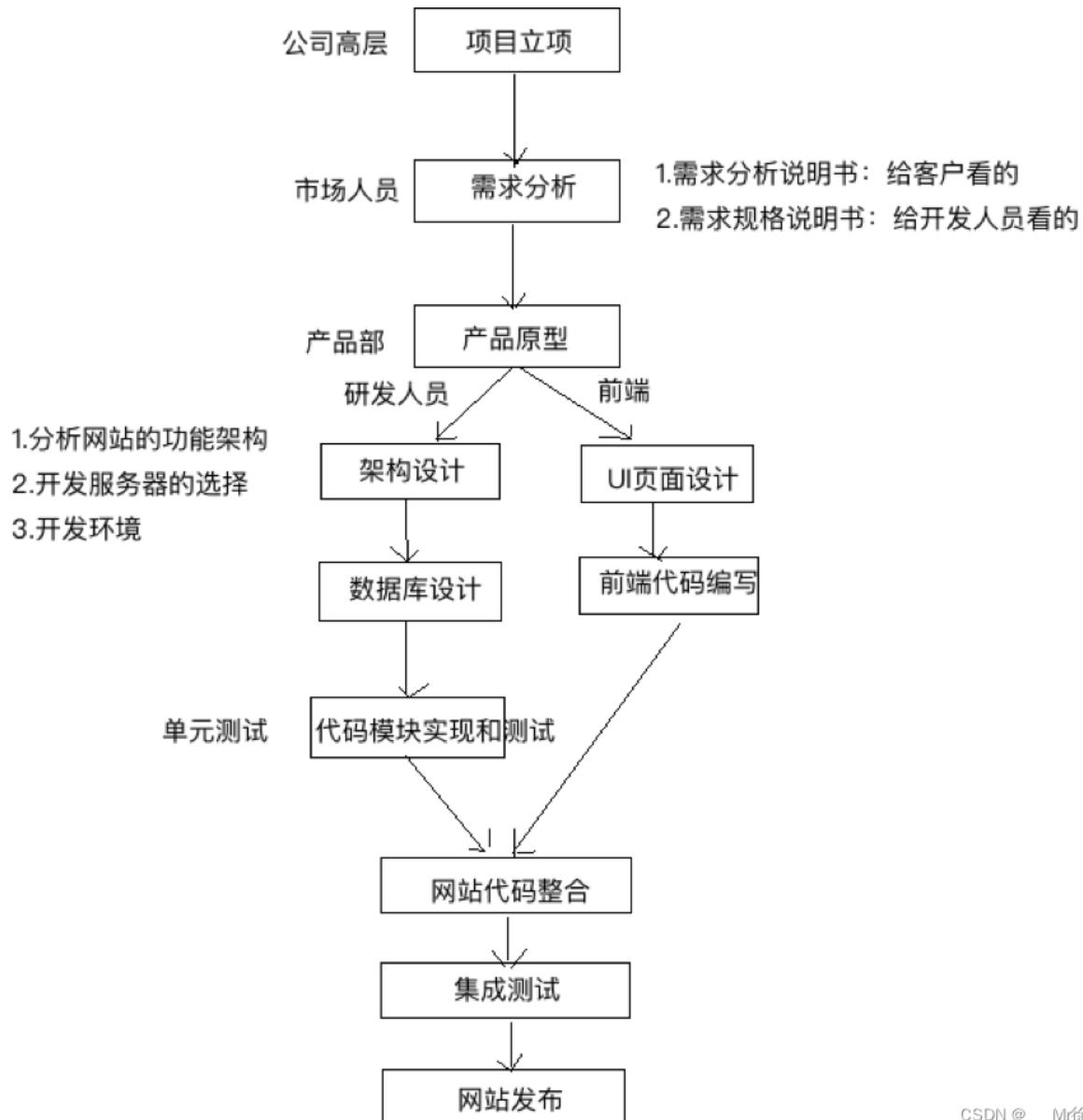


1. 项目准备

1.1 开发流程



- 构架设计

- 分析可能用到的技术点
- 前后端是否分离
- 前端使用哪些框架
- 后端使用哪些框架
- 选择什么数据库
- 如何实现缓存
- 是否搭建分布式服务
- 如何管理源代码

- 数据库设计

- 数据库表的设计至关重要
- 根据项目需求，设计合适的数据库表
- 数据库表在前期如果设计不合理，后期随需求增加会变得难以维护

- 集成测试
 - 在测试阶段要留意测试反馈平台的bug报告

1.2 需求分析

- 在需求分析阶段，我们可以借助产品原型图来分析。分析完后，前端按照产品原型图开发前端页面，后端开发响应业务处理。

1.2.1 注册, 登录

注册一个用户!

注册:

请输入您的邮箱

请输入密码

请输入您的昵称

注册

CSDN @__Mr徐



1.2.2 首页

统计信息

联系站长

文章总数: 18篇

网站运行: 1683天

请输入关键字

搜索

标签云

bb 0 另类 3 学 4 玩 3

热门推荐

1 2022年7月15日 85


2 2022年7月21日 11


3 2022年7月21日 9


4 2022年7月15日 7


5 2022年7月15日 8


6 2022年7月21日 5


7 2022年7月15日 3


8 2022年7月15日 2


9 2022年7月15日 0


10 2022年7月15日 0


11 2022年7月21日 1


12 2022年7月18日 0


13 2022年7月15日 0


14 2022年7月15日 0


15 2022年7月15日 0


16 2022年7月15日 0


17 2022年7月15日 0


18 2022年7月15日 0


19 2022年7月15日 0

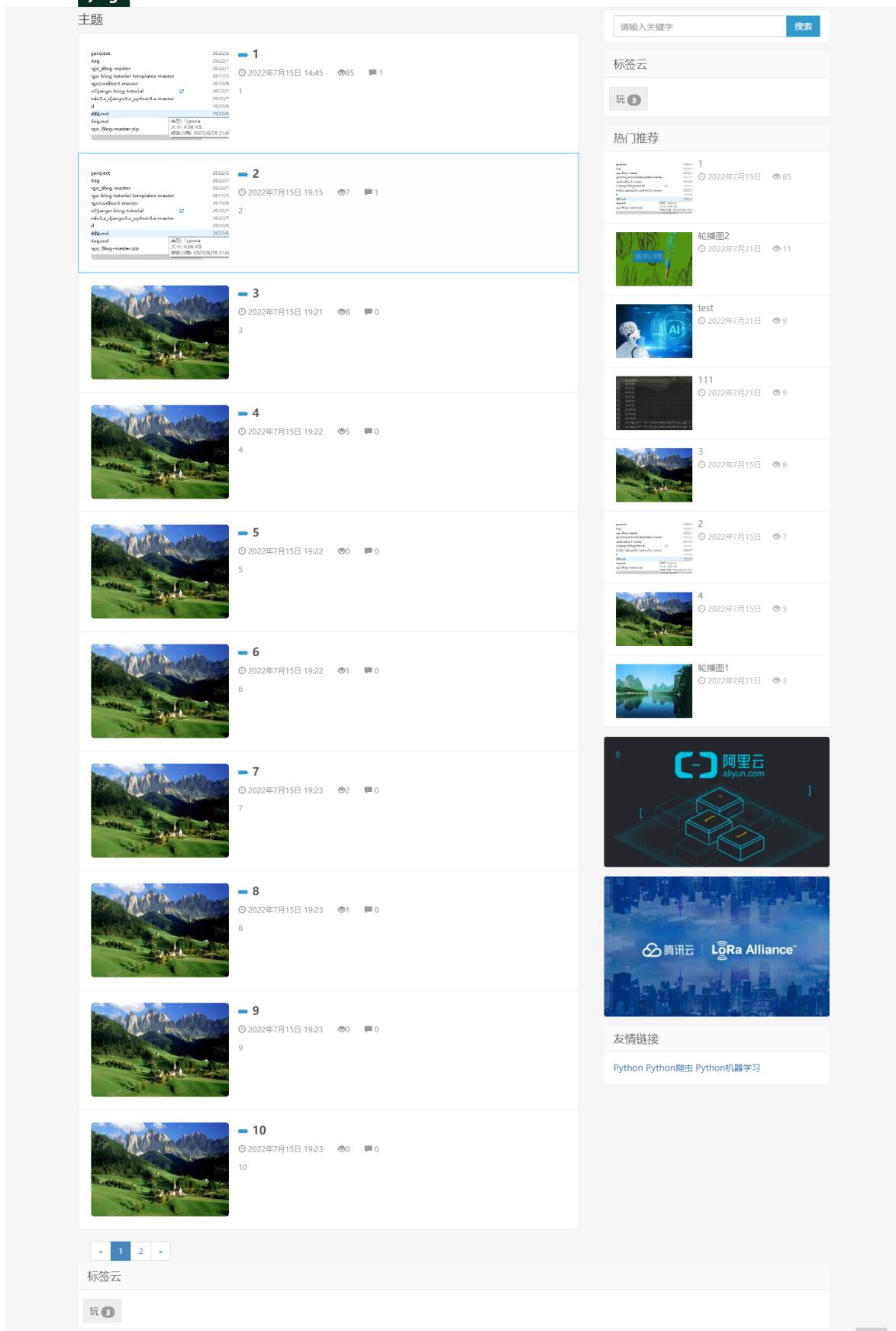

最新发布

游戏 学习 Django Flask 跳虫





1.2.3 列表页



1.2.4 详情页

Flask框架

© 2022年7月22日 14:07 | Flask | 1 | 0



Flask是一个使用 Python 编写的轻量级 Web 应用框架。其 WSGI 工具箱采用 Werkzeug，模板引擎则使用 Jinja2。Flask使用 BSD 授权。

Flask也被称为“microframework”，因为它使用简单的核心，用 extension 增加其他功能。Flask没有默认使用的数据库、窗体验证工具。[\[1\]](#)

Flask是一个轻量级的可定制框架，使用Python语言编写，较其他同类型框架更为灵活、轻便、安全且容易上手。它可以很好地结合MVC模式进行开发，开发人员分工合作，小型团队在短时间内就可以完成功能丰富的中小型网站或Web服务的实现。另外，Flask还有很强的定制性，用户可以根据自己的需求来添加相应的功能，在保持核心功能简单的同时实现功能的丰富与扩展。其强大的插件库可以让用户实现个性化的网站定制，开发出功能强大的网站。



请随手给个star，谢谢！

[打赏](#)

评论

您的评论或留言（必填）

[评论](#)

标签云

bb 0 | 另类 3 | 学 4 | 玩 3

请输入关键字

搜索

标签云

bb 0 | 另类 3 | 学 4 | 玩 3

热门推荐

1
app.py 4442 | 2022年7月15日 | 85
flask
WSGI
 Werkzeug
 Jinja2
 SQLAlchemy
 MySQL
 PostgreSQL
 Redis
 MongoDB
 Elasticsearch
 Redis Cluster
 RabbitMQ
 Celery
 SQLAlchemy
 MySQL
 PostgreSQL
 Redis
 MongoDB
 Elasticsearch
 Redis Cluster
 RabbitMQ
 Celery

2
轮播图2
© 2022年7月21日 | 11

3
test
© 2022年7月21日 | 9

4
111
© 2022年7月21日 | 9

5
3
© 2022年7月15日 | 8

6
2
© 2022年7月15日 | 7

7
轮播图1
© 2022年7月21日 | 3

8
阿里云
© 2022年7月15日 | 5

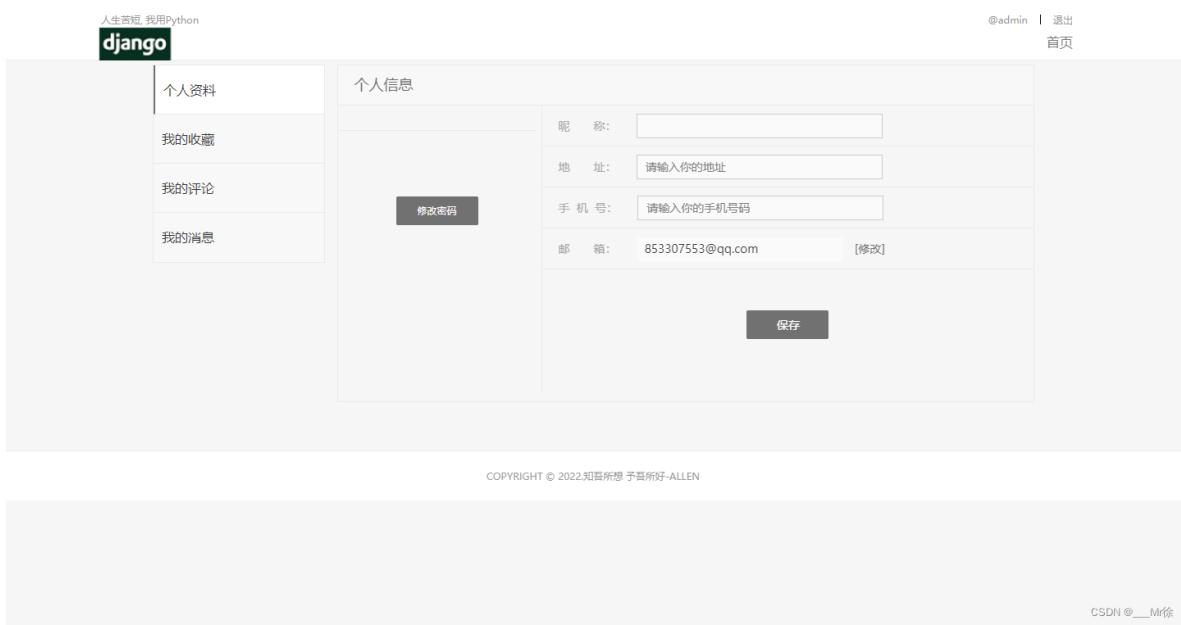
9
腾讯云 | LoRa Alliance™
© 2022年7月21日 | 3

友情链接

[Python](#) [Python爬虫](#) [Python机器学习](#)



1.2.5 用户中心

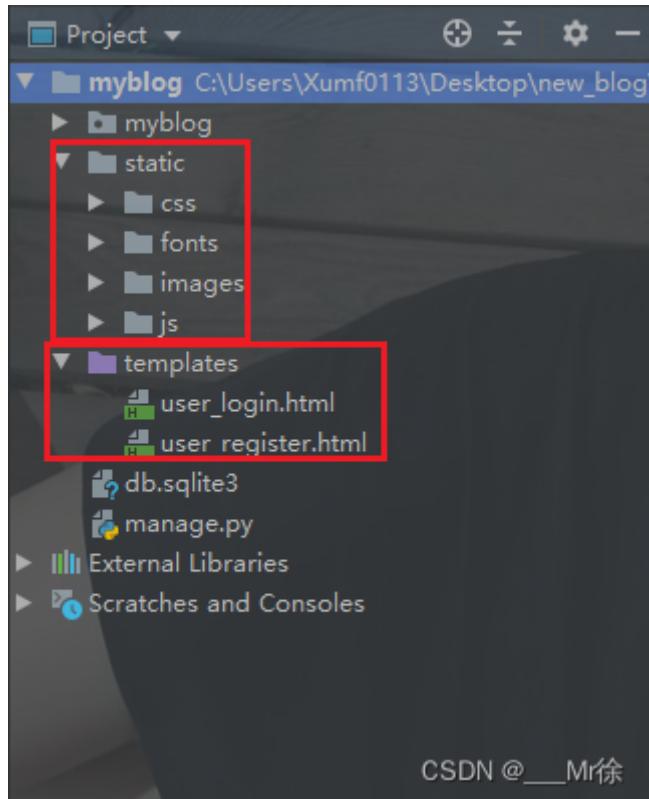


1.3 创建项目

- 通过命令安装 Django

```
pip install django==3.2.9
```

- 通过Pycharm创建Django项目，并且准备所有素材



- 创建MySQL数据库，并在Django当中配置好，然后使用Pycharm连接MySQL

```
# cmd窗口
create database myblog charset=utf8;
```

```
# settings.py文件

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'myblog', # 数据库名字
        'USER': 'root', # 数据库账号
        'PASSWORD': 'mysql', # 数据库密码
        'HOST': 'localhost', # 本地服务
        'PORT': 3306, # 端口号
    }
}
```

