# 第十三章 Django博客开发-数据建模与样式设定

本节所讲内容：

13.1 开发流程介绍

13.2 博客开发需求分析与建模

13.3 博客样式加载，模板渲染

## 13.1 开发流程介绍

之前Django的学习过程当中已经把基本Django开发学完了，现在以Django 的博客项目完成一遍课程的回顾和总结。同时来一次完整开发的Django体验。

一个产品从研究到编码我们要经历以下的过程：

数据建模

概要设计

需求分析

项目发布

项目测试

项目编码

## 13.2 博客开发需求分析与建模

需求分析

本次项目完成的是一个博客的项目，博客主要目的是为了分享个人的技术，进行技术积累。

主要是发布文章日志。但是也需要有评论和互动。需要完成以下功能点：

1. 文章的发布、展示、修改、删除。
2. 文章评论
3. 读者互动
4. 图片管理

概要设计

我们对上面的四个需求进行详细的分析

1. 文章的发布、展示、修改、删除。

文章发布、修改和删除，我们可以通过Django的admin后台完成，当然也可以自己定义或者用三方的后台比如xadmin，这个可以作为版本迭代的一个方向。

但是在开始的开发过程当中，我们着重要做的就是展示功能，文章展示通常分为以下几个技术点：

1. 文章列表页

按类型划分

搜索

分页

1. 文章详情页

文章排版

1. 文章评论
2. 评论发布
3. 评论回复
4. 读者互动
5. 展示个人联系方式
6. 图片管理
7. 文章图片展示管理
8. 照片展示管理

完成分析之后，开始创建Django项目

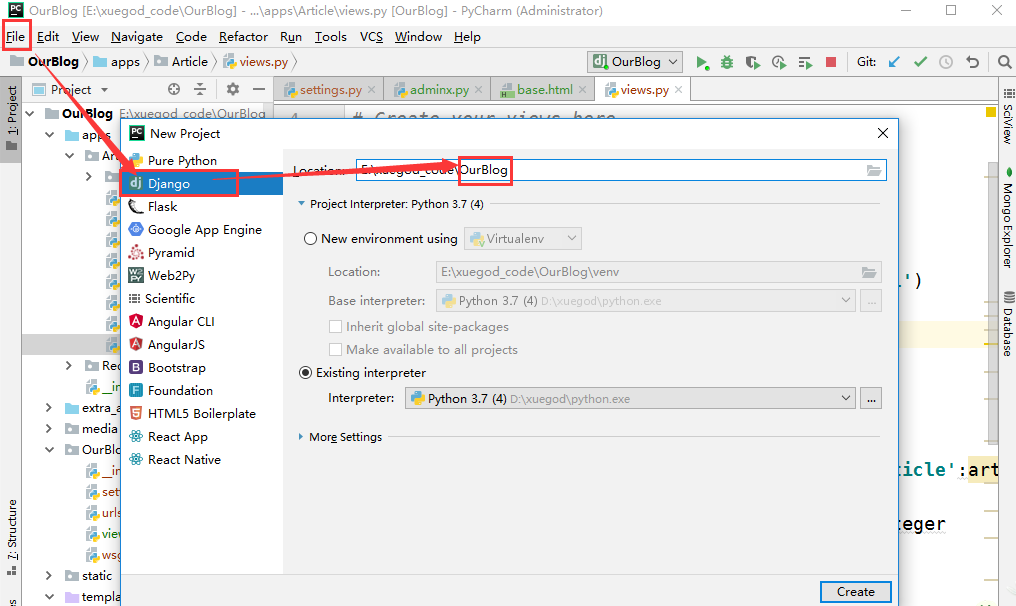
开发版本：

Python 3.7.2

Django 2.1.7

Pycharm

1、创建项目



2、切入项目，创建功能，app

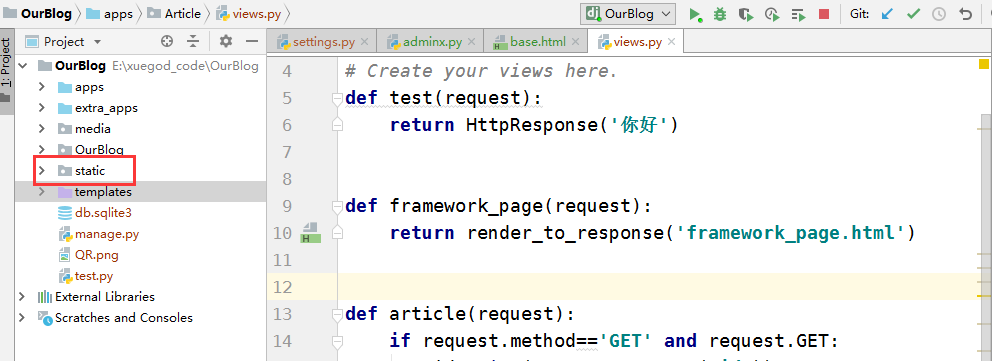
对于我们来说django app中的模板加载是必须要掌握的功能，其操作主要可总结成以下几个步骤。

1. 生成app文件
2. 在settings中的INSTALL\_APPS中添加（这个为了方便orm模板映射）
3. 在总urls.py中配置下url让程序能够通过url路由找到这个app
4. 配置app 中的views.py 指向已经生成的HTML模板渲染

在settings中的INSTALL\_APPS中添加（这个为了方便orm模板映射）



3、创建模板和静态文件存放的位置



4、安装依赖模块

pip install django-ckeditor==5.7.1

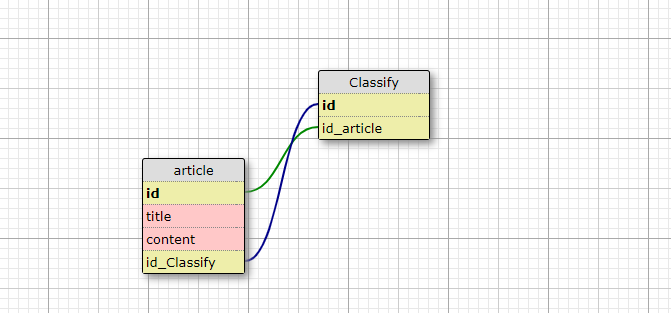
pip3 install pillow

注意：统一版本

数据建模

