



链滴

Django 个人博客搭建 (4)- 博客主页设计和显示

作者: [zyk](#)

原文链接: <https://ld246.com/article/1567660818315>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

一. 前言

对于博客而言，首页的合理布局和展示非常关键，本文将利用 [Boundless-UI](#) 这套我自己开发的个人客模板结合具体代码，讲述如何编写主页视图和在主页中动态显示文章。关于这套主题模板的结构和能在我的 [github](#) 主页中有详细介绍。

文章总共分为四个部分，一为前端准备，介绍如何导入静态文件；二编写视图，讲述如何在视图中获文章；三文章动态显示，讲述如何在主页中利用模板语言动态地显示文章；四样例测试，运行项目进效果测试。

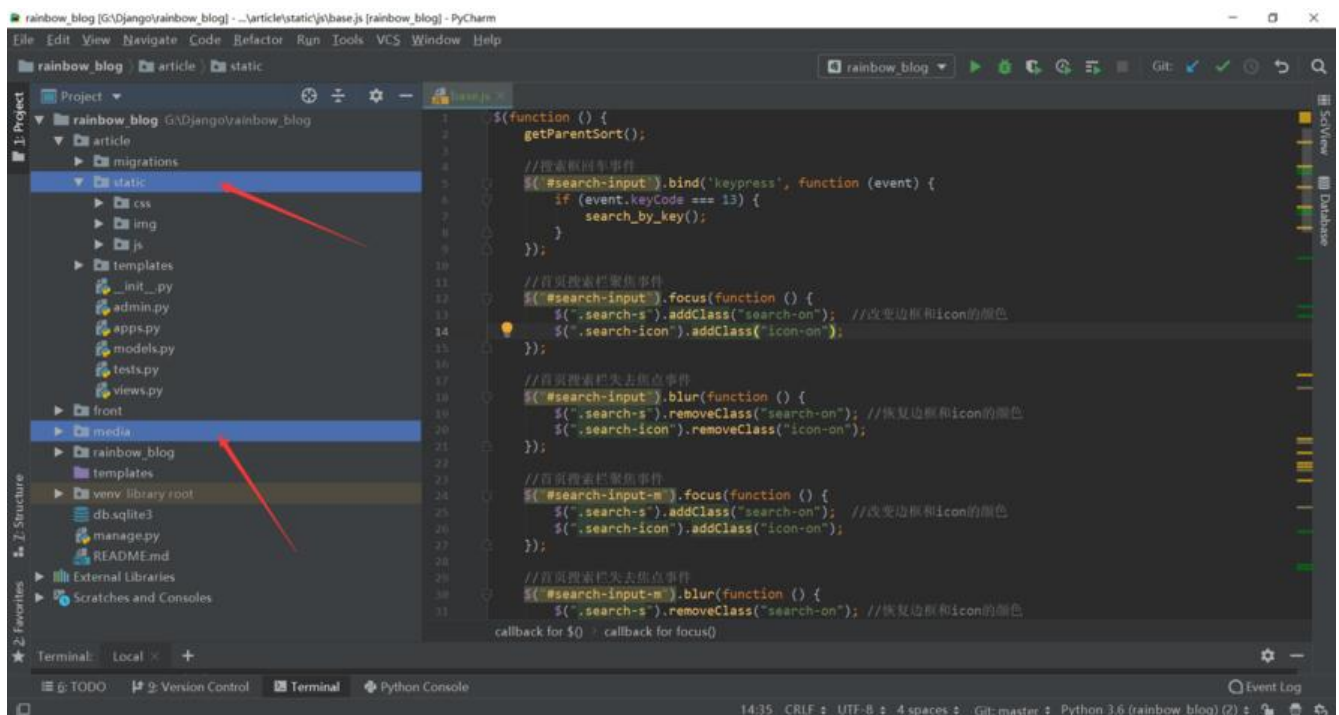
二. 前端准备

1. 下载模板

- 前往 [Boundless-UI](#)，在 github 主页下载项目的 zip 包，解压到指定目录，进入解压目录。

2. 初始化静态文件夹

- 将解压目录中的 [static](#)和[templates](#)文件夹复制到项目的[article](#)目录下。
- 将解压目录中的 [media](#)文件夹复制到项目根目录下，如下所示。