- 因为使用mysql, 所以要安装mysqlclient模块

```
pip install mysqlclient
```

- 调整Django的语言和时区

```
# settings.py文件

# LANGUAGE_CODE = 'en-us' 改为以下
LANGUAGE_CODE = 'zh-hans'

# TIME_ZONE = 'UTC' 改为以下
TIME_ZONE = 'Asia/Shanghai'
```

- 配置静态文件

```
# settings.py文件

# 静态文件在使用时的路径
STATIC_URL = '/static/'
# 静态文件存放的地方
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static')
]

# 媒体文件在使用时的路径
MEDIA_URL = '/media/'
# 媒体文件存放的地方
MEDIA_ROOT = os.path.join(BASE_DIR, 'static/media')
```

2.注册

- 整个博客在用户版块会用到哪些模型? 怎样去设计表?
 - 用户表
 - 用户验证
 - 如果只有两张表, 则根据两张表来进行设计
-
- Django提供了认证系统, 提供了用户模型类, 该模型类拥有很多方法, 所以我们在创建用户表的时候直接继承Django提供的用户模型
 - 如果使用了Django提供的用户类, 则得需要在settings.py当中配置

```
# settings.py文件

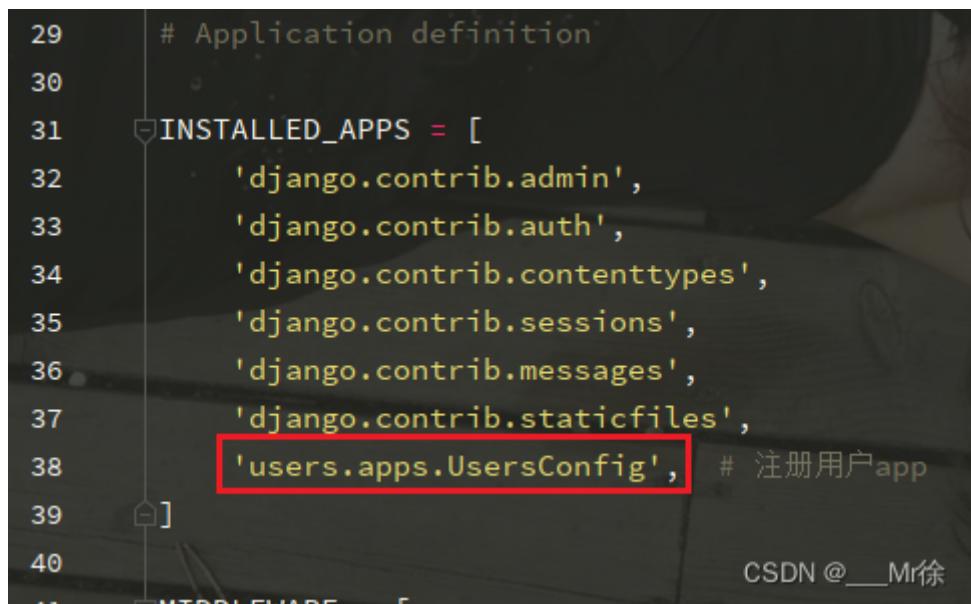
# 使用Django提供的用户类
AUTH_USER_MODEL = 'users.UserProfile'
```

2.1 创建用户app, 并配置

- 打开pycharm的Terminal使用命令创建app

```
python manage.py startapp users
```

- 注册user



```
29     # Application definition
30
31     INSTALLED_APPS = [
32         'django.contrib.admin',
33         'django.contrib.auth',
34         'django.contrib.contenttypes',
35         'django.contrib.sessions',
36         'django.contrib.messages',
37         'django.contrib.staticfiles',
38         'users.apps.UsersConfig', # 注册用户app
39     ]
40
```

CSDN @__Mr徐

- 创建子路由, 将并其配置到总路由中

```
# users\urls.py文件
from django.urls import path

urlpatterns = [
]
```

```
# myblog.py文件
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('users.urls')), # 导入users路由
]
```

2.2 创建模型类

- 在users的models中需要创建两个表，并生成迁移文件

```
# user\models.py文件

from django.db import models
from django.contrib.auth.models import AbstractUser
from datetime import datetime

class UserProfile(AbstractUser):
    """
    用户表
    1. username(Django提供字段): 用户名，唯一。存储用户的邮箱，用来校验该用户是否已存在
    2. email(Django提供字段): 邮箱，只用来存储邮箱
    3. nick_name: 用户名字，仅用来表示用户名，允许为空
    4. image: 用户头像(该博客中没暂没使用到)，允许为空
    5. add_time: 用户注册的时间，默认为创建用户时的时间
    6. is_start: 激活状态，默认未激活
    """

    nick_name = models.CharField(max_length=20, verbose_name='用户名', null=True, blank=True)
    image = models.ImageField(upload_to='user/%y/%m/%d', verbose_name='头像', max_length=200, null=True, blank=True)
    add_time = models.DateTimeField(default=datetime.now, verbose_name='添加时间')
    is_start = models.BooleanField(default=False, verbose_name='是否激活')

    def __str__(self):
        return self.username

    class Meta:
        db_table = 'UserProfile'
        verbose_name = '用户表'
        verbose_name_plural = verbose_name

class VerifyCodeEmail(models.Model):
    """
    验证表：用来对用户激活时做处理
    1. email: 用户邮箱
    """
```

```
2. code: 发送给邮箱的验证码
3. code_type: 邮箱的类型, 可以在注册, 重置等功能中
4. add_time: 发送验证码的时间, 默认发送的时间
"""

email = models.EmailField(max_length=30, verbose_name='用户邮箱')
code = models.CharField(max_length=128, verbose_name='验证码')
code_type = models.CharField(max_length=100,
                             choices=((('1', 'register'), ('2', 'reset'),
                                         ('3', 'changeemail'), ('4', 'sendpwd'))),
                             verbose_name='验证码类型')
add_time = models.DateTimeField(default=datetime.now, verbose_name='添加时间')

def __str__(self):
    return self.code

class Meta:
    db_table = 'VerifyCodeEmail'
    verbose_name = '验证码信息表'
    verbose_name_plural = verbose_name
```

- 注意: 使用了图片, 则需要安装pillow模块

```
pip install pillow
```

- Terminal中输入命令来生成迁移文件

```
python manage.py makemigrations
python manage.py migrate
```

```
Migrations for 'users':
  users\migrations\0001_initial.py
    - Create model VerifyCodeEmail
    - Create model UserProfile
```

```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, users
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002.Alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying users.0001_initial... OK
  Applying admin.0001_initial... OK
                                Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying sessions.0001_initial...                                     CSDN @__Mr徐
```

2.3 展示注册页面

- 根据路由, 视图, 来跳转到指定的模板页面

```
# usrs\usrs.py文件

from django.urls import path

from users import views

urlpatterns = [
    path('user_register', views.user_register, name='user_register'), # 用户注册路由
]
```

```
# users/views.py文件

from django.shortcuts import render

def user_register(request):
    """
    用户注册
    1. 用户想要注册，则直接进入到此页面
    2. 用户点击注册来注册账号
    可以根据不同的请求方法来做不同的请求事件
    """

    if request.method == 'GET':
        # 如果是get请求则直接进入到注册页面
        return render(request, 'user_register.html')
    else:
        # 如果是post请求，则代表是点击了注册进行提交数据
        return None
```

```
{# user_register.html文件#}

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <link href="/static/css/reglogin.css" rel='stylesheet' type='text/css' />
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
<div class="main">
    <div class="header">
        <h1>注册一个用户!</h1>
    </div>
    <p></p>
    <form action="#" method="post">
        <ul class="left-form">
            <h2>注册:</h2>
            {# 第一个表单 #}
            <li>
                <input type="text" placeholder="请输入您的邮箱" name="email"/>
                <div class="clear"></div>
            </li>
            {# 第二个表单 #}
            <li>
                <input type="password" placeholder="请输入密码" name="password"/>
                <div class="clear"></div>
            </li>
            {# 第三个表单 #}
            <li>
                <input type="text" placeholder="请输入您的昵称" name="nick_name"/>
                <div class="clear"></div>
            </li>
            <input type="submit" value="注册">
            <div class="clear"></div>
        </ul>
        <div class="clear"></div>

    </form>
</div>
</body>
</html>
```

- 用户注册的页面已经写好, 打开浏览器输入 `127.0.0.1:8000/user_register`, 即可访问



CSDN @__Mr徐

2.4 注册提交处理

- 此时注册页面已经能够展示出来, 当我们输入内容点击注册时, 怎样对数据进行校验, 同时怎样真正的将数据注册到咱们的博客里面去呢

2.4.1 表单

- 通过表单可以快速帮我们对数据进行校验以及错误时的反馈
- 在 `users` app 文件夹里面创建一个 `forms.py` 文件, 然后在里面创建表单

```
# users/forms.py文件

from django import forms

class RegisterForm(forms.Form):
    """
    用户注册的表单：当用户点击注册时，可以通过我们所写的表单字段来对数据进行校验
    email：用来校验用户提交的邮箱，EmailField(forms中定义的邮箱类型)，required(保存
    用户提交的数据不能为空)，error_messages(如果有错误，则根据字典中错误的key来提示对应信
    息)
    password：用来校验用户提交的密码，CharField(forms中定义的字符串类型)
    nick_name：用来校验用户提交的名字
    """

    email = forms.EmailField(required=True, error_messages={'required': '请
    填写邮箱', 'invalid': '请输入有效的邮箱'})
```

```
password = forms.CharField(required=True, error_messages={'required': '请填写密码'})
nick_name = forms.CharField(required=True, error_messages={'required': '请填写名字'})
```

2.4.2 校验数据

- 表单已经写完，可以在视图文件中通过表单对前端注册所提交的数据进行校验

```
from django.shortcuts import render

from .models import UserProfile
from .forms import RegisterForm


def user_register(request):
    """
    用户注册：当进入到此路由应该显示注册页面，但我们提交注册的数据时应该对数据进行处理
    所以可以根据不同的请求方法来做不同的请求事件
    """

    if request.method == 'GET':
        # 如果是get请求则直接进入到注册页面
        return render(request, 'user_register.html')
    else:
        # 如果是post请求，则代表是点击了注册进行提交数据
        register_form = RegisterForm(request.POST) # 生成表单进行验证
        if register_form.is_valid(): # 调用forms表单的方法来校验
            # 如果校验正确，获取前端提交的数据
            email = register_form.cleaned_data.get('email')
            pwd = register_form.cleaned_data.get('password')
            nick_name = register_form.cleaned_data.get('nick_name')

            # 在用户类当中查询当前用户是否存在，因为将email保存到username字段，所以通过这样来查询数据
            user = UserProfile.objects.filter(username=email)

            # 如果能够查询到数据，就代表该邮箱已经注册，查询不到代表该邮箱还没注册
            if user:
                # 如果用户存在，则回到注册页面，同时给于一定的提示，该字符串通过模板语言传递到模板当中去
                return render(request, 'user_register.html', {'msg': '用户名已存在'})
            else:
                # 如果用户不存在则保存信息
                user = UserProfile()
                user.username = email
                user.email = email
                user.nick_name = nick_name
                user.set_password(pwd) # 将密码加密后保存
                user.save()

                return render(request, 'wait_start.html')
        else:
            # 如果校验错误，就代表我们提交的数据不符合表单的要求，则回到注册页面，然后通过表单将错误信息展示到页面上
```

```
        return render(request, 'user_register.html', {'register_form': register_form})
```

2.4.3 模板调整

- 当我们点击注册的时候，根据不同的错误情况来展示不同的内容，通过模板语言来调整

```
{# user_register.html文件 #}

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <link href="/static/css/reglogin.css" rel='stylesheet' type='text/css'/>
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
<div class="main">
    <div class="header">
        <h1>注册一个用户！</h1>
    </div>
    <p></p>
    <form action="{% url 'user_register' %}" method="post">
        {% csrf_token %}
        <ul class="left-form">
            <h2>注册:</h2>
            {# 第一个表单 #-}
            <li>
                <input type="text" placeholder="请输入您的邮箱" name="email"/>
                <div class="clear"></div>
            </li>
            {# 通过表单.字段.errors来显示表单中给定的错误信息，这是个列表，所以得.0
        #}
            <div style="color: red">{{ register_form.email.errors.0 }}</div>
            {# 第二个表单 #-}
            <li>
                <input type="password" placeholder="请输入密码" name="password"/>
                <div class="clear"></div>
            </li>
            <div style="color: red">{{ register_form.password.errors.0 }}</div>
        </div>
            {# 第三个表单 #-}
            <li>
                <input type="text" placeholder="请输入您的昵称" name="nick_name"/>
                <div class="clear"></div>
            </li>
            <div style="color: red">{{ register_form.nick_name.errors.0 }}</div>
        </div>

        <input type="submit" value="注册">
        <div class="clear"></div>
    </ul>
    <div class="clear"></div>
```

```

</form>

{# 通过视图中传递的msg来展示错误提示 #}
<h3 style="width: 200px; height: 30px; margin: auto; color: red; text-align: center">{{ msg }}</h3>
</div>
</body>
</html>

```

2.4.4 发送邮件

- 登录博客需要激活此账号才行, 所以当我们注册账号时, 可以给该邮箱发送一封邮件, 只有用户点击该邮件后才能激活成功
- Django中内置了邮件发送功能, 被定义在django.core.mail模块中。发送邮件需要使用SMTP服务器, 常用的免费服务器有: 163, 126, QQ, 下面以163邮件为例。

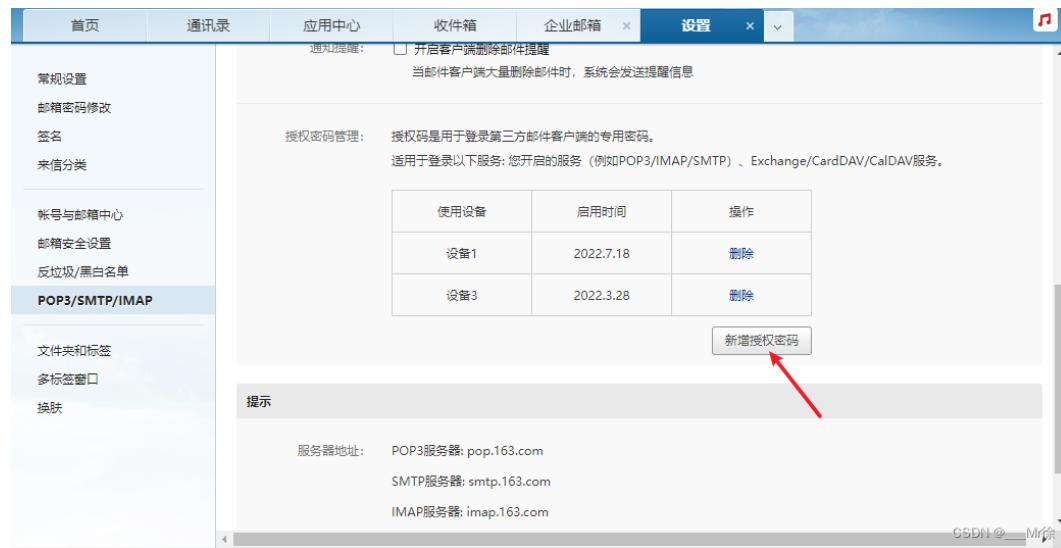
1. 注册163邮箱后登录并设置



2. 开启IMAP/SMTP服务, IMAP/SMTP服务



3. 新增授权密码



4. 记住授权密码(**一定不要忘了**), 此时该邮箱就已经能够用于在代码中发送邮箱了

- 接下来在Django中配置发送邮箱的代码

1. 在Django配置文件中，设置邮箱的配置信息

```
# 发送邮件
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.163.com'
EMAIL_PORT = 25
# 发送邮件的邮箱
EMAIL_HOST_USER = 'xumf0113@163.com'
# 在邮箱中设置的授权密码
EMAIL_HOST_PASSWORD = 'xxx'
# 收件人看到的发件人
EMAIL_FROM = '请激活<xumf0113@163.com>'
```