数据建模:数据建模就是用数据库描述上面分析的博客需求，然后形成对应的表和关系，数据建模是web开发至关总要的一个环节，一个优秀的模型是web开发成功的一大半。

在这里我介绍一个闪电哄哄的建表工具：<https://ondras.zarovi.cz/sql/demo/>



OurBlog/Article/models.py

from django.db import models  
*# Create your models here.*class Example(models.Model):  
 name = models.CharField(max\_length=32,verbose\_name='案例的名称')  
 type = models.CharField(max\_length=32,verbose\_name='案例的类型')  
  
 def \_\_str\_\_(self):  
 return self.name  
*#作者*class Author(models.Model):  
 gender\_choice = (  
 ('M','Male'),  
 ('F','Female'),  
 )  
 name = models.CharField(max\_length=32,verbose\_name='作者姓名')  
 age = models.IntegerField(verbose\_name='作者年龄',blank=True,null=True)  
 gender = models.CharField(max\_length=2,choices=gender\_choice,verbose\_name='作者姓名',blank=True,null=True)  
 email = models.EmailField(verbose\_name='作者邮箱',blank=True,null=True)  
 phone = models.CharField(max\_length=11,verbose\_name='作者电话',blank=True,null=True)  
 picture = models.ImageField(verbose\_name = "作者头像",upload\_to = "images/author/%Y%m",default='images/default.jpg')  
 *#在admin下展示这个表的中文名字* class Meta:  
 verbose\_name = '作者'  
 verbose\_name\_plural = verbose\_name  
 *# isdelete = models.IntegerField* def \_\_str\_\_(self):  
 return '作者:%s'%self.name  
  