templates 放置的是 html 模板文件，static 中存放的是图片，css 和 js 等静态资源文件，media 为体文件。

三. 编写视图

1. 修改模型字段

- 由于模板的主页文章中还包含文章图片，因此需要在 [models.py](#)中为文章模型类添加新的图片字

。修改 **article/models.py**，为**article**模型类添加图片字段，为后台的图片上传做准备。如下所示。

```
# 文章模型类
class Article(models.Model):
    id = models.BigAutoField(primary_key=True) # 主键
    img = models.ImageField(max_length=255, null=True, verbose_name="文章图片") # 图片字段
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, verbose_name="作者") # 与自带的auth.user关联
    label = models.ManyToManyField('Label', verbose_name="标签") # Label和Article为多对多关系
    title = models.CharField(max_length=100, verbose_name="标题") # 标题
    content = models.TextField(max_length=100000, verbose_name="内容") # 内容
    summary = models.CharField(blank=True, max_length=200, verbose_name="摘要") # 摘要
    gmt_created = models.DateTimeField(blank=True, null=True, auto_now_add=True) # 创建时间
    gmt_modified = models.DateTimeField(blank=True, null=True, auto_now=True) # 修改时间

class Meta:
    verbose_name = '文章'
    verbose_name_plural = '文章'

def __str__(self):
    return self.title
```

● 由于调用了 django 自带的图片上传方法，所以需要配置图片上传的路径。进入项目根目录的 **settings.py**，在其中加入以下配置。

```
MEDIA_URL = '/media/' # URL访问路径
MEDIA_ROOT = os.path.join(BASE_DIR, 'media') # 本地路径
```

● 修改 **article/models.py**，为文章图片指定保存路径，为了防止重名，重写保存路径方法 **upload_to**，具体代码如下。

upload_to 的值表示图片将被保存在相对于之前配置的 **MEDIA_ROOT**中，若**upload_to = 'mmm'** 图片将被保存到 **media/mmm/**目录下。

```
import uuid
from django.conf import settings
from django.db import models

# 重写保存图片路径函数
def article_image_upload_to(instance, filename):
    return 'article/{uuid}/{filename}'.format(uuid=uuid.uuid4().hex, filename=filename)

# 文章模型类
class Article(models.Model):
    id = models.BigAutoField(primary_key=True) # 主键
    img = models.ImageField(upload_to=article_image_upload_to, max_length=255, null=True, verbose_name="文章图片") # 图片字段
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, verbose_name="作者") # 与自带的auth.user关联
    label = models.ManyToManyField('Label', verbose_name="标签") # Label和Article为多对多关系
    title = models.CharField(max_length=100, verbose_name="标题") # 标题
    content = models.TextField(max_length=100000, verbose_name="内容") # 内容
    summary = models.CharField(blank=True, max_length=200, verbose_name="摘要") # 摘要
```

```

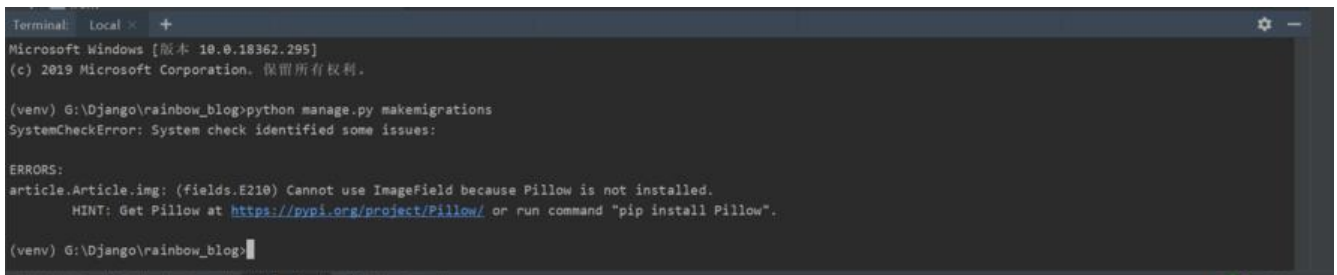
gmt_created = models.DateTimeField(blank=True, null=True, auto_now_add=True) # 创建
间
gmt_modified = models.DateTimeField(blank=True, null=True, auto_now=True) # 修改时间

class Meta:
    verbose_name = '文章'
    verbose_name_plural = '文章'

def __str__(self):
    return self.title