2. 在项目文件夹下创建一个utils文件夹, 该文件夹下放一些工具方法, 然后在该文件夹下创建一个 `send_verify_code.py` 文件, 用来写发送邮箱验证的代码

```
import time
import hashlib

from django.core.mail import send_mail
from django.conf import settings

from users.models import VerifyCodeEmail


def make_sign():
    """
    生成一个加密字符串
    时间戳+Django秘钥 通过md5加密
    """
    times = str(time.time())
    sign_str = times + settings.SECRET_KEY
    md5 = hashlib.md5()
```

```

        md5.update(sign_str.encode())
        sign = md5.hexdigest()
        return sign

def send_verify_email(to_email):
    """
    发送验证邮件
    为了保证唯一性，该邮件只能是某一用户的激活邮件，该邮件后面的参数通过加密来得到一个字符串
    """

    sign = make_sign()
    verify_url = '点击链接激活账号\nhttp://127.0.0.1:8000/user_active?sign=' + sign

    # 生成一个验证的对象，将发送的数据存储进去，在激活的时候去验证
    verify_code = VerifyCodeEmail()
    verify_code.email = to_email
    verify_code.code = sign
    verify_code.code_type = 1
    verify_code.save()

    subject = "邮箱验证"
    html_message = '<p>尊敬的用户您好！</p> \
                  '<p>感谢您使用此网站。</p> \
                  '<p>您的邮箱为: %s 。请点击此链接激活您的邮箱: </p> \
                  '<p><a href="%s">%s</a></p>' % (to_email, verify_url,
    verify_url)

    # 调用Django提供的方法来发送邮件
    send_mail(subject, "", settings.EMAIL_FROM, [to_email],
              html_message=html_message)

```

3. 紧接着在视图当中来调用该方法, 只要用户提交正确的数据, 就会发送邮件

```

# users/views.py文件

def user_register(request):
    ...省略其他代码
    else:
        # 如果用户不存在则保存信息
        user = UserProfile()
        user.username = email
        user.email = email
        user.nick_name = nick_name
        user.set_password(pwd)  # 将密码加密后保存
        user.save()

        # 发送邮件
        send_verify_email(email)
        return render(request, 'wait_start.html')
    ...省略其他代码

```



3. 登录

- 既然注册页面已经写完了, 那么可以拿着我们的账号来进行登录

3.1 展示登录页面

- 根据路由, 视图, 来跳转到指定的模板页面

```
# usrs\usrs.py文件

from django.urls import path

from users import views

urlpatterns = [
    path('user_register', views.user_register, name='user_register'), # 用户注册
    path('user_login', views.user_login, name='user_login'), # 用户登录
]
```

```
# users/views.py文件

def user_login(request):
    """
    用户登录
    1. 用户想要登录, 则直接进入到此页面
    2. 用户点击登录进入到主页
    可以根据不同的请求方法来做不同的请求事件
    """

    if request.method == 'GET':
        # 如果是get请求则直接进入到登录页面
        return render(request, 'user_login.html')
    else:
        # 如果是post请求, 则代表是点击了登录进行提交数据
        return None
```

```
{# usre_login.html文件 #}

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <link href="/static/css/reglogin.css" rel='stylesheet' type='text/css' />
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
<div class="main">
    <div class="header">
        <h1>登录!</h1>
    </div>
    <p></p>
    <form action="#" method="post">
        <ul class="left-form">
            <h2>登录:</h2>
            {# 输入的邮箱 #}
            <li>
                <input type="text" placeholder="请输入您的邮箱" name="email"/>
                <div class="clear"></div>
            </li>
            {# 输入的密码 #}
            <li>
                <input type="password" placeholder="请输入密码" name="password"/>
                <div class="clear"></div>
            </li>
            <input type="submit" value="登录">
            <div class="clear"></div>
        </ul>
        <div class="clear"></div>
    </form>
</div>
</body>
</html>
```

- 用户注册的页面已经写好, 打开浏览器输入 `127.0.0.1:8000/user_login`, 即可访问



3.2 登录提交处理

- 此时登录页面已经能够展示出来，当我们输入内容点击登录时，怎样对数据进行校验，同时实现真正的登录呢？

3.2.1 表单

- 打开user里面的forms表单，添加登录的表单

```
# users/forms.py文件

class UserLoginForm(forms.Form):
    ...
    用户登录的表单：当用户点击登录时，可以通过我们所写的表单字段来对数据进行校验
    email：用来校验用户提交的邮箱，EmailField(forms中定义的邮箱类型)，required(保存
    用户提交的数据不能为空)，error_messages(如果有错误，则根据字典中错误的key来提示对应信
    息)
    password：用来校验用户提交的密码，CharField(forms中定义的字符串类型)
    ...
    email = forms.EmailField(required=True, error_messages={'required': '请
    填写邮箱', 'invalid': '请输入有效的邮箱'})
    password = forms.CharField(required=True, error_messages={'required':
    '请填写密码'})
```

3.2.2 校验数据

- 表单已经写完，可以在视图文件中通过表单对前端登录所提交的数据进行校验

```

# users/views.py文件

def user_login(request):
    """
    用户登录
    1. 用户想要登录，则直接进入到此页面
    2. 用户点击登录进入到主页
    可以根据不同的请求方法来做不同的请求事件
    """

    if request.method == 'GET':
        # 如果是get请求，则直接进入到登录页面
        return render(request, 'user_login.html')
    else:
        # 如果是post请求，则是登录的提交
        login_form = UserLoginForm(request.POST) # 生成表单进行验证
        if login_form.is_valid(): # 调用forms表单的方法来校验
            # 如果校验正确，则拿前端发送的数据
            email = login_form.cleaned_data.get('email')
            pwd = login_form.cleaned_data.get('password')

            # 通过django的方法来校验账号和密码
            user = authenticate(username=email, password=pwd)
            if user:
                # 如果账号和密码正确先拿到该用户，查询该账号是否激活，激活则跳转主页，否则提示用户激活
                user_obj = UserProfile.objects.get(username=email)
                if user_obj.is_start: # 如果激活
                    login(request, user)
                    return redirect('/') # 重定向到主页
                else:
                    # 如果没激活
                    return render(request, 'user_login.html', {'msg': '请激活邮件'})
            else:
                # 如果用户账号和密码错误
                return render(request, 'user_login.html', {'msg': '用户名或密码错误'})
        else:
            # 如果用户提交的数据有问题
            return render(request, 'user_login.html', {'login_form': login_form})

```

3.2.3 模板调整

- 当我们点击登录的时候，根据不同的错误情况来展示不同的内容，通过模板语言来调整

```

{# user_login.html文件 #-}

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <link href="/static/css/reglogin.css" rel='stylesheet' type='text/css' />
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

```

```
<body>
<div class="main">
    <div class="header">
        <h1>登录!</h1>
    </div>
    <p></p>
    <form action="{% url 'user_login' %}" method="post">
        {% csrf_token %}
        <ul class="left-form">
            <h2>登录:</h2>
            {# 输入的邮箱 #}
            <li>
                <input type="text" placeholder="请输入您的邮箱" name="email"/>
                <div class="clear"></div>
            </li>
            <div style="color: red">{{ login_form.email.errors.0 }}</div>
            {# 输入的密码 #}
            <li>
                <input type="password" placeholder="请输入密码" name="password"/>
                <div class="clear"></div>
            </li>
            <div style="color: red">{{ login_form.password.errors.0 }}</div>

            <input type="submit" value="登录">
            <div class="clear"></div>
        </ul>
        <div class="clear"></div>
    </form>

    <h3 style="width: 200px; height: 30px; margin: auto; color: red; text-align: center">{{ msg }}</h3>
</div>
</body>
</html>
```

- 此时当们登录刚刚注册的用户时, 会提示需要激活邮件



CSDN @__Mr徐

3.3 用户激活

- 之前我们已经能够实现通过163邮箱发送邮件, 发送过去的邮件是用来激活的, 所以需要通过方法来实现
- 添加路由和视图

```
# user/urls.py文件

from django.urls import path

from users import views

urlpatterns = [
    path('user_register', views.user_register, name='user_register'), # 用户注册
    path('user_login', views.user_login, name='user_login'), # 用户登录
    path('user_active', views.user_active, name='user_active'), # 用户激活
]
```

```
# user/views.py文件
from django.urls import reverse
from .models import UserProfile, verifyCodeEmail

def user_active(request):
    """用户激活"""
    # 拿到激活链接上的秘钥
    sign = request.GET.get('sign')
```

```

# 如果有秘钥才进行校验
if sign:
    # 查看校验类里是否有数据
    verify_code = VerifyCodeEmail.objects.filter(code=sign)
    # 如果有，则将其用户激活
    if verify_code:
        # 拿到当前激活的邮箱号
        email = verify_code[0].email
        # 将用户激活
        usr_obj = UserProfile.objects.get(email=email)
        usr_obj.is_start = True
        usr_obj.save()
        # 删除当前验证类
        verify_code.delete()
        return redirect(reverse('user_login'))
return render(request, '404.html')

```

```

{# 在templates中创建404.html文件 #}
<!doctype html>
<html lang="zh-CN">
<head>
    <meta name="360-site-verification"
content="85326d9c1b0d512826605334e6eb1d5c">
    <meta charset="utf-8">
    <meta name="renderer" content="webkit">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="baidu_union_verify"
content="6c3c4420bcc5cb0d05563cc88180cd88">
    <title>很抱歉，没有相关内容</title>
    <meta name="keywords" content="Django博客">
    <link rel="stylesheet" type="text/css"
href="/static/css/bootstrap.min.css">
    <link rel="stylesheet" type="text/css" href="/static/css/nprogress.css">
    <link rel="stylesheet" type="text/css" href="/static/css/style.css">
    <link rel="stylesheet" type="text/css" href="/static/css/shang.css">
    <link rel="stylesheet" type="text/css" href="/static/css/font-
awesome.min.css">
    <link rel="apple-touch-icon-precomposed" href="/static/images/icon.png">
    <link rel="shortcut icon" href="/static/images/favicon.ico">
    <script src="/static/js/jquery-2.1.4.min.js"></script>
    <script src="/static/js/nprogress.js"></script>
    <script src="/static/js/jquery.lazyload.min.js"></script>
    <style type="text/css">
        .panel {
            padding: 80px 20px 0px;
            min-height: 400px;
            cursor: default;
        }

        .text-center {
            margin: 0 auto;
            text-align: center;
            border-radius: 10px;
            max-width: 900px;
            -moz-box-shadow: 0px 0px 5px rgba(0, 0, 0, .3);
            -webkit-box-shadow: 0px 0px 5px rgba(0, 0, 0, .3);
        }
    </style>

```

```

        box-shadow: 0px 0px 5px rgba(0, 0, 0, .1);
    }

    .float-left {
        float: left !important;
    }

    .float-right {
        float: right !important;
    }

    img {
        border: 0;
        vertical-align: bottom;
    }

    h2 {
        padding-top: 20px;
        font-size: 20px;
    }

    .padding-big {
        padding: 20px;
    }

    .alert {
        border-radius: 5px;
        padding: 15px;
        border: solid 1px #ddd;
        background-color: #f5f5f5;
    }

```

</style>

</head>

<body class="user-select">

<header class="header">

<nav class="navbar navbar-default" id="navbar">

<div class="container">

<div class="header-topbar hidden-xs link-border">

<ul class="site-nav topmenu">

!--标签云1-->

{% if request.user.is_authenticated %}

@{{ request.user.username }}

退出

{% else %}

登录

注册

{% endif %}

人生苦短，我用Python

</div>

<div class="navbar-header">

```
        <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#header-navbar" aria-expanded="false">
            <span class="sr-only"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
        </button>
        <h1 class="logo hvr-bounce-in">
            <a href="#" title="首页">
                
            </a>
        </h1>
    </div>
    {# 头部下面右 #}
    <div class="collapse navbar-collapse" id="header-navbar">
        <ul class="nav navbar-nav navbar-right">
            {# 所有的分类 #}
            <li>
                <a data-cont="python" title="Python" href="#">Python</a>
            </li>
            <li>
                <a data-cont="python" title="Java" href="#">Java</a>
            </li>
            <li>
                <a data-cont="python" title="C" href="#">C</a>
            </li>
            <li>
                <a data-cont="python" title="C++" href="#">C++</a>
            </li>
        </ul>
    </div>
</div>
</nav>
</header>
<section class="container">
    <div class="panel">
        <div class="text-center">
            <h2>
                <strong>很抱歉，没有相关内容 o(╥﹏╥)o</strong>
            </h2>
            <div class="padding-big"><a href="#" class="btn btn-primary">返回首页</a>
            </div>
        </div>
    </div>
</section>
<footer class="footer">
    <div class="container">
        <p>
        <p>
            Copyright &copy; 2022.知吾所想 予吾所好-Alen
        </p>
    </div>
    <div id="gotop"><a class="gotop"></a></div>
</footer>
<script src="/static/js/bootstrap.min.js"></script>
```

```

<script src="/static/js/jquery.ias.js"></script>
<script src="/static/js/scripts.js"></script>
<script>
    var _hmt = _hmt || [];
    (function () {
        var hm = document.createElement("script");
        hm.src = "https://hm.baidu.com/hm.js?e8ae61fbc1aa8b44823aae4cd23690b0";
        var s = document.getElementsByTagName("script")[0];
        s.parentNode.insertBefore(hm, s);
    })();
</script>

</body>
</html>

```

4 退出功能

- 用户已经能够登录了，但是我们可以测试多个账号，所以需要能够实现退出，再登录其他账号

4.1 添加路由

```

# users\urls.py文件

from django.urls import path

from users import views

urlpatterns = [
    path('user_register', views.user_register, name='user_register'), # 用户注册
    path('user_login', views.user_login, name='user_login'), # 用户登录
    path('user_active', views.user_active, name='user_active'), # 用户激活
    path('user_logout', views.user_logout, name='user_logout'), # 用户退出
]

```

4.2 编写视图

```

# users\views.py文件

from django.contrib.auth import authenticate, login, logout

def user_logout(request):
    """用户退出"""
    logout(request)
    return redirect('/')

```

5. 静态文件

5.1 静态文件

- 静态文件是指项目中的CSS、js、图片
- 为了安全可以通过配置项隐藏真实图片路径，在模板中写成固定路径，后期维护太麻烦，可以使用static标签，根据配置项生成静态文件路径
- 语法

```
{% load static %}  