  
*#种类表*class Classify(models.Model):  
 label = models.CharField(max\_length=32,verbose\_name='分类标签')  
 description = models.TextField(verbose\_name='分类描述')  
  
 class Meta:  
 verbose\_name='分类'  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return '标签:%s'%self.label  
from ckeditor\_uploader.fields import RichTextUploadingField  
*#文章*class Article(models.Model):  
 title = models.CharField(max\_length=32,verbose\_name='文章标题')  
 time = models.DateField(verbose\_name='文章发表日期')  
 description = RichTextUploadingField(verbose\_name='文章描述')  
 content = RichTextUploadingField(verbose\_name='文章内容')  
 *#外键关联（一对多关系……）* author = models.ForeignKey(Author,on\_delete=models.CASCADE)  
 *# isdelete =  
 #多对多关系* classify = models.ManyToManyField(Classify)  
  
 picture = models.ImageField(upload\_to='images/article/%Y%m',default='images/default.jpg',verbose\_name='文章图片')  
  
 class Meta:  
 verbose\_name = '文章'  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return '文章:%s'%self.title  
  
class Comment(models.Model):  
 content = RichTextUploadingField(verbose\_name = "评论内容")  
 content\_name = models.CharField(max\_length = 32,verbose\_name = "评论用户")  
 agree = models.IntegerField(verbose\_name = "评论点赞")  
 time = models.DateTimeField(verbose\_name = "评论时间")  
 article = models.ForeignKey(Article,on\_delete=models.CASCADE)  
  
 class Meta:  
 verbose\_name = '评论'  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return "[%s] 评论: %s"%(self.content\_name,self.content)  
  
class Picture(models.Model):  
 label = models.CharField(max\_length = 32, verbose\_name = "图片标签")  
 image = models.ImageField(verbose\_name = "图片链接", upload\_to = "images/picture/%Y%m",default='images/default.jpg')  
 description = RichTextUploadingField(verbose\_name = "图片描述")  
  
 classify = models.ForeignKey(Classify, verbose\_name = "图片分类",on\_delete=models.CASCADE)  
 comment = models.ForeignKey(Comment, verbose\_name = "图片评论",on\_delete=models.CASCADE)  
  
 class Meta:  
 verbose\_name = '图片'  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return "图片名称: %s" % self.label

完成模型的定义之后，接着进行项目的配置的修改

OurBlog/OurBlog/settings.py

**INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'Article',  
 'xadmin',  
 'crispy\_forms',  
 'ckeditor',  
 'ckeditor\_uploader',  
]  
  
MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
ROOT\_URLCONF = 'OurBlog.urls'  
  
TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [os.path.join(BASE\_DIR, 'templates')]  
 ,  
 'APP\_DIRS': True,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug',  
 'django.template.context\_processors.request',  
 'django.contrib.auth.context\_processors.auth',  
 'django.contrib.messages.context\_processors.messages',  
 ],  
 },  
 },  
]  
  
WSGI\_APPLICATION = 'OurBlog.wsgi.application'  
  
  
*# Database  
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases*DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.mysql',  
 'NAME': 'dbname',  
 'HOST':'127.0.0.1',  
 'PORT':'3306',  
 'USER':'root',  
 'PASSWORD':'123456'  
 }  
}  
  
  
*# Password validation  
# https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators*AUTH\_PASSWORD\_VALIDATORS = [  
 {  
 'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',  
 },  
]  
  
  
*# Internationalization  
# https://docs.djangoproject.com/en/2.1/topics/i18n/*LANGUAGE\_CODE = 'zh-hans'  
  
TIME\_ZONE = 'Asia/Shanghai'  
  
USE\_I18N = True  
  
USE\_L10N = True  
  
USE\_TZ = True  
  
  
*# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/2.1/howto/static-files/*STATIC\_URL = '/static/'  
STATICFILES\_DIRS =[  
 os.path.join(BASE\_DIR,'static')  
]  
*# STATIC\_ROOT = os.path.join(BASE\_DIR,'static/appstatic')  
#媒体卤路由用户将上传的数据存储到该目录下*MEDIA\_URL = '/media/'  
MEDIA\_ROOT = os.path.join(BASE\_DIR,'media')  
  
*#ckeditor上传路径，文件的上传路径，必须配合图形操作文件pillow。必要配置*CKEDITOR\_UPLOAD\_PATH = 'uploads/'  
CKEDITOR\_IMAGE\_GACKEND = 'pillow'**