```

- 进入项目根目录，依次执行 `python manage.py makemigrations`和`python manage.py migrate` 令行更新数据表。
- 执行过程中出现了错误，提示 Pillow 库未安装，于是利用 `pip install pillow`命令安装好 pillow 之后，继续执行上述步骤中的两句命令行。



```

Terminal: Local > +
Microsoft Windows [版本 10.0.18362.295]
(c) 2019 Microsoft Corporation. 保留所有权利。

(venv) G:\Django\rainbow_blog>python manage.py makemigrations
SystemCheckError: System check identified some issues:

ERRORS:
article.Article.img: (fields.E210) Cannot use ImageField because Pillow is not installed.
    HINT: Get Pillow at https://pypi.org/project/Pillow/ or run command "pip install Pillow".

(venv) G:\Django\rainbow_blog>

```

pillow 库用于图像处理，报错的原因是在模型类中调用了`models.ImageField()`，调用此字段，django 就会利用 pillow 库自动保存上传的图片（我们只需指定保存图片的路径即可），而保存图片依赖于 pillow 库。

2. 编写主页视图

- 在 `article/views.py` 中编写主页视图函数，获取文章列表，将文章列表信息传入 `index.html` 模板中，具体代码如下所示。

```

from django.shortcuts import render
from article.models import Article

# 主页
def index(request):
    articles = Article.objects.all() # 获取所有文章
    return render(request, 'index.html', {'articles': articles}) # 返回至index.html，传入articles

```

- 将此视图函数注册到项目根目录的 `urls.py` 中，如下所示。

```

from django.contrib import admin
from django.urls import path, include
from article.views import index

urlpatterns = [
    path("", index, name='index'), # path第一个参数为空表示主页
    path('admin/', admin.site.urls),
    path('front/', include('front.urls')),
]

```

四. 文章动态显示

1. 导入静态文件

在 django 中 css , js , img 等静态文件资源，需要在模板文件中加入 `{% load static %}` 来加载。

- 编辑 `article/templates/index.html` 模板文件，在顶部加入 `{% load static %}`，改写 css 和 js 的路径，利用 `{% static '具体路径' %}` 导入静态资源，如下所示。

```
{% load static %} <!--别忘记在html顶部导入-->
<!--样式-->
<link rel="stylesheet" type="text/css" href="{% static 'css/global.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'css/index.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'css/myPagination.css' %}">
<link rel="shortcut icon" href="{% static 'img/favicon.ico' %}" />
<!--脚本-->
<script type="text/javascript" src="{% static 'js/jquery-3.4.1.min.js' %}"></script>
<script type="text/javascript" src="{% static 'js/global.js' %}"></script>
<script type="text/javascript" src="{% static 'js/myPagination.js' %}"></script>
<script type="text/javascript" src="{% static 'js/index.js' %}"></script>
```

2. 遍历文章列表

- 找到class 属性值为 `article-box` 的 div，保留一个 div 即可，利用模板中的 for 循环语句遍历从后获取的 `articles` 列表，并获取循环变量中的标题、摘要、作者、发布时间等值，如下所示。

小伙伴若是不大懂模板语句，可以看我的另外一篇博文 [Django模板介绍和基本变量语法](#)，在本文中详细论述。

```
<!--文章内容-->
<div id="article-holder" style="width: 100%; float: left">
  {% for article in articles %}
    <div class="article-box">
      <div class="ab-content">
        <div class="article-title"><a href="article-detail.html">{{ article.title }}</a><
div>
        <a href="article-detail.html" class="article-img-box">
          
          </a>
          <div class="article-detail-box c-666">
            {{ article.summary }}
          </div>
          <span class="article-tail-box"> <i class="fl"
x.svg' %}"></i>
            style="background-image: url('{% static 'img/read-ind
          <span class="read-number c-999 fl">0</span>
          <i class="fl" style="background-image: url('{% static 'img/comment-index
svg' %}"></i>
          <span class="comment-number c-999 fl">0</span>
          <span class="article-date c-999">{{ article.gmt_created | date:'Y-m-d' }}</
pan>
```

```

        <span class="article-author one-line-overflow c-999">{{ article.user.name
    }</span>

    </span>
</div>
</div>
{% endfor %}
</div>