```

- 调整templates文件夹下面所有html文件里面的静态文件

6. 首页

- 根据首页图，能够看到首页当中展示的都是文章相关的信息，所以此时需要思考【文章】模型应该怎样来构建

- 文章表
- 分类表
- 标签表



6.1 创建文章app，并展示

- 打开pycharm的Terminal使用命令创建app

```
python manage.py startapp articles
```

- 注册 articles

```

29     # Application definition
30
31     INSTALLED_APPS = [
32         'django.contrib.admin',
33         'django.contrib.auth',
34         'django.contrib.contenttypes',
35         'django.contrib.sessions',
36         'django.contrib.messages',
37         'django.contrib.staticfiles',
38         'users.apps.UsersConfig', # 注册用户app
39         'articles.apps.ArticlesConfig', # 注册文章app
40     ]

```

CSDN @__Mr徐

- 创建子路由, 将并其配置到总路由中

```

# articles\urls.py文件

from django.urls import path

from articles import views

urlpatterns = [
    path('', views.index, name='index'), # 主页
]

# myblog.py文件
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('users.urls')), # 导入users路由
    path('', include('articles.urls')), # 导入articles路由
]

```

- 创建视图并展示首页样式

```

# articles\views.py文件

def index(request):
    """首页展示"""
    return render(request, 'index.html')

```

- 创建模型文件(在templates下面创建index.html文件)

```
{# templates\index.html #}
```

```
{% load static %}
<!doctype html>
<html lang="zh-CN">
<head>
    <meta name="360-site-verification" content="85326d9c1b0d512826605334e6eb1d5c">
    <meta charset="utf-8">
    <meta name="renderer" content="webkit">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="baidu_union_verify" content="6c3c4420bcc5cb0d05563cc88180cd88">
    <title>Django博客</title>
    <meta name="keywords" content="Django博客">
    <link rel="stylesheet" type="text/css" href="{% static 'css/bootstrap.min.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/nprogress.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/shang.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/fontawesome.min.css' %}">
    <link rel="apple-touch-icon-precomposed" href="{% static 'images/icon.png' %}">
    <link rel="shortcut icon" href="{% static 'images/favicon.ico' %}">
    <script src="{% static 'js/jquery-2.1.4.min.js' %}"></script>
    <script src="{% static 'js/nprogress.js' %}"></script>
    <script src="{% static 'js/jquery.lazyload.min.js' %}"></script>
</head>
<body class="user-select">
<header class="header">
    <nav class="navbar navbar-default" id="navbar">
        <div class="container">
            {# 头部最上层 #}
            <div class="header-topbar hidden-xs link-border">
                <ul class="site-nav topmenu">
                    {# 如果用户是登录则显示名字#}
                    {% if request.user.is_authenticated %}
                        <li><a href="#">@{{ request.user.username }}</a>
                    </li>
                    <li><a href="#" style="border-left: 1px solid black">退出</a></li>
                    {# 如果未登录，则显示登录或注册 #}
                    {% else %}
                        <li><a href="{% url 'user_login' %}">登录</a></li>
                        <li><a href="{% url 'user_register' %}">
                            注册</a></li>
                        {% endif %}
                    </ul>
                    人生苦短，我用Python
                </div>
            {# 头部下面左 #}
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#header-navbar" aria-expanded="false">
```

```
        <span class="sr-only"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
    </button>
    <h1 class="logo hvr-bounce-in">
        <a href="{% url 'index' %}" title="首页">
            
        </a>
    </h1>
</div>
{# 头部下面右 #}
<div class="collapse navbar-collapse" id="header-navbar">
    <ul class="nav navbar-nav navbar-right">
        <li>
            <a data-cont="首页" title="首页" href="{% url 'index'
%}">首页
            </a>
        </li>
        <li>
            <a data-cont="Python" title="Python"
href="#">Python</a>
            </li>
        <li>
            <a data-cont="Java" title="Java" href="#">Java</a>
            </li>
        <li>
            <a data-cont="Java" title="Java" href="#">Java</a>
            </li>
        <li>
            <a data-cont="Python" title="Python"
href="#">Python</a>
            </li>
        <li>
            <a data-cont="Python" title="Python"
href="#">Python</a>
            </li>
        </ul>
    </div>
</nav>
</header>
{# 中间区域， 轮播图+文章+标签等所有 #}
<section class="container">
    {# 左边部分 #}
    <div class="content-wrap">
        {# 中间文章的区域 #}
        <div class="content">
            {# 轮播图 #}
            <div id="focusslide" class="carousel slide" data-
ride="carousel">
                <ol class="carousel-indicators">
                    <li data-target="#focusslide" data-slide-to="0"
class="active"></li>
                    <li data-target="#focusslide" data-slide-to="1"></li>
                </ol>
```

```
<div class="carousel-inner" role="listbox">
    <div class="item active">
        <a href="javascript:;" title="Python">
            </a>
        </div>
        <div class="item">
            <a href="javascript:;" title="Scratch">
                </a>
            </div>
        </div>
        <a class="left carousel-control" href="#focusslide" role="button" data-slide="prev" rel="nofollow">
            <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>
            <span class="sr-only">上一个</span>
        </a>
        <a class="right carousel-control" href="#focusslide" role="button" data-slide="next" rel="nofollow">
            <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
            <span class="sr-only">下一个</span>
        </a>
    </div>

{# 推荐内容 #}
<article class="excerpt-minic excerpt-minic-index">
    <h2>
        <span class="red">【推荐】</span>
    {# 文章标签 #}
        <a target="_blank" href="#" title="Spring Boot">Spring Boot</a>
    </h2>
    {# 文章简介 #}
    <p class="note">Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而
    </p>
</article>

{# 最新发布的文字，包括分类的文字 #}
<div class="title">
    <h3>最新发布</h3>
    <div class="more">
        <a href="#" title="Python">Python</a>
        <a href="#" title="Python">Python</a>
        <a href="#" title="Python">Python</a>
    </div>
</div>
{# 所有文章 #}
<article class="excerpt excerpt-1" style="">
    <a class="focus" href="#" title="Django框架" target="_blank">
        {# 文章图片 #}
        
    </a>
```

```
<header>
    {# 文章分类的名字 #-}
    <a class="cat" href="#" title="Python"
target=_blank>Python<i></i></a>
    {# 文章名字 #-}
    <h2><a href="#" title="Django" target="_blank">Django框架</a></h2>
</header>
<p class="meta">
    {# 文章添加时间 #-}
    <time class="time"><i class="glyphicon glyphicon-time"></i> 2022-7-25</time>
    {# 点击数 #-}
    <span class="views"><i class="glyphicon glyphicon-eye-open"></i> 0</span>
    {# 留言数 #-}
    <a class="comment" title="留言">
        <i class="glyphicon glyphicon-comment"></i> 0
    </a>
</p>
    {# 文章简介 #-}
    <p class="note">Django是一个开放源代码的web应用框架，由Python写成。</p>
</article>
<article class="excerpt excerpt-1" style="">
    <a class="focus" href="#" title="Django框架" target=_blank>
        {# 文章图片 #-}
        
    </a>
    <header>
        {# 文章分类的名字 #-}
        <a class="cat" href="#" title="Python"
target=_blank>Python<i></i></a>
        {# 文章名字 #-}
        <h2><a href="#" title="Django" target="_blank">Django框架</a></h2>
    </header>
    <p class="meta">
        {# 文章添加时间 #-}
        <time class="time"><i class="glyphicon glyphicon-time"></i> 2022-7-25</time>
        {# 点击数 #-}
        <span class="views"><i class="glyphicon glyphicon-eye-open"></i> 0</span>
        {# 留言数 #-}
        <a class="comment" title="留言">
            <i class="glyphicon glyphicon-comment"></i> 0
        </a>
    </p>
    {# 文章简介 #-}
    <p class="note">Django是一个开放源代码的web应用框架，由Python写成。</p>
</article>
<article class="excerpt excerpt-1" style="">
```

```
        <a class="focus" href="#" title="Django框架"
target=_blank">
    {# 文章图片 #-}
    
</a>
<header>
    {# 文章分类的名字 #-}
    <a class="cat" href="#" title="Python"
target=_blank">Python<i></i></a>
    {# 文章名字 #-}
    <h2><a href="#" title="Django" target=_blank">Django框
架</a></h2>
</header>
<p class="meta">
    {# 文章添加时间 #-}
    <time class="time"><i class="glyphicon glyphicon-time">
</i> 2022-7-25</time>
    {# 点击数 #-}
    <span class="views"><i class="glyphicon glyphicon-eye-
open"></i> 0</span>
    {# 留言数 #-}
    <a class="comment" title="留言">
        <i class="glyphicon glyphicon-comment"></i> 0
    </a>
</p>
    {# 文章简介 #-}
    <p class="note">Django是一个开放源代码的web应用框架，由Python写
成。</p>
</article>
<article class="excerpt excerpt-1" style="">
    <a class="focus" href="#" title="Django框架"
target=_blank">
        {# 文章图片 #-}
        
</a>
<header>
    {# 文章分类的名字 #-}
    <a class="cat" href="#" title="Python"
target=_blank">Python<i></i></a>
    {# 文章名字 #-}
    <h2><a href="#" title="Django" target=_blank">Django框
架</a></h2>
</header>
<p class="meta">
    {# 文章添加时间 #-}
    <time class="time"><i class="glyphicon glyphicon-time">
</i> 2022-7-25</time>
    {# 点击数 #-}
    <span class="views"><i class="glyphicon glyphicon-eye-
open"></i> 0</span>
    {# 留言数 #-}
    <a class="comment" title="留言">
        <i class="glyphicon glyphicon-comment"></i> 0
    </a>
</p>
```

```
</p>
{# 文章简介 #-}
<p class="note">Django是一个开放源代码的web应用框架，由Python写
成。</p>
</article>
<article class="excerpt excerpt-1" style="">
    <a class="focus" href="#" title="Django框架"
target=_blank">
        {# 文章图片 #-}
        
    </a>
    <header>
        {# 文章分类的名字 #-}
        <a class="cat" href="#" title="Python"
target=_blank">Python<i></i></a>
        {# 文章名字 #-}
        <h2><a href="#" title="Django" target=_blank">Django框
架</a></h2>
        </header>
        <p class="meta">
            {# 文章添加时间 #-}
            <time class="time"><i class="glyphicon glyphicon-time">
</i> 2022-7-25</time>
            {# 点击数 #-}
            <span class="views"><i class="glyphicon glyphicon-eye-
open"></i> 0</span>
            {# 留言数 #-}
            <a class="comment" title="留言">
                <i class="glyphicon glyphicon-comment"></i> 0
            </a>
        </p>
        {# 文章简介 #-}
        <p class="note">Django是一个开放源代码的web应用框架，由Python写
成。</p>
    </article>
</div>
<div class="widget widget_sentence">
    <h3>标签云</h3>
    <div class="widget-sentence-content">
        <ul class="plinks ptags">
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">Flask
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">爬虫
                    <span class="badge">1</span>
                </a>
            </li>
            <li>

```

```
<a href="#" title="Django" draggable="false">spring
    <span class="badge">1</span>
</a>
</li>
<li>
    <a href="#" title="Django" draggable="false">Django
        <span class="badge">1</span>
    </a>
</li>
</ul>
</div>
</div>
</div>
{# 右边部分 #}
<aside class="sidebar">
    <div class="fixed">
        {# 统计区域，只有主页才有 #}
        <div class="widget widget-tabs">
            <ul class="nav nav-tabs" role="tablist">
                <li role="presentation" class="active"><a href="#notice" aria-controls="notice" role="tab" data-toggle="tab">统计信息</a></li>
                <li role="presentation"><a href="#contact" aria-controls="contact" role="tab" data-toggle="tab">联系站长</a></li>
            </ul>
            <div class="tab-content">
                {# 统计信息 #}
                <div role="tabpanel" class="tab-pane contact active" id="notice">
                    <h2>文章总数：
                        10篇
                    </h2>
                    <h2>网站运行：
                        <span id="sitetime">10天 </span></h2>
                </div>
                {# 联系站长 #}
                <div role="tabpanel" class="tab-pane contact" id="contact">
                    <h2>WX:Allen
                        <a href="" target="_blank" rel="nofollow" data-toggle="tooltip" data-placement="bottom" title="" data-original-title="QQ:853307553"></a>
                    </h2>
                    <h2>Email:449837498@qq.com
                        <a href="#" target="_blank" data-toggle="tooltip" rel="nofollow" data-placement="bottom" title="" data-original-title="#"></a></h2>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

<div class="widget widget_search">
    <form class="navbar-form" action="#">
        <div class="input-group">
            <input type="text" name="keyword" class="form-control" size="35" placeholder="请输入关键字" maxlength="15" autocomplete="off">
            <span class="input-group-btn">
```



```
<span class="text">Flask框架</span>
<span class="muted">
    <i class="glyphicon glyphicon-time"></i>
    2022-7-25 {# 文章的发布时间 #-}
</span>
<span class="muted">
    <i class="glyphicon glyphicon-eye-open"></i>
    1 {# 文章的观看数 #-}
</span>
</a>
</li>
<li>
    <a title="Django框架" href="#" target="_blank">
        <span class="thumbnail">
            {# 文章的图片 #-}
            
        </span>
            {# 文章的名字 #-}
            <span class="text">Flask框架</span>
            <span class="muted">
                <i class="glyphicon glyphicon-time"></i>
                2022-7-25 {# 文章的发布时间 #-}
            </span>
            <span class="muted">
                <i class="glyphicon glyphicon-eye-open"></i>
                1 {# 文章的观看数 #-}
            </span>
        </a>
    </li>
    <li>
        <a title="Django框架" href="#" target="_blank">
            <span class="thumbnail">
                {# 文章的图片 #-}
                
            </span>
            {# 文章的名字 #-}
            <span class="text">Flask框架</span>
            <span class="muted">
                <i class="glyphicon glyphicon-time"></i>
                2022-7-25 {# 文章的发布时间 #-}
            </span>
            <span class="muted">
                <i class="glyphicon glyphicon-eye-open"></i>
                1 {# 文章的观看数 #-}
            </span>
        </a>
    </li>
    <li>
        <a title="Django框架" href="#" target="_blank">
            <span class="thumbnail">
                {# 文章的图片 #-}
                
            </span>
            {# 文章的名字 #-}
            <span class="text">Flask框架</span>
            <span class="muted">
                <i class="glyphicon glyphicon-time"></i>
                2022-7-25 {# 文章的发布时间 #-}
            </span>
            <span class="muted">
                <i class="glyphicon glyphicon-eye-open"></i>
                1 {# 文章的观看数 #-}
            </span>
        </a>
    </li>
    <li>
```

```
</span>
{# 文章的名字 #-}
<span class="text">Flask框架</span>
<span class="muted">
    <i class="glyphicon glyphicon-time"></i>
    2022-7-25 {# 文章的发布时间 #-}
</span>
<span class="muted">
    <i class="glyphicon glyphicon-eye-open"></i>
    1 {# 文章的观看数 #-}
</span>
</a>
</li>
<li>
    <a title="Django框架" href="#" target="_blank">
        <span class="thumbnail">
            {# 文章的图片 #-}
            
        </span>
        {# 文章的名字 #-}
        <span class="text">Flask框架</span>
        <span class="muted">
            <i class="glyphicon glyphicon-time"></i>
            2022-7-25 {# 文章的发布时间 #-}
        </span>
        <span class="muted">
            <i class="glyphicon glyphicon-eye-open"></i>
            1 {# 文章的观看数 #-}
        </span>
        </a>
    </li>
</ul>
</div>

{# 广告区域 #
<div class="widget widget_sentence">
    <a href="https://www.aliyun.com" rel="nofollow" title="阿里云" target="_blank">
        
    </a>
</div>
<div class="widget widget_sentence">
    <a href="https://cloud.tencent.com/" rel="nofollow" title="腾讯云" target="_blank">
        
    </a>
</div>
<div class="widget widget_sentence">
    <h3>友情链接</h3>
    <div class="widget-sentence-link">
        <a href="https://www.python.org/" title="Python">Python官网</a>
        <a href="https://docs.djangoproject.com/zh-hans/3.2/" title="Django官网">Django官网</a>
        <a href="https://github.com/" title="github">github</a>
    </div>
</div>
```

```

        </div>
    </aside>
    {# 向上的按钮 #-}
    <ul class="sidebar">
        <li class="totop"></li>
    </ul>
</section>
{# 底部信息 #-}
<footer class="footer">
    <div class="container">
        <p>
            Copyright &copy; 2022.知吾所想 予吾所好-Alle
        </p>
    </div>
    <div id="gotop"><a class="gotop"></a></div>
</footer>

<script src="{% static 'js/bootstrap.min.js' %}"></script>
<script src="{% static 'js/jquery.ias.js' %}"></script>
<script src="{% static 'js/scripts.js' %}"></script>
<script>
    var _hmt = _hmt || [];
    (function () {
        var hm = document.createElement("script");
        hm.src = "https://hm.baidu.com/hm.js?
e8ae61fbc1aa8b44823aae4cd23690b0";
        var s = document.getElementsByTagName("script")[0];
        s.parentNode.insertBefore(hm, s);
    })();
</script>

</body>
</html>