然后进行ckeditor的路由指出。

OurBlog/OurBlog/urls.py

from django.urls import path,re\_path  
*# from django.conf.urls import url*from django.views.static import serve  
  
import xadmin  
from OurBlog.settings import MEDIA\_ROOT  
  
urlpatterns = [  
 path('xadmin/', xadmin.site.urls),  
 url(r'^article/',include('Article.urls',namespace='article')),  
 url(r'^ckeditor/', include('ckeditor\_uploader.urls')),  
 *# 处理图片显示的url,使用Django自带serve,传入参数告诉它去哪个路径找，我们有配置好的路径MEDIAROOT，对应上面的media* url(r'^media/(?P<path>.\*)', serve, {"document\_root": MEDIA\_ROOT }),  
]

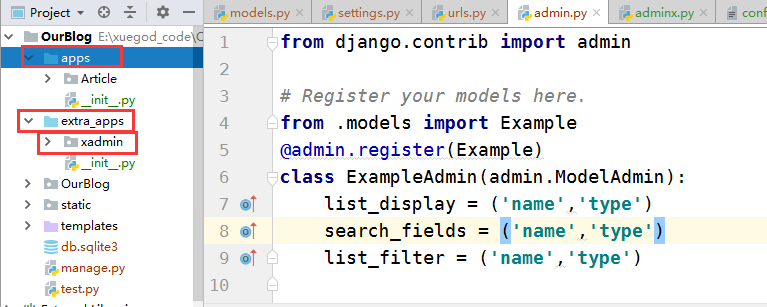
进行模型后台的安装

OurBlog/Article/adminx.py

1.配置apps

为了方便管理django生成的app我们写一个python Packpage 统一管理，如下图所示：

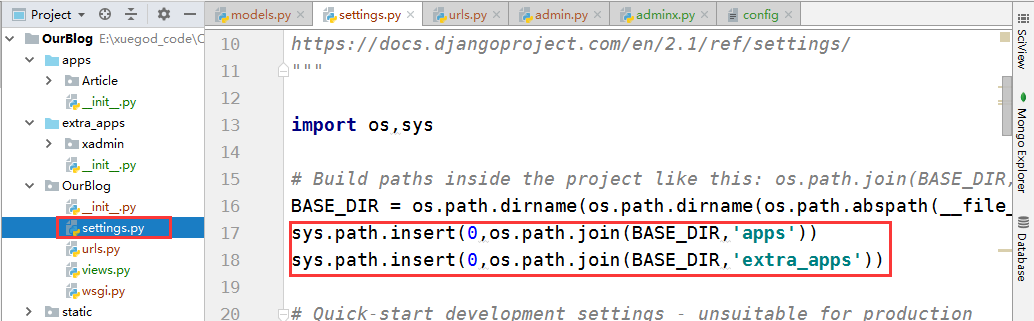
新建Python Package "extra\_apps",把源码xadmin文件夹放到extra\_apps文件夹下面，此时目录结构如下：



把apps 和extra\_apps全部键mark为Source Root（主要是让django项目能够在apps下寻找）

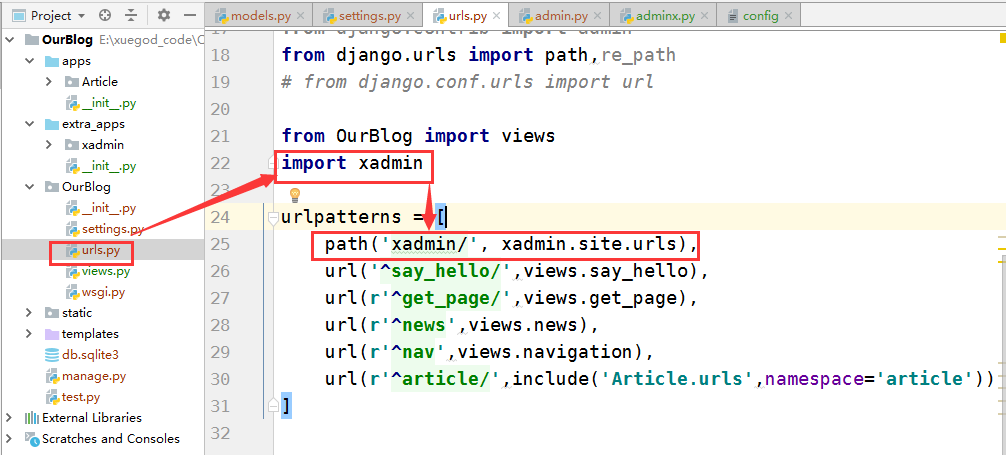
但是这种设置后我们的cmd运行也会报错所以说要在settings中加入配置，配置信息如下：

