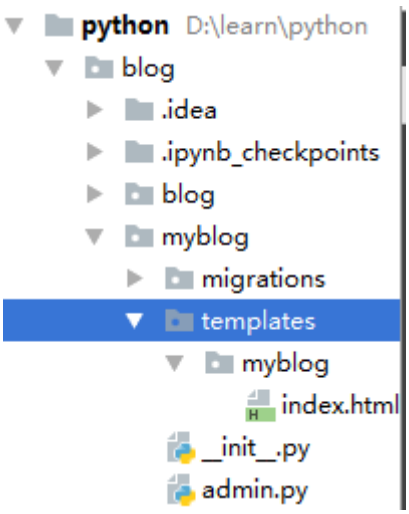
结果：



2.模板系统

创建myblog/templates/myblog/index.html文件

目录如下：



在index下写入：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{{ title }}</title>
</head>
<body>
<h1>{{ welcome }}</h1>
</body>
</html>
```

更改view.py

```
from django.http import HttpResponse
from django.shortcuts import render


def index(request):
    return render(request, 'myblog/index.html', context={
        'title': 'index_page',
        'welcome': 'welcome to my blog'
    })
    # return HttpResponse("welcome to my blog!")
```

运行工程就可以看到如下页面：

welcome to my blog

生成首页-true

1.更改view.py

index函数更改如下：

```
def index(request):
    article_list = Article.objects.all().order_by('-created_time')  # -created_time 逆序排序
    return render(request, 'myblog/index.html', context={'article_list': article_list})
```

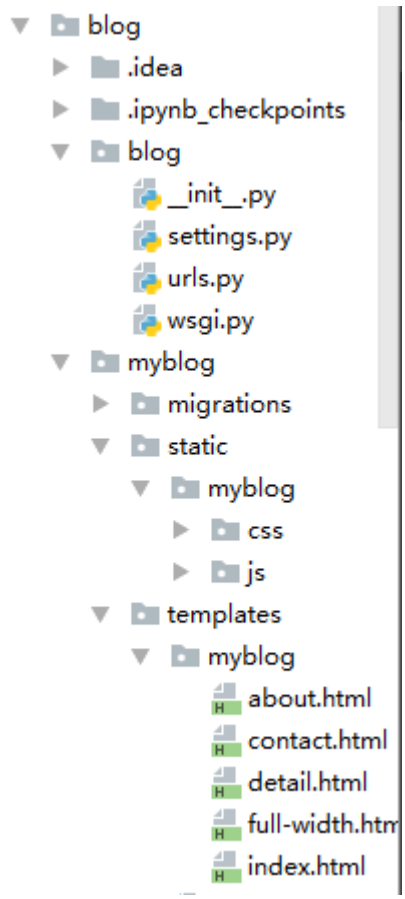
2.下载模板

> 地址：https://github.com/zmrenwu/django-blog-tutorial-templates

> 或用git执行：`git clone git@github.com:zmrenwu/django-blog-tutorial-templates.git`

3.导入前端页面

在myblog下新建文件夹templates\myblog和static\myblog，分别存放html和css、js页面，目录结构如下：



4.绝对路径更改

在index.html页面将引用的css和js文件路径改为相对路径。进行如下更改：

{% %}包裹模板标签

```
{% load static %} //在页面顶部load

<link rel="stylesheet" href="{% static 'myblog/css/bootstrap.min.css' %}">
```

load static后页面自动加载路径到static文件夹下面找文件。

其配置在setting.py中STATIC_URL字段，可以更改static的路径。

5.修改模板

在index.html中找到article标签：

```
...
<article class="post post-1">
  ...
</article>

<article class="post post-2">
  ...
</article>

<article class="post post-3">
  ...
</article>
...
```

改成如下：

```
...
{% for post in article_list %}//article_list是view文件中传递的值
<article class="post post-1">
         ...
</article>
    {% empty %}
        <div class="no-post">暂时还没有发布的文章！</div>
    {% endfor %}
...
```

header标签中更改如下：

```
<h1 class="entry-title">
    <a href="{{ post.get_absolute_url }}">{{ post.title }}</a>
</h1>
<div class="entry-meta">
    <span class="post-category"><a href="#">{{ post.category.name }}</a></span>
    <span class="post-date"><a href="#"><time class="entry-date" datetime="{{ post.created_time
}}">{{ post.created_time }}</time></a></span>
    <span class="post-author"><a href="#">{{ post.author }}</a></span>
    <span class="comments-link"><a href="#">4 评论</a></span>
    <span class="views-count"><a href="#">588 阅读</a></span>
</div>
```

文章摘要部分：

```
<div class="entry-content clearfix">
    <p>{{ post.excerpt }}</p>
    <div class="read-more cl-effect-14">
        <a href="{{ post.get_absolute_url }}" class="more-link">继续阅读 <span class="meta-
nav">→</span></a>
    </div>
</div>
```

使用django后台管理发布文章

创建Admin后台管理账户

命令：`python manage.py createsuperuser`

按照提示输入用户名，邮箱，密码

在myblog\admin.py中加入代码：

```
from django.contrib import admin
from .models import Article, Category, Label

admin.site.register(Article)
admin.site.register(Category)
admin.site.register(Label)
```

运行服务器访问地址： http://127.0.0.1:8000/admin/进行管理。

指定Admin后台

在admin.py中加入如下代码：

```
class PostAdmin(admin.ModelAdmin):
    list_display = ['title', 'created_time', 'modified_time', 'category', 'author']
admin.site.register(Article, PostAdmin)
```

可以更改文章列表项

同样编写*detail.html*页面

对应的url：`url(r'^post/(?P<pk>[0-9]+)/$', views.detail, name='detail'),`

在view中增加对应的函数detail

模板继承

index.html 文件和 detail.html 文件除了 main 标签包裹的部分不同外，其它地方都是相同的，把相同的地方提取出来作为模板放到base.html中。

1.建立base.html

在templates\目录下

把 index.html 的内容全部拷贝到 base.html 文件里，然后删掉 main 标签包裹的内容，替换成如下的内容。

```
...
<main class="col-md-8">
    {% block main %}//占位框
    {% endblock main %}
</main>
<aside class="col-md-4">
  {% block toc %}
  {% endblock toc %}
  ...
</aside>
...
```

在index.html中最顶部添加`{% extends 'base.html' %}`就可以继承base.html

index.html如下：

```
{% extends 'base.html' %}

{% block main %}
    {% for post in article_list %}
        <article class="post post-1">
          ...
        </article>
    {% empty %}
        <div class="no-post">暂时还没有发布的文章！</div>
    {% endfor %}
    <!-- 简单分页效果
    <div class="pagination-simple">
        <a href="#">上一页</a>
        <span class="current">第 6 页 / 共 11 页</span>
        <a href="#">下一页</a>
    </div>
    -->
    <div class="pagination">
      ...
    </div>
{% endblock main %}
```

detail.html更改为：

```
{% extends 'base.html' %}

{% block main %}
    <article class="post post-1">
      ...
    </article>
    <section class="comment-area">
      ...
    </section>
{% endblock main %}
{% block toc %}
    <div class="widget widget-content">
        <h3 class="widget-title">文章目录</h3>
        <ul>
            <li>
                <a href="#">教程特点</a>
            </li>
            <li>
                <a href="#">谁适合这个教程</a>
            </li>
            <li>
                <a href="#">在线预览</a>
            </li>
            <li>
                <a href="#">资源列表</a>
            </li>
            <li>
                <a href="#">获取帮助</a>
            </li>
        </ul>
    </div>
{% endblock toc %}
```