```

人生苦短,我用Python

django

统计信息 联系站长

文章总数: 10篇 网站运行: 10天

请输入关键字 搜索

标签云

Django ① Django ① Django ①
Django ① Django ①

热门推荐

Flask 框架 2022-7-25 ① 1

Flask 框架 2022-7-25 ① 1

Flask 框架 2022-7-25 ① 1

最新发布

Python Django Python Python

【推荐】Spring Boot
Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而

Python Django Python

django Python Django框架 2022-7-25 ① 0 ① 0
Django是一个开放源代码的Web应用框架，由Python写成。

django Python Django框架 2022-7-25 ① 0 ① 0
Django是一个开放源代码的Web应用框架，由Python写成。

CSDN @__Mr徐

6.2 创建模型类

- 在articles的models中需要创建三个表，并生成迁移文件

```
# articles\models.py文件

import random
import time
from datetime import datetime

from django.db import models

from users.models import UserProfile


class Category(models.Model):
    """文章分类"""
    name = models.CharField(max_length=20, verbose_name='分类名')
    add_time = models.DateTimeField(default=datetime.now, verbose_name='创建时间')
    is_tab = models.BooleanField(default=True, verbose_name='是否导航')

    def __str__(self):
        return self.name

    class Meta:
        db_table = 'Category'
        verbose_name = '类别表'
        verbose_name_plural = verbose_name


class TagInfo(models.Model):
    """文章标签"""
    name = models.CharField(max_length=20, verbose_name='标签名')
    add_time = models.DateTimeField(default=datetime.now, verbose_name='创建时间')
    category = models.ForeignKey(Category, verbose_name='所属分类',
                                 on_delete=models.DO_NOTHING)

    def __str__(self):
        return self.name

    class Meta:
        db_table = 'TagInfo'
        verbose_name = '标签表'
        verbose_name_plural = verbose_name


m_y = (time.strftime("%Y"))
m_m = (time.strftime("%m"))

class ArticleInfo(models.Model):
    """文章信息"""
    title = models.CharField(max_length=50, verbose_name='标题')
    desc = models.TextField(max_length=80, verbose_name='简介')
    content = models.TextField(verbose_name='文章内容')
    click_num = models.IntegerField(default=0, verbose_name='浏览数')
    cont_num = models.IntegerField(default=0, verbose_name='留言数')
```

```
love_num = models.IntegerField(default=0, verbose_name='点赞数')
image = models.ImageField(upload_to='%y/%m/%d', verbose_name='封面',
max_length=200,
                           default='article/default' +
str(random.choice('12345')) + '.jpg', null=True, blank=True)
author = models.ForeignKey(UserProfile, verbose_name='文章作者',
on_delete=models.DO_NOTHING)
category = models.ForeignKey(Category, verbose_name='所属类别',
on_delete=models.DO_NOTHING)
add_time = models.DateTimeField(default=datetime.now, verbose_name='发表
时间')
is_recommend = models.BooleanField(default=False, verbose_name='首页推
荐')
taginfo = models.ForeignKey(TagInfo, verbose_name='所属标签', null=True,
blank=True, on_delete=models.DO_NOTHING)

def __str__(self):
    return self.title

class Meta:
    db_table = 'ArticleInfo'
    verbose_name = '文章表'
    verbose_name_plural = verbose_name
```

- Terminal中输入命令来生成迁移文件

```
python manage.py makemigrations
python manage.py migrate
```

6.3 添加数据

- 首页要展示文章，我们就得添加数据，刚刚已经在模型当中添加了三个表，此时可以创建超级用户，在后面添加一些数据来展示

6.3.1 创建超级用户

- 打开Terminal输入命令来创建超级用户

```
python manage.py createsuperuser
```

The screenshot shows a terminal window with the following text:

```
Terminal: Local × Local (2) × +
(myblog) C:\Users\Xumf0113\Desktop\new_blog\3\myblog>python manage.py createsuperuser
用户名: admin
电子邮件地址:
Password:
Password (again):
密码跟用户名太相似了。
密码长度太短。密码必须包含至少 8 个字符。
这个密码太常见了。
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

(myblog) C:\Users\Xumf0113\Desktop\new_blog\3\myblog>
```

At the top of the terminal window, there are tabs: Local, Local (2), and +. The current tab is Local. The status bar at the bottom shows: 6: TODO, 4: Run, Python Console, Terminal, and CSDN @__Mr徐.

6.3.2 注册模型类

- 为了能够在后台手动添加数据, 还得将模型类添加到admin文件中, 后台才能实施

```
# articles\admin.py文件

from django.contrib import admin

from .models import TagInfo, ArticleInfo, Category


@admin.register(ArticleInfo)
class ArticleInfoAdmin(admin.ModelAdmin):
    # 展示的字段
    list_display = ['title', 'click_num', 'cont_num', 'author', 'category',
    'taginfo', 'image', 'add_time']
    # 可以搜索
    search_fields = ['title']


@admin.register(TagInfo)
class TagInfoAdmin(admin.ModelAdmin):
    # 展示的字段
    list_display = ['name', 'category', 'add_time']
    # 可以搜索
    search_fields = ['name']


@admin.register(Category)
class CategoryAdmin(admin.ModelAdmin):
    # 展示的字段
    list_display = ['name', 'add_time']
    # 可以搜索
    search_fields = ['name']
```

6.3.3 添加数据

- 通过在admin后台手动添加数据到博客中

ARTICLES

文章表	+ 增加
标签表	+ 增加
类别表	+ 增加
认证和授权	
组	+ 增加

«

选择 类别表 来修改

动作 5个中 0 个被选

<input type="checkbox"/>	分类名	创建时间
<input type="checkbox"/>	PHP	2022年7月25日 19:55
<input type="checkbox"/>	C++	2022年7月25日 19:55
<input type="checkbox"/>	C	2022年7月25日 19:55
<input type="checkbox"/>	Java	2022年7月25日 19:55
<input type="checkbox"/>	Python	2022年7月25日 19:54

5类别表

CSDN @__Mr徐

ARTICLES

文章表	+ 增加
标签表	+ 增加
类别表	+ 增加
认证和授权	
组	+ 增加

«

选择 标签表 来修改

动作 8个中 0 个被选

<input type="checkbox"/>	标签名	所属分类	创建时间
<input type="checkbox"/>	ThinkPHP	PHP	2022年7月25日 19:58
<input type="checkbox"/>	Laravel	PHP	2022年7月25日 19:58
<input type="checkbox"/>	指针	C	2022年7月25日 19:57
<input type="checkbox"/>	IntelliJ IDEA	Java	2022年7月25日 19:56
<input type="checkbox"/>	spring	Java	2022年7月25日 19:55
<input type="checkbox"/>	爬虫	Python	2022年7月25日 19:55
<input type="checkbox"/>	Flask	Python	2022年7月25日 19:55
<input type="checkbox"/>	Django	Python	2022年7月25日 19:55

8标签表

CSDN @__Mr徐

6.4 CKEditor

- 在运营后台，运营人员需要录入文章并编辑文章的详情信息，而文章的详情信息不是普通的文本，可以是包含了HTML语法格式的字符串。为了快速简单的让用户能够在页面中编辑带格式的文本，我们引入富文本编辑器。富文本即具备丰富样式格式的文本。
- 我们使用功能强大的CKEditor富文本编辑器。

详细介绍*

MacBook Pro 您最得力的助手

它纤薄如刃，轻盈如羽，却又比以往速度更快、性能更强大。它为你展现的，是迄今最明亮、最多彩的 Mac 笔记本显示屏。它更配备了触控栏，一个内置于键盘的玻璃面多点触控条，让你能在需要时快速取用各种工具。MacBook Pro 是对我们突破性理念的一场出色演绎，而它，也正期待着演绎你的奇思妙想。

MacBook Pro 13 英寸 Multi-Touch Bar 速览 [详细商品介绍请往下浏览 >>](#)

全新互动方式

Multi-Touch Bar 会根据你当前所在的 app，将有

13 英寸 Retina 显示屏

拥有出色对比度和P3广色域技术，让你看到的白色更明

CSDN @__Mr徐

1. 安装

```
pip install django-ckeditor
```

2. 添加应用

```
# settings.py文件

INSTALLED_APPS = [
    ...
    'ckeditor', # 富文本编辑器
    'ckeditor_uploader', # 富文本编辑器上传图片模块
    ...
]
```

3. 添加ckeditor配置

```
# settings.py文件

# 富文本编辑器ckeditor配置
CKEDITOR_CONFIGS = {
    'default': {
        'toolbar': 'full', # 工具条功能
        'height': 300, # 编辑器高度
        # 'width': 300, # 编辑器宽
    },
}

# 这里是否配置了图片上传地址，跟media路径一样
CKEDITOR_UPLOAD_PATH = ''
```

4. 在总路由中添加ckeditor路由

```
# urls.py文件

path('ckeditor', include('ckeditor_uploader.urls')), # 富文本编辑器
```

5. 为模型类添加字段

- 将文章类中的 `content` 字段变成 `ckeditor` 提供的字段

```
# articles\models.py文件

from ckeditor_uploader.fields import RichTextUploadingField

# content = models.TextField(verbose_name='文章内容')
content = RichTextUploadingField(verbose_name='文章内容')
```

6. 修改了模型的字段后得生成并迁移文件

```
python manage.py makemigrations  
python manage.py migrate
```

6.5 添加文章数据

- 已经添加 ckeditor，可以通过后台添加数据

动作	标题	浏览数	评论数	文章作者	所属类别	所属标签	封面	发表时间
<input type="checkbox"/>	数据分析	0	0	admin	Python	-	22/07/25/数据分析.jpg	2022年7月25日 20:44
<input type="checkbox"/>	爬虫	0	0	449837498@qq.com	Python	爬虫	22/07/25/default2.jpg	2022年7月25日 20:42
<input type="checkbox"/>	Laravel	0	0	449837498@qq.com	PHP	Laravel	22/07/25/Laravel.jpg	2022年7月25日 20:40
<input type="checkbox"/>	IntelliJ IDEA	0	0	admin	Java	IntelliJ IDEA	22/07/25/IDEA.jpeg	2022年7月25日 20:37
<input type="checkbox"/>	Spring Boot	0	0	449837498@qq.com	Java	spring	22/07/25/spring.jpeg	2022年7月25日 20:34
<input type="checkbox"/>	Flask框架	0	0	449837498@qq.com	Python	-	22/07/25/Flask.jpg	2022年7月25日 20:25
<input type="checkbox"/>	Django框架	0	0	admin	Python	Django	article/default5.jpg	2022年7月25日 20:23

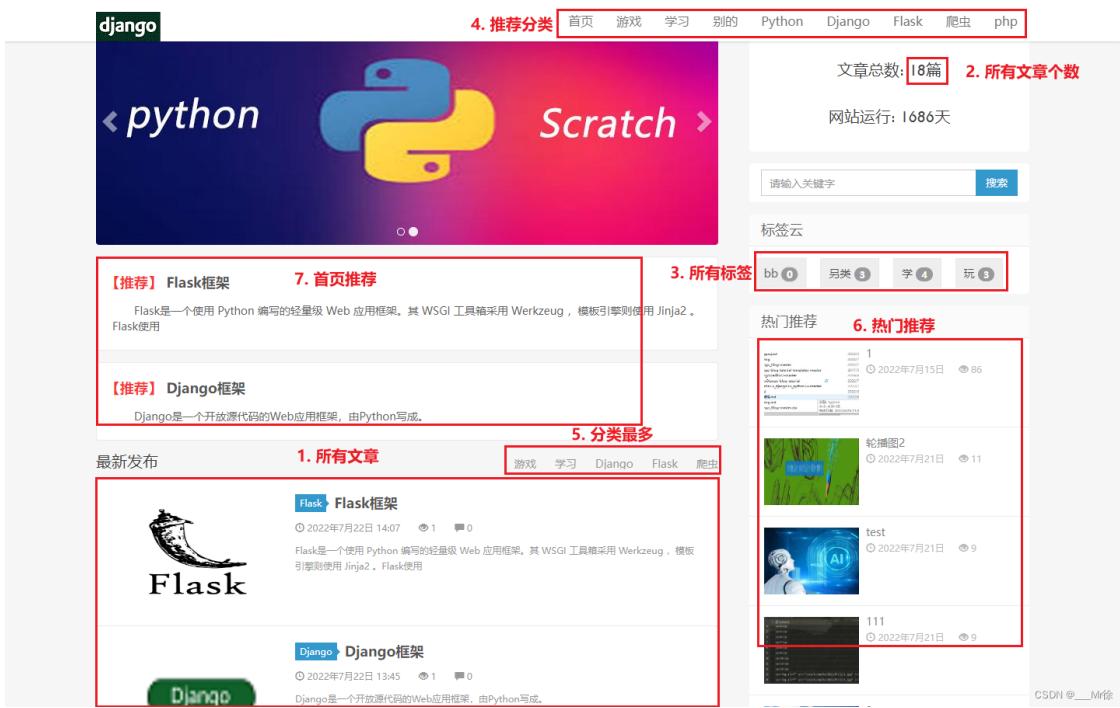
6.6 首页展示

- 数据已经有了，可以通过首页将数据展示出来

6.6.1 功能实现

- 首页需要展示的数据有哪些

- 拿所有文章
- 所有文章个数
- 所有标签
- 推荐分类
- 分类最多
- 热门推荐8个
- 首页推荐2个



- 修改视图, 获取数据

```
# articles\views.py文件

import datetime

from django.db.models import Q, Count
from django.shortcuts import render, redirect, get_object_or_404

from utils.format_time import Caltime
from .models import Category, ArticleInfo, TagInfo


def index(request):
    """
    主页的展示
    1. 拿所有文章
    2. 所有文章个数
    3. 所有标签
    4. 推荐分类
    5. 分类最多
    6. 热门推荐8个
    7. 首页推荐2个
    """

    # 1. 拿所有的文章
    all_articles = ArticleInfo.objects.all().order_by('-add_time')
    # 2. 获取所有文章的个数
    article_total = all_articles.count()
    # 3. 获取所有文章的标签
    all_tags = TagInfo.objects.all()

    # 4. 根据分类时间进行升序拿所有推荐分类
    all_category = Category.objects.filter(is_tab=True).order_by('add_time')

    # 5. 根据文章数拿分类中最多的5个
    
```

```

many_category =
ArticleInfo.objects.values('category__name').annotate(Count('category__name'))[:5]

# 6. 热门推荐前8篇(点击量最多的)
hot_articles = ArticleInfo.objects.order_by('-click_num')[:8]

# 7. 根据创建时间来拿首页推荐的前2个
recommend_article =
ArticleInfo.objects.filter(is_recommend=True).order_by('-add_time')[:2]

# 计算网站已经运营时长
all_day = 10

return render(request, 'index.html', {
    'all_category': all_category, # 所有分类
    'all_tags': all_tags, # 所有标签
    'hot_articles': hot_articles, # 热门推荐的文章
    'recommend_article': recommend_article, # 推荐文章
    'many_category': many_category, # 分类最多的
    'all_day': all_day, # 总天数
    'article_total': article_total, # 文章个数
})