import os,sys  
*# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)*BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  
sys.path.insert(0,os.path.join(BASE\_DIR,'apps'))  
sys.path.insert(0,os.path.join(BASE\_DIR,'extra\_apps'))

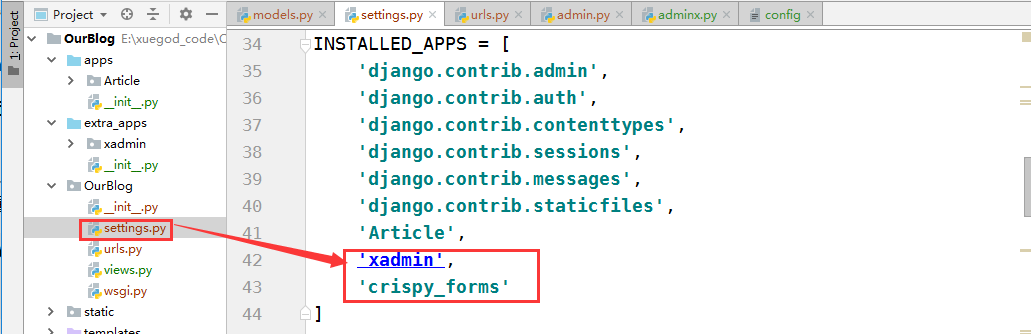


2.配置总路由（urls.py）

导入xadmin并修改总路由把admin修改成xadmin



3.注册app

把下面两个app注册到settings.py的INSTALLED\_APPS中

说明：这个crispy\_forms中主要是配合xadmin后台中表单提交的，功能更加强大

4.重新生成数据库

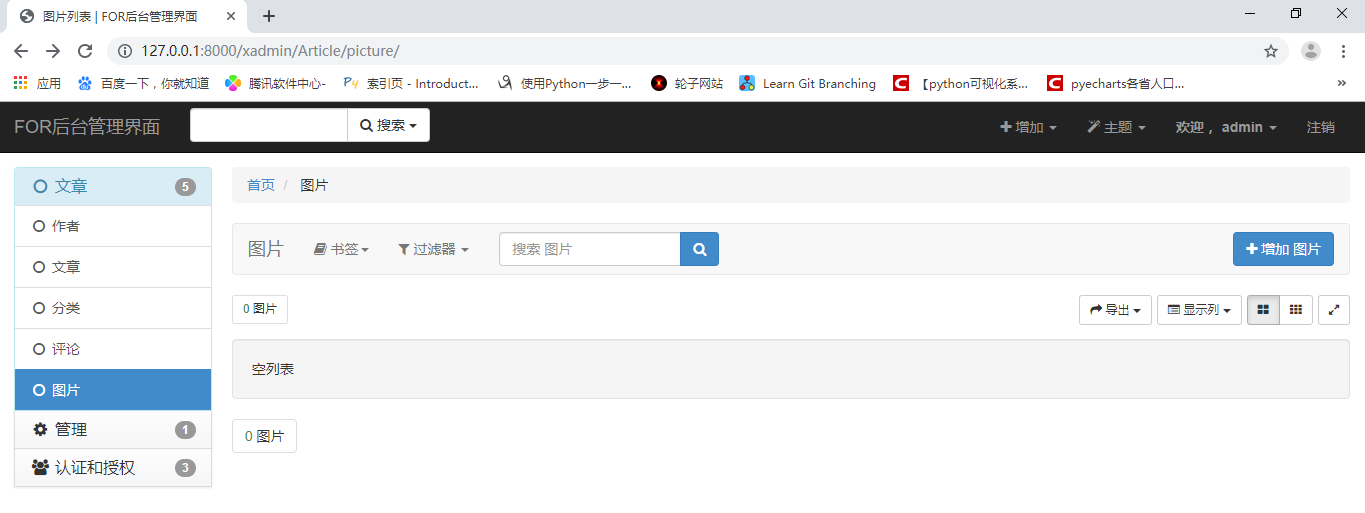
python manage.py makemigrations

python manage.py migrate

定制自己的xadmin后台样式

import xadmin  
from .models import Author,Article,Classify,Comment,Picture  
  
class AuthorAdmin(object):  
 list\_display = ['name','age','gender','email','phone']  
 search\_fields = ['name','age','gender','email','phone']  
 list\_filter = ['name','age','gender','email','phone']  
  
class ArticleAdmin(object):  
 list\_display = ['title','time','description','content','author','classify']  
 search\_fields = ['title','time','description','content','author','classify']  
 list\_filter = ['title','time','description','content','author','classify']  
  
class ClassifyAdmin(object):  
 list\_display = ['label','description']  
 search\_fields = ['label','description']  
 list\_filter = ['label','description']  
  
class CommentAdmin(object):  
 list\_display = ['content','content\_name','agree','time','article']  