```

五. 样例测试

- 运行项目，在浏览器中输入 <http://127.0.0.1/admin>，登录进入后台。给文章添加图片，丰富内容。
- 然后在浏览器中输入 <http://127.0.0.1> 回到主页，查看具体效果，若操作无误即可看到更新之后文章列表信息。但是我们发现图片无法访问。



- 修改项目根目录下的 **urls.py** (总路由)，加入 media 配置信息，如下所示。

```

from django.contrib import admin
from django.urls import path, include, re_path
from django.views.static import serve
from article.views import index
from rainbow_blog import settings

urlpatterns = [
    path("", index, name='index'), # path第一个参数为空表示主页
    path('admin/', admin.site.urls),
    path('front/', include('front.urls')),
    re_path(r'media/(?P<path>.*)$', serve, {'document_root': settings.MEDIA_ROOT}), # media
    置
]

```

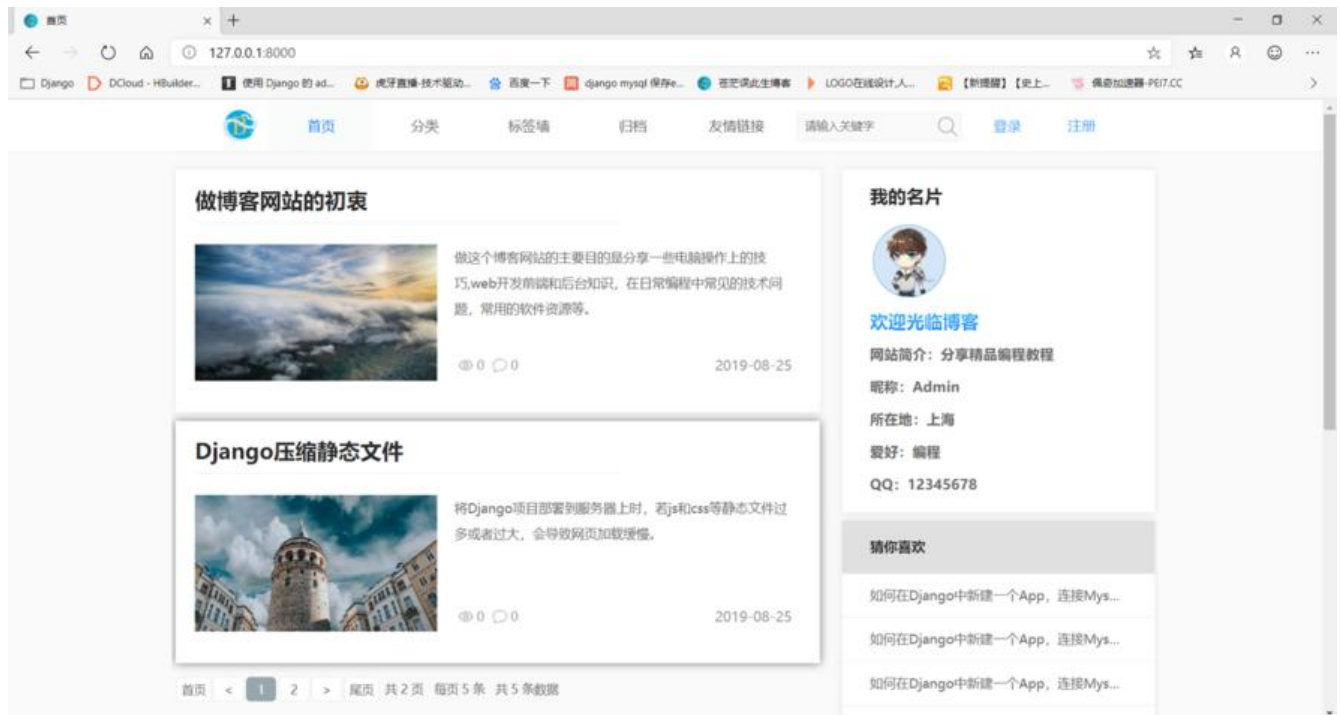
- 然后在 **settings.py** 中将 **MEDIA_URL** 加入其中，如下所示。

```

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',
                'django.template.context_processors.media', # 加入MEDIA_URL
            ],
        },
    },
]

```

- 重新运行项目，刷新主页，就能看到我们之前上传的文章图片了。



本文主要论述了Django个人博客主页视图的编写，模板的导入以及Django中静态文件的处理，如有不足之处可以在评论区留言。