```

6.6.2 功能提取

- 在上面的视图函数中, 运营时长得修改
- 根据真正运营开始的时间开始计算, 得到最终的时间
- 在utils文件夹中创建 `format_time.py` 文件

```

import time
import datetime

# 计算两个日期相差天数, 自定义函数名, 和两个日期的变量名。
def Caltime():
    # 生成当前日期, 样式为: 2022-07-15
    date1 = '2022-07-15'
    today_time = datetime.datetime.now().strftime("%Y-%m-%d")

    date1 = time.strptime(date1, "%Y-%m-%d")
    date2 = time.strptime(today_time, "%Y-%m-%d")
    date1 = datetime.datetime(date1[0], date1[1], date1[2])
    date2 = datetime.datetime(date2[0], date2[1], date2[2])
    return (date2 - date1).days

```

6.6.3 模板传递

- 使用Django模板语言, 将视图当中的数据传递给到模板当中

```
{% load static %}

<!doctype html>
<html lang="zh-CN">
<head>
    <meta name="360-site-verification" content="85326d9c1b0d512826605334e6eb1d5c">
    <meta charset="utf-8">
    <meta name="renderer" content="webkit">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="baidu_union_verify" content="6c3c4420bcc5cb0d05563cc88180cd88">
    <title>Django博客</title>
    <meta name="keywords" content="Django博客">
    <link rel="stylesheet" type="text/css" href="{% static 'css/bootstrap.min.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/nprogress.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/shang.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/fontawesome.min.css' %}">
    <link rel="apple-touch-icon-precomposed" href="{% static 'images/icon.png' %}">
    <link rel="shortcut icon" href="{% static 'images/favicon.ico' %}">
    <script src="{% static 'js/jquery-2.1.4.min.js' %}"></script>
    <script src="{% static 'js/nprogress.js' %}"></script>
    <script src="{% static 'js/jquery.lazyload.min.js' %}"></script>
</head>
<body class="user-select">
<header class="header">
    <nav class="navbar navbar-default" id="navbar">
        <div class="container">
            {# 头部最上层 #}
            <div class="header-topbar hidden-xs link-border">
                <ul class="site-nav topmenu">
                    {# 如果用户是登录则显示名字#}
                    {% if request.user.is_authenticated %}
                        <li><a href="#"@{{ request.user.username }}></a>
                    </li>
                    {# 如果未登录，则显示登录或注册 #}
                    {% else %}
                        <li><a href="{% url 'user_login' %}">登录</a></li>
                        <li><a href="{% url 'user_register' %}">注册</a></li>
                    {% endif %}
                </ul>
                人生苦短，我用Python
            </div>
            {# 头部下面左 #}
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#header-navbar" aria-expanded="false">
                    <span class="sr-only"></span>
                </button>
            </div>
            {# 头部中间 #}
            <div class="navbar-collapse collapse" id="header-navbar">
                <ul class="nav navbar-nav" style="list-style-type: none; padding-left: 0;">
                    {# 如果未登录，则显示登录或注册 #}
                    {% else %}
                        <li><a href="{% url 'user_login' %}">登录</a></li>
                        <li><a href="{% url 'user_register' %}">注册</a></li>
                    {% endif %}
                </ul>
            </div>
        </div>
    </nav>
</header>
<div class="content" style="margin-top: 10px; margin-bottom: 10px;">
    {# 中间内容 #}
</div>
<div class="footer" style="text-align: center; margin-top: 10px; margin-bottom: 10px;">
    {# 底部内容 #}
</div>

```

```

        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
    </button>
    <h1 class="logo hvr-bounce-in">
        <a href="{% url 'index' %}" title="首页">
            
        </a>
    </h1>
</div>
{# 头部下面右 #}
<div class="collapse navbar-collapse" id="header-navbar">
    <ul class="nav navbar-nav navbar-right">
        <li>
            <a data-cont="首页" title="首页" href="{% url 'index'
%}">首页
            </a>
        </li>
        {# 所有的分类 #}
        {% for cate in all_category %}
            <li>
                <a data-cont="python" title="{{ cate.name }}" href="#>{{ cate.name }}</a>
            </li>
        {% endfor %}
    </ul>
</div>
</div>
</div>
</nav>
</header>
{# 中间区域， 轮播图+文章+标签等所有 #}
<section class="container">
    {# 左边部分 #}
    <div class="content-wrap">
        {# 中间文章的区域 #}
        <div class="content">
            {# 轮播图 #}
            <div id="focusslide" class="carousel slide" data-
ride="carousel">
                <ol class="carousel-indicators">
                    <li data-target="#focusslide" data-slide-to="0"
class="active"></li>
                    <li data-target="#focusslide" data-slide-to="1"></li>
                </ol>

                <div class="carousel-inner" role="listbox">
                    <div class="item active">
                        <a href="javascript:;" title="Python">
                            </a>
                        </div>
                    <div class="item">
                        <a href="javascript:;" title="Scratch">
                            </a>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

        <a class="left carousel-control" href="#focusslide"
role="button" data-slide="prev" rel="nofollow">
            <span class="glyphicon glyphicon-chevron-left" aria-
hidden="true"></span>
            <span class="sr-only">上一个</span>
        </a>
        <a class="right carousel-control" href="#focusslide"
role="button" data-slide="next" rel="nofollow">
            <span class="glyphicon glyphicon-chevron-right" aria-
hidden="true"></span>
            <span class="sr-only">下一个</span>
        </a>
    </div>

{# 推荐内容 #-}
{% for art in recommend_article %}
    <article class="excerpt-minic excerpt-minic-index">
        <h2>
            <span class="red">【推荐】</span>
            {# 文章标签 #-}
            <a target="_blank" href="#" title="{{ art.title }}">
{{ art.title }}</a>
        </h2>
        {# 文章简介 #-}
        <p class="note">{{ art.desc }}</p>
    </article>
    {% endfor %}

{# 最新发布的文字，包括分类的文字 #-}
<div class="title">
    <h3>最新发布</h3>
    <div class="more">
        {% for cate in many_category %}
            <a href="#" title="{{ cate.category__name }}>{{
cate.category__name }}</a>
        {% endfor %}
    </div>
</div>
{# 所有文章 #-}
{% for art in all_articles %}
    <article class="excerpt excerpt-1" style="">
        <a class="focus" href="#" title="{{ art.title }}"
target=_blank>
            {# 文章图片 #-}
            
        </a>
        <header>
            {# 文章分类的名字 #-}
            <a class="cat" href="#" title="{{ art.category.name
}}" target=_blank>{{ art.category.name }}<i></i></a>
            {# 文章名字 #-}
            <h2><a href="#" title="{{ art.title }}>
{{ art.title }}</a></h2>
            </header>
            <p class="meta">
                {# 文章添加时间 #-}

```

```

                <time class="time"><i class="glyphicon glyphicon-time"></i> {{ art.add_time }}</time>
                    {# 点击数 #}
                    <span class="views"><i class="glyphicon glyphicon-eye-open"></i> {{ art.click_num }}</span>
                        {# 留言数 #}
                        <a class="comment" title="留言">
                            <i class="glyphicon glyphicon-comment"></i> {{ art.cont_num }}
                        </a>
                    </p>
                    {# 文章简介 #}
                    <p class="note">{{ art.desc }}</p>
                </article>
            {% endfor %}
        </div>
        <div class="widget widget_sentence">
            <h3>标签云</h3>
            <div class="widget-sentence-content">
                <ul class="plinks ptags">
                    {# 只拿最新的前11个 #}
                    {% for tag in all_tags|slice:'10:-1' %}
                        <li>
                            <a href="#" title="{{ tag.name }}" draggable="false">{{ tag.name }}</a>
                            <span class="badge">{{ tag.articleinfo_set.count }}</span>
                        </li>
                    {% endfor %}
                </ul>
            </div>
        </div>
        {# 右边部分 #}
        <aside class="sidebar">
            <div class="fixed">
                {# 统计区域，只有主页才有 #}
                <div class="widget widget-tabs">
                    <ul class="nav nav-tabs" role="tablist">
                        <li role="presentation" class="active"><a href="#notice" aria-controls="notice" role="tab" data-toggle="tab">统计信息</a></li>
                        <li role="presentation"><a href="#contact" aria-controls="contact" role="tab" data-toggle="tab">联系站长</a></li>
                    </ul>
                    <div class="tab-content">
                        {# 统计信息 #}
                        <div role="tabpanel" class="tab-pane contact active" id="notice">
                            <h2>文章总数:</h2>
                            {{ article_total }}篇
                            </h2>
                            <h2>网站运行:</h2>
                            <span id="sitetime">{{ all_day }}天 </span></h2>
                        </div>
                        {# 联系站长 #}
                        <div role="tabpanel" class="tab-pane contact" id="contact">

```

```
<h2>wx:Allen
      <a href="" target="_blank" rel="nofollow" data-
toggle="tooltip" data-placement="bottom" title=""
          data-original-title="QQ:853307553"></a>
    </h2>
    <h2>Email:449837498@qq.com
        <a href="#" target="_blank" data-
toggle="tooltip" rel="nofollow" data-placement="bottom" title="" data-
original-title="#"></a></h2>
    </div>
</div>

<div class="widget widget_search">
    <form class="navbar-form" action="#">
        <div class="input-group">
            <input type="text" name="keyword" class="form-
control" size="35" placeholder="请输入关键字" maxlength="15"
autocomplete="off">
            <span class="input-group-btn">
                <button class="btn btn-default btn-search"
type="submit">搜索</button>
            </span>
        </div>
    </form>
</div>
{# 右下的标签云 #-}
<div class="widget widget_sentence">
    <h3>标签云</h3>
    <div class="widget-sentence-content">
        <ul class="plinks ptags">
            {# 只将分类当中的后10个拿出来 #}
            {% for tag in all_tags|slice:'10:-1' %}
                <li>
                    <a href="#" title="{{ tag.name }}"
draggable="false">
                        {{ tag.name }}
                    <span class="badge">{{
tag.articleinfo_set.count }}</span>
                    </a>
                </li>
            {% endfor %}
        </ul>
    </div>
</div>
{# 最新留言文章 #-}
<div class="widget widget_hot">
    <h3>热门推荐</h3>
    <ul>
        {% for art in hot_articles %}
            <li>
                <a title="{{ art.title }}" href="#" target="_blank">
                    <span class="thumbnail">
                        {# 文章的图片 #-}
                        
        </span>
        {# 文章的名字 #
<span class="text">{{ art.title }}</span>
<span class="muted">
        <i class="glyphicon glyphicon-time"></i>
{{ art.add_time|date }}  {# 文章的发布时间
#}
        </span>
<span class="muted">
        <i class="glyphicon glyphicon-eye-open">

```

</i>

```

                {{ art.click_num }}  {# 文章的观看数 #
            </span>
            </a>
        </li>
    {% endfor %}
</ul>
</div>

{# 广告区域 #
<div class="widget widget_sentence">
    <a href="https://www.aliyun.com" rel="nofollow" title="阿里云"
target="_blank">
        </a>
    </div>
    <div class="widget widget_sentence">
        <a href="https://cloud.tencent.com/" rel="nofollow" title="腾讯
云" target="_blank">
            </a>
        </div>
    <div class="widget widget_sentence">
        <h3>友情链接</h3>
        <div class="widget-sentence-link">
            <a href="https://www.python.org/" title="Python">Python官网
</a>
            <a href="https://docs.djangoproject.com/zh-hans/3.2/" title="Django官网">Django官网</a>
            <a href="https://github.com/" title="github">github</a>
        </div>
    </div>
</aside>
{# 向上的按钮 #
<ul class="sidebar">
    <li class="totop"></li>
</ul>
</section>
{# 底部信息 #
<footer class="footer">
    <div class="container">
        <p>
            Copyright &copy; 2022.知吾所想 予吾所好-Allen
        </p>

```

```

        </div>
    <div id="gotop"><a class="gotop"></a></div>
</footer>

<script src="{% static 'js/bootstrap.min.js' %}"></script>
<script src="{% static 'js/jquery.ias.js' %}"></script>
<script src="{% static 'js/scripts.js' %}"></script>
<script>
    var _hmt = _hmt || [];
    (function () {
        var hm = document.createElement("script");
        hm.src = "https://hm.baidu.com/hm.js?
e8ae61fbc1aa8b44823aae4cd23690b0";
        var s = document.getElementsByTagName("script")[0];
        s.parentNode.insertBefore(hm, s);
    })();
</script>

</body>
</html>

```



7. 分页

- 一旦文章数很多, 那么肯定得实现能够翻页的功能, 可以通过 `django_pure_pagination` 来实现分页
- `django_pure_pagination` 是第三方扩展包, 所以得通过命令来安装

1. 安装模块

- 打开cmd输入 `pip install django_pure_pagination`

2. 注册分页器

- 在settings.py中增加添加分页器

```
# settings.py文件

INSTALLED_APPS = [
    ...
    'pure_pagination', # 分页器应用
]
```

3. 配置分页器

- 在settings.py文件中配置分页器的样式

```
# settings.py文件

# 配置分页器的样式
PAGINATION_SETTINGS = {
    'PAGE_RANGE_DISPLAYED': 5, # 展示出来的数字个数
    'MARGIN_PAGES_DISPLAYED': 2, # 省略号前后有几个
    'SHOW_FIRST_PAGE_WHEN_INVALID': True, # 在无效页码的时候显示到第1页
}
```

4. 视图中的分页

- 在视图当中编写分页代码

```
# articles\views.py文件
from pure_pagination import Paginator

def index(request):
    ...
    # 分页
    # 获取从前面点击的页面数字， 默认1
    current_page = request.GET.get('page', 1)
    # 每页有5条数据
    paginator = Paginator(all_articles, 5)
    # 如果手动输入的数字大于实际页码数， 就回到第1页
    if int(current_page) > paginator.num_pages:
        current_page = 1
    # 生成一页中的文章
    page_obj = paginator.page(current_page)
    ...
    # 同时将render中的all_articles传递值变成page_obj
```