 search\_fields = ['content','content\_name','agree','time','article']  
 list\_filter = ['content','content\_name','agree','time','article']  
  
class PictureAdmin(object):  
 list\_display = ['label','image','description','classify','comment']  
 search\_fields = ['label','image','description','classify','comment']  
 list\_filter = ['label','image','description','classify','comment']  
*# xadmin.site.register(Example,ExampleAdmin)*xadmin.site.register(Author,AuthorAdmin)  
xadmin.site.register(Article,ArticleAdmin)  
xadmin.site.register(Classify,ClassifyAdmin)  
xadmin.site.register(Comment,CommentAdmin)  
xadmin.site.register(Picture,PictureAdmin)  
  
from xadmin import views  
class BaseSetting(object):  
 enable\_themes = True  
 use\_bootswatch = True  
xadmin.site.register(views.BaseAdminView,BaseSetting)  
  
class GlobalSettings(object):  
 *# 修改title* site\_title = 'FOR后台管理界面'  
 *# 修改footer* site\_footer = 'FOR的公司'  
 *# 收起菜单* menu\_style = 'accordion'  
*# 将title和footer信息进行注册*xadmin.site.register(views.CommAdminView,GlobalSettings)

运行结果如下所示：



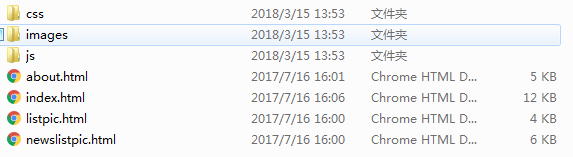
## 13.3 博客样式加载，模板渲染

上面的步骤完成之后，我们就可以进行页面的编写，加载，和渲染了，在工作当中，除非是个人项目，一般情况下，当进行Django开发的时候，前台的页面也就具备了。在这个项目当中，我们采用一个准备好的模板进行演示：

1. 模板渲染的第一步，是打开模板文件，进行页面特征的查看，对结构相同的页面构建base结构。

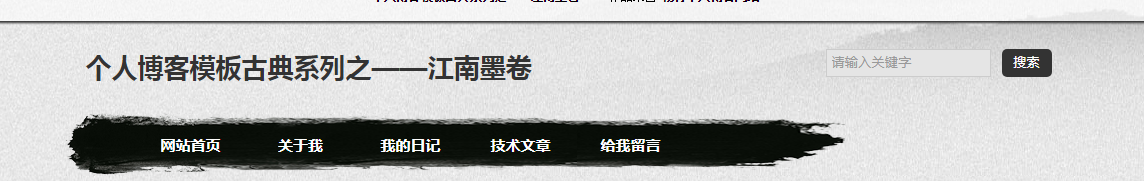
我们课上演示的是一款中国风格的博客模板，当然，大家也可以自己定义或者下载自己喜欢的模板。