5. 前端样式

- 在templates下面创建 `pagination.html` 文件, 使用bootstrap美化分页

```
{# 分页器 #}
```

```

<nav style="text-align: center">
    <ul class="pagination">
        {% if page_obj.has_previous %} {# 如果有上一页才可以点击<< #}
            <li class="previous">
                {# 此处相对下面加了a标签, 可以点击 #}
                <a href="?{{ page_obj.previous_page_number.querystring }}">
                    <span aria-hidden="true">&laquo;/</span>
                </a>
            </li>
        {% else %} {# 否则禁止点击上一页 #}
            <li class="previous disabled">
                <span aria-hidden="true">&laquo;/</span>
            </li>
        {% endif %}

        {% for page in page_obj.pages %}{# 遍历所有页数 #}
            {% if page %}
                {# 如果遍历到的数字和当前页相同, 则添加active, 高度 #}
                {% ifequal page page_obj.number %}
                    <li class="active"><a href="javascript:;">{{ page }}</a>
                </li>
                {% else %}
                    <li><a href="?{{ page.querystring }}" class="page">{{ page }}</a></li>
                {% endifequal %}
            {% else %}
                <li class="disabled">
                    <span>...</span>
                </li>
            {% endif %}
        {% endfor %}

        {% if page_obj.has_next %}{# 如果有下一页才可以点击>> #}
            <li class="next">
                {# 此处相对下面加了a标签, 可以点击 #}
                <a href="?{{ page_obj.next_page_number.querystring }}">
                    <span aria-hidden="true">&raquo;/</span>
                </a>
            </li>
        {% else %}
            <li class="next disabled">
                <span aria-hidden="true">&raquo;/</span>
            </li>
        {% endif %}
    </ul>
</nav>

```

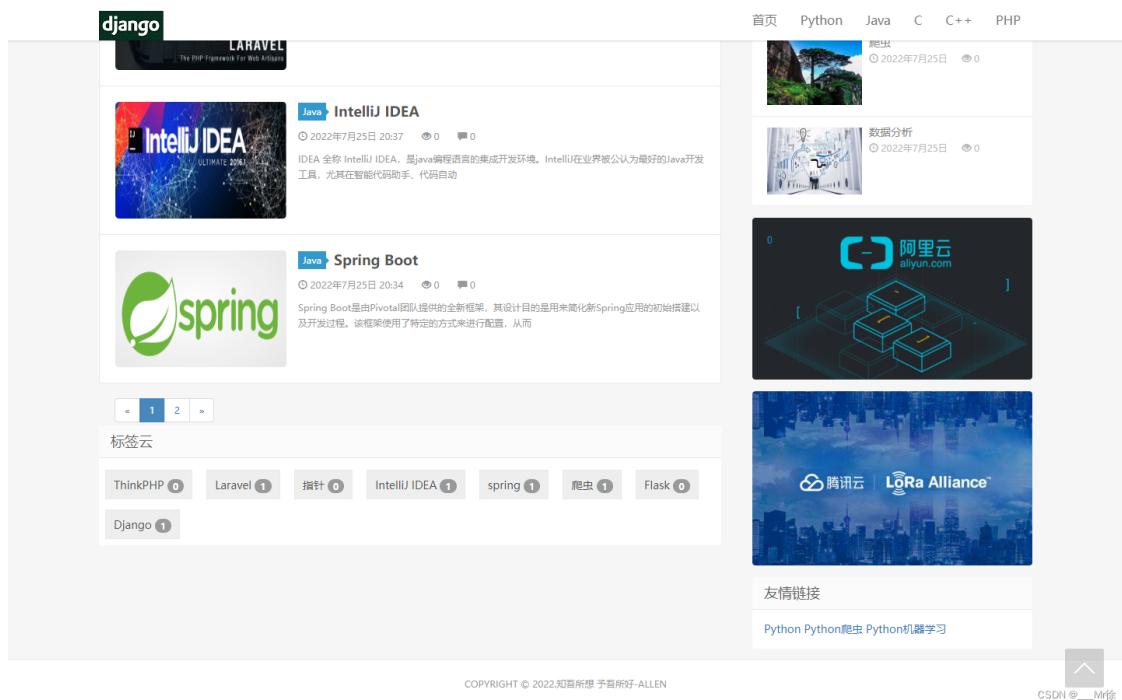
6. 在 `index.html` 中将分页的模板引入进来, 同时调整展示的文章

```

{% for art in all_articles %}
{# 1. 将以下内容替换成以下内容 #}
{% for art in page_obj.object_list %}

{# 2. 遍历所有文章的下面添加以下代码 #}
{# 分页器 #}
{% include 'pagination.html' %}

```



8. 列表页

- 首页数据已经能够展示,但是页面的数据还不能够实现相对应的跳转,当我们点击某文章应该能够进入到文章详情页面里面去,当我们点击某分类应该能够拿到某分类列表页

8.1 列表页需求

- 在首页有两块地方有分类信息
 1. 首页最上面推荐分类
 2. 最新发布文章右边的分类最多
- 只需要给他们添加跳转的路由,然后实现对应的视图函数来实现功能即可
- 注意:
 - 在点击进入到列表页的时候,应该得知道我们当前点击的是哪个分类列表,所以进入到视图时得拿到一个分类参数
 - 同时得知道列表页最终的样式是怎样的,然后将数据进行传递

8.2 页面展示

- 先将静态页面展示

```
# articles\urls.py文件

from django.urls import path

from articles import views

urlpatterns = [
    path('', views.index, name='index'), # 主页
    path('list/<str:cate>', views.list, name='list'), # 分类信息
]
```

```
# articles\views.py文件

def list(request, cate):
    return render(request, 'list.html')
```

```
{% load static %}

<!doctype html>
<html lang="zh-CN">
<head>
    <meta name="360-site-verification"
content="85326d9c1b0d512826605334e6eb1d5c">
    <meta charset="utf-8">
    <meta name="renderer" content="webkit">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="baidu_union_verify"
content="6c3c4420bcc5cb0d05563cc88180cd88">
    <title>Python</title>
    <meta name="keywords" content="Django博客">
    <link rel="stylesheet" type="text/css" href="{% static
'css/bootstrap.min.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static
'css/nprogress.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/style.css'
%}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/shang.css'
%}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/fontawesome.min.css' %}">
    <link rel="apple-touch-icon-precomposed" href="{% static
'images/icon.png' %}">
    <link rel="shortcut icon" href="{% static 'images/favicon.ico' %}">
    <script src="{% static 'js/jquery-2.1.4.min.js' %}"></script>
    <script src="{% static 'js/nprogress.js' %}"></script>
    <script src="{% static 'js/jquery.lazyload.min.js' %}"></script>
</head>
<body class="user-select">
<header class="header">
    <nav class="navbar navbar-default" id="navbar">
        <div class="container">
            {# 头部最上层 #}
            <div class="header-topbar hidden-xs link-border">
                <ul class="site-nav topmenu">
                    {# 如果用户是登录则显示名字#}
                    {% if request.user.is_authenticated %}<li>
```

```
                <li><a href="#">@{{ request.user.username }}</a>
            </li>
            <li><a href="#" style="border-left: 1px solid black">退出</a></li>
                {% if not request.user.is_authenticated %}
                {% else %}
                    <li><a href="{% url 'user_login' %}">登录</a></li>
                    <li><a href="{% url 'user_register' %}" style="border-left: 1px solid black">注册</a></li>
                {% endif %}
            </ul>
            人生苦短，我用Python
        </div>
        {% include "header.html" %}
        <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#header-navbar" aria-expanded="false">
                <span class="sr-only"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <h1 class="logo hvr-bounce-in">
                <a href="{% url 'index' %}" title="首页">
                    
                </a>
            </h1>
        </div>
        {% include "header.html" %}
        <div class="collapse navbar-collapse" id="header-navbar">
            <ul class="nav navbar-nav navbar-right">
                <li>
                    <a data-cont="首页" title="首页" href="{% url 'index' %}">首页
                </a>
                </li>
                <li>
                    <a data-cont="所有的分类" title="所有的分类" href="#">所有的分类
                </a>
                </li>
                <li>
                    <a data-cont="Python" title="Python" href="#">Python</a>
                </li>
                <li>
                    <a data-cont="Java" title="Java" href="#">Java</a>
                </li>
                <li>
                    <a data-cont="Java" title="Java" href="#">Java</a>
                </li>
                <li>
                    <a data-cont="Python" title="Python" href="#">Python</a>
                </li>
                <li>
                    <a data-cont="Python" title="Python" href="#">Python</a>
                </li>
            </ul>
        </div>
```

```

        </div>
    </nav>
</header>
{# 中间区域，轮播图+文章+标签等所有 #-}
<section class="container">
{# 左边部分 #-}
    <div class="content-wrap">
        {# 中间文章的区域 #-}
            <div class="content">
                <div class="title">
                    <h3 style="line-height: 1.3">{{ cate_obj.name }}主题</h3>
                </div>

                {# 所有文章 #-}
                <article class="excerpt excerpt-1" style="">
                    <a class="focus" href="#" title="Django框架"
target=_blank">
                        {# 文章图片 #-}
                        
                    </a>
                    <header>
                        {# 文章分类的名字 #-}
                        <a class="cat" href="#" title="Python"
target=_blank">Python<i></i></a>
                        {# 文章名字 #-}
                        <h2><a href="#" title="Django" target=_blank>Django框
架</a></h2>
                    </header>
                    <p class="meta">
                        {# 文章添加时间 #-}
                        <time class="time"><i class="glyphicon glyphicon-time">
</i> 2022-7-25</time>
                        {# 点击数 #-}
                        <span class="views"><i class="glyphicon glyphicon-eye-
open"></i> 0</span>
                        {# 评论数 #-}
                        <a class="comment" title="评论">
                            <i class="glyphicon glyphicon-comment"></i> 0
                        </a>
                    </p>
                    {# 文章简介 #-}
                    <p class="note">Django是一个开放源代码的web应用框架，由Python写
成。</p>
                </article>
                {# 所有文章 #-}
                <article class="excerpt excerpt-1" style="">
                    <a class="focus" href="#" title="Django框架"
target=_blank">
                        {# 文章图片 #-}
                        
                    </a>
                    <header>
                        {# 文章分类的名字 #-}

```

```
<a class="cat" href="#" title="Python"
target="_blank">Python<i></i></a>
{# 文章名字 #-}
<h2><a href="#" title="Django" target="_blank">Django框架</a></h2>
</header>
<p class="meta">
{# 文章添加时间 #-}
<time class="time"><i class="glyphicon glyphicon-time"></i> 2022-7-25</time>
{# 点击数 #-}
<span class="views"><i class="glyphicon glyphicon-eye-open"></i> 0</span>
{# 评论数 #-}
<a class="comment" title="评论">
<i class="glyphicon glyphicon-comment"></i> 0
</a>
</p>
{# 文章简介 #-}
<p class="note">Django是一个开放源代码的web应用框架，由Python写成。</p>
</article>
{# 所有文章 #-}
<article class="excerpt excerpt-1" style="">
<a class="focus" href="#" title="Django框架"
target="_blank">
{# 文章图片 #-}

</a>
<header>
{# 文章分类的名字 #-}
<a class="cat" href="#" title="Python"
target="_blank">Python<i></i></a>
{# 文章名字 #-}
<h2><a href="#" title="Django" target="_blank">Django框架</a></h2>
</header>
<p class="meta">
{# 文章添加时间 #-}
<time class="time"><i class="glyphicon glyphicon-time"></i> 2022-7-25</time>
{# 点击数 #-}
<span class="views"><i class="glyphicon glyphicon-eye-open"></i> 0</span>
{# 评论数 #-}
<a class="comment" title="评论">
<i class="glyphicon glyphicon-comment"></i> 0
</a>
</p>
{# 文章简介 #-}
<p class="note">Django是一个开放源代码的web应用框架，由Python写成。</p>
</article>
{# 所有文章 #-}
<article class="excerpt excerpt-1" style="">
```

```
<a class="focus" href="#" title="Django框架"
target=_blank>
    {# 文章图片 #-}
    
</a>
<header>
    {# 文章分类的名字 #-}
    <a class="cat" href="#" title="Python"
target=_blank>Python<i></i></a>
    {# 文章名字 #-}
    <h2><a href="#" title="Django" target=_blank>Django框
架</a></h2>
</header>
<p class="meta">
    {# 文章添加时间 #-}
    <time class="time"><i class="glyphicon glyphicon-time">
</i> 2022-7-25</time>
    {# 点击数 #-}
    <span class="views"><i class="glyphicon glyphicon-eye-
open"></i> 0</span>
    {# 评论数 #-}
    <a class="comment" title="评论">
        <i class="glyphicon glyphicon-comment"></i> 0
    </a>
</p>
    {# 文章简介 #-}
    <p class="note">Django是一个开放源代码的web应用框架，由Python写
成。</p>
</article>
    {# 所有文章 #-}
    <article class="excerpt excerpt-1" style="">
        <a class="focus" href="#" title="Django框架"
target=_blank>
            {# 文章图片 #-}
            
</a>
<header>
            {# 文章分类的名字 #-}
            <a class="cat" href="#" title="Python"
target=_blank>Python<i></i></a>
            {# 文章名字 #-}
            <h2><a href="#" title="Django" target=_blank>Django框
架</a></h2>
</header>
            <p class="meta">
                {# 文章添加时间 #-}
                <time class="time"><i class="glyphicon glyphicon-time">
</i> 2022-7-25</time>
                {# 点击数 #-}
                <span class="views"><i class="glyphicon glyphicon-eye-
open"></i> 0</span>
                {# 评论数 #-}
                <a class="comment" title="评论">
                    <i class="glyphicon glyphicon-comment"></i> 0
                </a>
            </p>
```

```
        </a>
    </p>
    {# 文章简介 #-}
    <p class="note">Django是一个开放源代码的web应用框架，由Python写
成。</p>
</article>

</div>
<div class="widget widget_sentence">
    <h3>标签云</h3>
    <div class="widget-sentence-content">
        <ul class="plinks ptags">
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
        </ul>
    </div>
</div>
{# 右边部分 #-}
<aside class="sidebar">
    <div class="fixed">
        <div class="widget widget_search">
            <form class="navbar-form" action="#">
                <div class="input-group">
                    <input type="text" name="keyword" class="form-
control" size="35" placeholder="请输入关键字" maxlength="15"
autocomplete="off">
                        <span class="input-group-btn">
                            <button class="btn btn-default btn-search"
type="submit">搜索</button>
                        </span>
                </div>
            </form>
        </div>
    {# 右下的标签云 #-}