模板文件如下：



比对四个页面的效果发现：

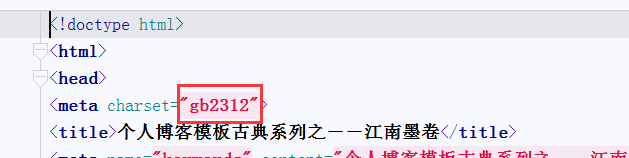
顶部:



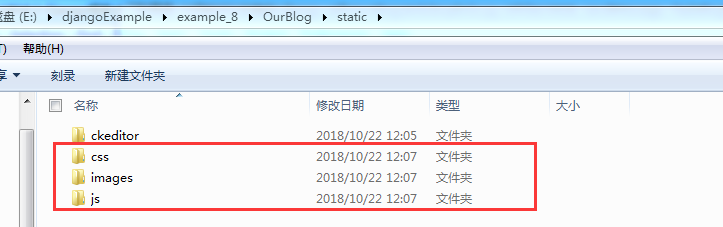
尾部：



效果一样。所以根据这个我们搭建自己的base页面。在这里要注意的是这套模板的编码是gb2312,我们需要进行修改。修改完成定制base页面



首先将静态文件迁入项目的static目录



然后根据前端样式分析，编写base.html（手生的同学可以复制一个页面过来，进行修改）

<!doctype html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <title>  
 {% block title %}  
 for 个人博客  
 {% endblock %}  
 </title>  
 <meta name="keywords" content="for 个人博客" />  
 <meta name="description" content="for 个人博客" />  
 <link href="/static/css/base.css" rel="stylesheet">  
 <link href="/static/css/index.css" rel="stylesheet">  
 *<!--[if lt IE 9]>  
<script src="/static/js/modernizr.js"></script>  
<![endif]-->* <script type="text/javascript" src="/static/js/jquery.js"></script>  
 {% block style %}  
 {% endblock %}  
</head>  
<body>  
  
<div id="wrapper">  
 <header>  
 <div class="headtop"></div>  
 <div class="contenttop">  
 <div class="logo f\_l">个人博客</div>  
 <div class="search f\_r">  
 <form action="/e/search/index.php" method="post" name="searchform" id="searchform">  
 <input name="keyboard" id="keyboard" class="input\_text" value="请输入关键字" style="color: rgb(153, 153, 153);" onfocus="if(value=='请输入关键字'){this.style.color='#000';value=''}" onblur="if(value==''){this.style.color='#999';value='请输入关键字'}" type="text">  
 <input name="show" value="title" type="hidden">  
 <input name="tempid" value="1" type="hidden">  
 <input name="tbname" value="news" type="hidden">  
 <input name="Submit" class="input\_submit" value="搜索" type="submit">  
 </form>  
 </div>  
 <div class="blank"></div>  
 <nav>  
 <div class="navigation">  
 <ul class="menu">  
 <li><a href="/index/">网站首页</a></li>  
 <ul>  
 <li><a href="bf/about.html">个人简介</a></li>  
 <li><a href="bf/listpic.html">个人相册</a></li>  
 </ul>  
 </li>  
 <li><a href="/Article/myBlog/">我的博客</a>  
 <ul>  
 <li><a href="bf/newslistpic.html">个人博客</a></li>  
 <li><a href="bf/newslistpic.html">技术博客</a></li>  
 </ul>  
 </li>  
 <li><a href="/Article/myPicture/">我的照片</a></li>  
 <li><a href="/aboutMe/">关于我</a>  
 <li><a href="#">给我留言</a> </li>  
 </ul>  
 </div>  
 </nav>  
 <SCRIPT type=text/javascript>  
 *// Navigation Menu* $(function() {  
 $(".menu ul").css({display: "none"}); *// Opera Fix* $(".menu li").hover(function(){  
 $(this).find('ul:first').css({visibility: "visible",display: "none"}).slideDown("normal");  
 },function(){  
 $(this).find('ul:first').css({visibility: "hidden"});  
 });  
 });  
 </SCRIPT>  
 </div>  
 </header>  
 {% block content %}  
  
 {% endblock %}  
 <footer>  
 <div class="footer">  
 <div class="f\_l">  
 <p>All Rights Reserved 版权所有：<a href="http://www.yangqq.com">for 个人博客</a> 备案号：蜀ICP备00000000号</p>  
 </div>  
 <div class="f\_r textr">  
 <p>let's say hello world</p>  
 </div>  
 </div>  
 </footer>  
</div>  
</body>  
</html>

1. 然后开始继承base模板开始定义指出页面

首先定义视图文件

from django.shortcuts import render  
  
def index(request):  
 return render(request,"index.html")  
  
def myArticle(request):  
 return render(request,"myArticle.html")  
  
def myPicture(request):  
 return render(request,"myPicture.html")  
  
def aboutMe(request):  
 return render(request,"aboutMe.html")  
  
def connectMe(request):  
 return render(request,"aboutMe.html")

1. 进行基本的样式修改，我们以aboutMe页面为例

{% extends "base.html" %}  
  
{% block title %}  
 关于我们  
{% endblock %}  
  
{% block style %}  
<link href="/static/css/main.css" rel="stylesheet">  
{% endblock %}  
  
{% block content %}  
 <div class="container">  
 <div class="con\_content">  
 <div class="about\_box">  
 <h2 class="nh1"><span>您现在的位置是：<a href="/" target="\_blank">网站首页</a>>><a href="#" target="\_blank">信息浏览</a></span><b>个人简介</b></h2>  
 <div class="f\_box">  
 <p class="a\_title">个人简介</p>  
 <p class="p\_title"></p>  
 *<!-- <p class="box\_p"><span>发布时间：2017-07-07 15:12:42</span><span>作者：唐孝文</span><span>来源：稽查支队</span><span>点击：111056</span></p>-->   
 <!-- 可用于内容模板 -->* </div>  
 <ul class="about\_content">  
 <p> 人生就是一个得与失的过程，而我却是一个幸运者，得到的永远比失去的多。生活的压力迫使我放弃了轻松的前台接待，放弃了体面的编辑，换来虽有些蓬头垢面的工作，但是我仍然很享受那些熬得只剩下黑眼圈的日子，因为我在学习使用Photoshop、Flash、Dreamweaver、ASP、PHP、JSP...中激发了兴趣，然后越走越远....</p>  
 <p><img src="/static/images/01.jpg"></p>  
 <p>“冥冥中该来则来，无处可逃”。 </p>  
 </ul>  
  
 </div>  
 </div>  
 <div class="blank"></div>  
 </div>  
 *<!-- container代码 结束 -->* {% endblock %}

1. 完成路由指出

import xadmin  
from OurBlog.settings import MEDIA\_ROOT  
from Article.views import \*  
urlpatterns = [  
 path('xadmin/', xadmin.site.urls),  
 url(r'^ckeditor/', include('ckeditor\_uploader.urls')),  
 *# 处理图片显示的url,使用Django自带serve,传入参数告诉它去哪个路径找，我们有配置好的路径MEDIAROOT，对应上面的media* url(r'^media/(?P<path>.\*)', serve, {"document\_root": MEDIA\_ROOT }),  
 url(r'^index/', index),  
 url(r'^myArticle/', myArticle),  
 url(r'^myPicture/', myPicture),  
 url(r'^aboutMe/', aboutMe),  
 url(r'^connectMe/', connectMe),  
]

1. 通过base页完成前端路由的修整,这样，我们的网站就可以点进选择了。

**<nav>  
 <div class="navigation">  
 <ul class="menu">  
 <li><a href="/index/">网站首页</a></li>  
 <ul>  
 <li><a href="/aboutMe/">个人简介</a></li>  
 <li><a href="/myPicture/">个人相册</a></li>  
 </ul>  
 </li>  
 <li><a href="/myArticle/">我的博客</a>  
 <ul>  
 <li><a href="/myArticle/">个人博客</a></li>  
 <li><a href="/myArticle/">技术博客</a></li>  
 </ul>  
 </li>  
 <li><a href="/myPicture/">我的照片</a></li>  
 <li><a href="/aboutMe/">关于我</a>  
 <li><a href="/aboutMe/">给我留言</a> </li>  
 </ul>  
 </div>  
</nav>**

总结：

13.1 开发流程介绍

13.2 博客开发需求分析与建模

13.3 博客样式加载，模板渲染