```



```
    1  {# 文章的观看数 #}
        </span>
    </a>
</li>
<li>
    <a title="Django框架" href="#" target="_blank">
        <span class="thumbnail">
            {# 文章的图片 #}
            
        </span>
        {# 文章的名字 #}
        <span class="text">Flask框架</span>
        <span class="muted">
            <i class="glyphicon glyphicon-time"></i>
            2022-7-25 {# 文章的发布时间 #}
        </span>
        <span class="muted">
            <i class="glyphicon glyphicon-eye-open"></i>
            1 {# 文章的观看数 #}
        </span>
    </a>
</li>
<li>
    <a title="Django框架" href="#" target="_blank">
        <span class="thumbnail">
            {# 文章的图片 #}
            
        </span>
        {# 文章的名字 #}
        <span class="text">Flask框架</span>
        <span class="muted">
            <i class="glyphicon glyphicon-time"></i>
            2022-7-25 {# 文章的发布时间 #}
        </span>
        <span class="muted">
            <i class="glyphicon glyphicon-eye-open"></i>
            1 {# 文章的观看数 #}
        </span>
    </a>
</li>
<li>
    <a title="Django框架" href="#" target="_blank">
        <span class="thumbnail">
            {# 文章的图片 #}
            
        </span>
        {# 文章的名字 #}
        <span class="text">Flask框架</span>
        <span class="muted">
            <i class="glyphicon glyphicon-time"></i>
            2022-7-25 {# 文章的发布时间 #}
        </span>
        <span class="muted">
            <i class="glyphicon glyphicon-eye-open"></i>
            1 {# 文章的观看数 #}
        </span>
    </a>
</li>
<li>
```

```
<span class="muted">
    <i class="glyphicon glyphicon-eye-open"></i>
    1  {"# 文章的观看数 #}
</span>
</a>
</li>
<li>
    <a title="Django框架" href="#" target="_blank">
        <span class="thumbnail">
            {"# 文章的图片 #}
            
        </span>
        {"# 文章的名字 #}
        <span class="text">Flask框架</span>
        <span class="muted">
            <i class="glyphicon glyphicon-time"></i>
            2022-7-25 {"# 文章的发布时间 #}
        </span>
        <span class="muted">
            <i class="glyphicon glyphicon-eye-open"></i>
            1 {"# 文章的观看数 #}
        </span>
        </a>
    </li>
    <li>
        <a title="Django框架" href="#" target="_blank">
            <span class="thumbnail">
                {"# 文章的图片 #}
                
            </span>
            {"# 文章的名字 #}
            <span class="text">Flask框架</span>
            <span class="muted">
                <i class="glyphicon glyphicon-time"></i>
                2022-7-25 {"# 文章的发布时间 #}
            </span>
            <span class="muted">
                <i class="glyphicon glyphicon-eye-open"></i>
                1 {"# 文章的观看数 #}
            </span>
            </a>
        </li>
        <li>
            <a title="Django框架" href="#" target="_blank">
                <span class="thumbnail">
                    {"# 文章的图片 #}
                    
                </span>
                {"# 文章的名字 #}
                <span class="text">Flask框架</span>
                <span class="muted">
                    <i class="glyphicon glyphicon-time"></i>

```

```
2022-7-25 {# 文章的发布时间 #}
</span>
<span class="muted">
    <i class="glyphicon glyphicon-eye-open"></i>
    1 {# 文章的观看数 #}
</span>
</a>
</li>
<li>
    <a title="Django框架" href="#" target="_blank">
        <span class="thumbnail">
            {# 文章的图片 #}
            
        </span>
        {# 文章的名字 #}
        <span class="text">Flask框架</span>
        <span class="muted">
            <i class="glyphicon glyphicon-time"></i>
            2022-7-25 {# 文章的发布时间 #}
        </span>
        <span class="muted">
            <i class="glyphicon glyphicon-eye-open"></i>
            1 {# 文章的观看数 #}
        </span>
        </a>
    </li>
</ul>
</div>

{# 广告区域 #


<a href="https://www.aliyun.com" rel="nofollow" title="阿里云" target="_blank">
        
    </a>



<a href="https://cloud.tencent.com/" rel="nofollow" title="腾讯云" target="_blank">
        
    </a>



<h3>友情链接</h3>
    <div class="widget-sentence-link">
        <a href="javascript:;" title="Python">Python</a>
        <a href="javascript:;" title="Python爬虫教程">Python爬虫</a>
        <a href="javascript:;" title="Python机器学习 深度学习">Python机器学习</a>
    </div>


</div>
</div>
</aside>
{# 向上的按钮 #
<ul class="sidebar">
    <li class="totop"></li>
</ul>
```

```

</section>
{# 底部信息 #
<footer class="footer">
    <div class="container">
        <p>
            Copyright &copy; 2022.知吾所想 予吾所好-Alle
        </p>
    </div>
    <div id="gotop"><a class="gotop"></a></div>
</footer>

<script src="{% static 'js/bootstrap.min.js' %}"></script>
<script src="{% static 'js/jquery.ias.js' %}"></script>
<script src="{% static 'js/scripts.js' %}"></script>
<script>
    var _hmt = _hmt || [];
    (function () {
        var hm = document.createElement("script");
        hm.src = "https://hm.baidu.com/hm.js?
e8ae61fbc1aaa8b44823aae4cd23690b0";
        var s = document.getElementsByTagName("script")[0];
        s.parentNode.insertBefore(hm, s);
    })();
</script>

</body>
</html>

```

8.3 功能实现

- 根据观察知道, 列表页需要的数据有以下

1. 推荐分类
2. 当前分类的标签云
3. 热门推荐8个
4. 拿到当前分类下所有文章
5. 根据标签云拿到当前分类下的文章

- 调整视图功能实现

```

# articles\views.py文件
from django.shortcuts import render, redirect

def list(request, cate):
    """
    主页中根据分类显示不同的文章
    1. 推荐分类
    2. 当前分类的标签云
    3. 热门推荐8个

    4. 拿到当前分类下所有文章
    """

```

```

# 1. 根据分类时间进行升序拿所有推荐分类
all_category = Category.objects.filter(is_tab=True).order_by('add_time')
# 2. 当前分类的标签云
all_tags = TagInfo.objects.filter(category__name=cate)
# 3. 热门推荐前8篇(点击量最多的)
hot_articles = ArticleInfo.objects.order_by('-click_num')[:8]

# 根据前端传递过来的id来得到当前分类
cate = Category.objects.filter(name=cate)
if cate: # 如果分类存在，则展示数据，因为分类的文本有可能是自己在url上输入的
    # 4. 拿到当前分类下的所有文章
    all_articles = ArticleInfo.objects.filter(category_id=cate[0].id)
    # 分页
    # 获取从前面点击的页面数字，默认1
    current_page = request.GET.get('page', 1)
    # 每页有10条数据
    paginator = Paginator(all_articles, 10)
    # 生成一页中的文章
    page_obj = paginator.page(current_page)

    return render(request, 'list.html', {
        'all_category': all_category, # 1. 推荐分类
        'all_tags': all_tags, # 2. 当前分类的标签云
        'hot_articles': hot_articles, # 3. 热门推荐8个
        'page_obj': page_obj, # 4. 拿到当前分类下所有文章
    })
else:
    # 如果用户随意输入的，没有这分类，则到主页
    return redirect('/')

```

8.4 模板传递

- 使用Django模板语言，将视图当中的数据传递给到模板当中
- 同时在首页的模板中，修改分类列表的点击路由，即可实现点击分类来跳转到列表页

8.5 标签云

- 点击标签云有两种时候
 1. 首页的标签云
 - 此处的标签云应该是直接跳转到该列表下的标签云
 2. 列表页的标签云
 - 此处的标签云应该是该列表所存在的标签云
- 所以调整 `index.html` 和 `list.html` 标签云的跳转地址为如下：

```

127.0.0.1:8000/list/分类/?tag=标签id
127.0.0.1:8000/list/python/?tag=1

```

- 同时调整视图中对标签云的处理

```

# articles\views.py文件

def list(request, cate):
    """列表"""

    ...
    cate = Category.objects.filter(name=cate)
    if cate: # 如果分类存在，则展示数据，因为分类的文本有可能是自己在url上输入的
        # 拿到前端传递的标签(可能点了分类后又点了标签云)
        tag = request.GET.get('tag', '')
        if tag:
            # 如果点了标签云，则把该分类下不同的标签文章展示出来
            all_articles =
ArticleInfo.objects.filter(category_id=cate[0].id, taginfo=tag)
        else:
            # 如果没点标签云，则只展示分类的所有文章，得进行分页
            # 4. 拿到当前分类下的所有文章
            all_articles =
ArticleInfo.objects.filter(category_id=cate[0].id)
            # 分页
    ...

```

9. 模板继承

- 模板继承和类的继承含义是一样的，主要是为了提高代码重用，减轻开发人员的工作量。

9.1 父模板

- 如果发现在多个模板中某些内容相同，那就应该把这段内容定义到父模板中。
- 标签block：用于在父模板中预留区域，留给子模板填充差异性的内容，名字不能相同。为了更好的可读性，建议给endblock标签写上名字，这个名字与对应的block名字相同。父模板中也可以使用上下文中传递过来的数据。

```

{% block 名称 %}
    预留区域，可以编写默认内容，也可以没有默认内容
{% endblock 名称 %}

```

9.2 子模板

- 标签extends：继承，写在子模板文件的第一行。

```

{% extends "父模板路径"%}

```

- 子模版不用填充父模版中的所有预留区域，如果子模版没有填充，则使用父模版定义的默认值。
- 填充父模版中指定名称的预留区域。

```
{% block 名称 %}  
实际填充内容  
{% endblock 名称 %}
```

9.3 模板抽离

- 首页和列表页有很多相似的功能, 同时接下来要实现的详情页和它们也有很多相似的功能, 所以我们可以将这三个模板抽离出公共的部分
- 通过观察, 我们发现首页、列表页及详情页的上层、右侧、下标签都存在相同的部分, 所以可以将公司的部分抽离出来

9.3.1 base.html

- 在 `templates` 下创建一个 `base.html` 文件, 该文件中保存公有的部分

```
{% load static %}  
<!doctype html>  
<html lang="zh-CN">  
<head>  
    <meta name="360-site-verification"  
content="85326d9c1b0d512826605334e6eb1d5c">  
    <meta charset="utf-8">  
    <meta name="renderer" content="webkit">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <meta name="baidu_union_verify"  
content="6c3c4420bcc5cb0d05563cc88180cd88">  
    {% block title %}{% endblock %}  
    <meta name="keywords" content="Django博客">  
    <link rel="stylesheet" type="text/css" href="{% static  
'css/bootstrap.min.css' %}">  
    <link rel="stylesheet" type="text/css" href="{% static  
'css/nprogress.css' %}">  
    <link rel="stylesheet" type="text/css" href="{% static 'css/style.css'  
%}">  
    <link rel="stylesheet" type="text/css" href="{% static 'css/shang.css'  
%}">  
    <link rel="stylesheet" type="text/css" href="{% static 'css/font-  
awesome.min.css' %}">  
    <link rel="apple-touch-icon-precomposed" href="{% static  
'images/icon.png' %}">  
    <link rel="shortcut icon" href="{% static 'images/favicon.ico' %}">  
    <script src="{% static 'js/jquery-2.1.4.min.js' %}"></script>  
    <script src="{% static 'js/nprogress.js' %}"></script>  
    <script src="{% static 'js/jquery.lazyload.min.js' %}"></script>  
    {# 预留css的继承 #}  
    {% block mycss %}  
    {% endblock %}  
</head>  
<body class="user-select">  
<header class="header">
```

```

<nav class="navbar navbar-default" id="navbar">
    <div class="container">
        {# 头部最上层 #-}
        <div class="header-topbar hidden-xs link-border">
            <ul class="site-nav topmenu">
                {# 如果用户是登录则显示名字#}
                {% if request.user.is_authenticated %}
                    <li><a href="#"@{{ request.user.username }}></a>
                </li>
                <li><a href="#" style="border-left: 1px solid black">退出</a></li>
                    {# 如果未登录，则显示登录或注册 #}
                    {% else %}
                        <li><a href="{% url 'user_login' %}">登录</a></li>
                        <li><a href="{% url 'user_register' %}" style="border-left: 1px solid black">注册</a></li>
                    {% endif %}
                </ul>
                人生苦短，我用Python
            </div>
            {# 头部下面左 #-}
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#header-navbar" aria-expanded="false">
                    <span class="sr-only"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <h1 class="logo hvr-bounce-in">
                    <a href="{% url 'index' %}" title="首页">
                        
                    </a>
                </h1>
            </div>
            {# 头部下面右 #-}
            <div class="collapse navbar-collapse" id="header-navbar">
                <ul class="nav navbar-nav navbar-right">
                    <li>
                        <a data-cont="首页" title="首页" href="{% url 'index' %}">首页
                        </a>
                    </li>
                    {# 所有的分类 #}
                    {% for cate in all_category %}
                        <li>
                            <a data-cont="python" title="{{ cate.name }}" href="{% url 'list' cate.name %}">{{ cate.name }}</a>
                        </li>
                    {% endfor %}
                </ul>
            </div>
        </div>
    </nav>
</header>
{# 中间区域，轮播图+文章+标签等所有 #-}
<section class="container">

```

```
{# 左边部分 #-}


{# 中间文章的区域 #-}
{% block content %}{% endblock %}


### 标签云



{# 只拿最新的前11个 #-}
{% for tag in all_tags|slice:'10:-1' %}
- {{ tag.name }}
{{ tag.articleinfo_set.count }}


{# 右边部分 #-}


{# 统计区域，只有主页才有 #-}
{% block TongJi %}{% endblock %}



<form class="navbar-form" action="#">
    <div class="input-group">
        <input type="text" name="keyword" class="form-control" size="35" placeholder="请输入关键字" maxlength="15" autocomplete="off">
        <span class="input-group-btn">
            <button class="btn btn-default btn-search" type="submit">搜索</button>
        </span>
    </div>
</form>


{# 右下的标签云 #-}


### 标签云



{# 只将分类当中的后10个拿出来 #-}
{% for tag in all_tags|slice:'10:-1' %}
- {{ tag.name }}
{{ tag.articleinfo_set.count }}


```

```
        </div>
    </div>
</div>
{# 最新留言文章 #-}


### 热门推荐




{% for art in hot_articles %}
- {# 文章的图片 #-}
!\[{{ art.title }}\]\({% static 'media/' %}{% if art.image %}{{ MEDIA\_URL }}{{ art.image }}{% else %}1{% endif %}\)

{# 文章的名字 #-}
{{ art.title }}

</i>
{{ art.add\_time|date }} {# 文章的发布时间 #}



{{ art.click_num }} {# 文章的观看数 #}


{# 广告区域 #-}


!\[阿里云\]\({% static 'images/阿里云.jpeg' %}\)



!\[腾讯云\]\({% static 'images/腾讯云.jpg' %}\)



### 友情链接



Python官网


```

```

        </div>
    </div>
</aside>
{# 向上的按钮 #-}
<ul class="sidebar">
    <li class="totop"></li>
</ul>
</section>
{# 底部信息 #-}
<footer class="footer">
    <div class="container">
        <p>
            Copyright &copy; 2022.知吾所想 予吾所好-Alle
        </p>
    </div>
    <div id="gotop"><a class="gotop"></a></div>
</footer>

<script src="{% static 'js/bootstrap.min.js' %}"></script>
<script src="{% static 'js/jquery.ias.js' %}"></script>
<script src="{% static 'js/scripts.js' %}"></script>
<script>
    var _hmt = _hmt || [];
    (function () {
        var hm = document.createElement("script");
        hm.src = "https://hm.baidu.com/hm.js?
e8ae61fbc1aaa8b44823aae4cd23690b0";
        var s = document.getElementsByTagName("script")[0];
        s.parentNode.insertBefore(hm, s);
    })();
</script>

{# 预留js的继承 #-}
{% block myjs %}
{% endblock %}
</body>
</html>

```

9.3.2 index.html

- 调整 `index.html` 的代码

```

{% extends 'base.html' %}
{% load static %}

{% block title %}
    <title>Django博客-首页</title>
{% endblock %}

{% block content %}
    {# 中间文章的区域 #-}
    <div class="content">
        {# 轮播图 #-}
        <div id="focusslide" class="carousel slide" data-ride="carousel">
            <ol class="carousel-indicators">

```

```

                <li data-target="#focusslide" data-slide-to="0"
class="active"></li>
                <li data-target="#focusslide" data-slide-to="1"></li>
            </ol>

            <div class="carousel-inner" role="listbox">
                <div class="item active">
                    <a href="javascript:;" title="Python">
                        </a>
                    </div>
                <div class="item">
                    <a href="javascript:;" title="Scratch">
                        </a>
                    </div>
                </div>
                <a class="left carousel-control" href="#focusslide"
role="button" data-slide="prev" rel="nofollow">
                    <span class="glyphicon glyphicon-chevron-left" aria-
hidden="true"></span>
                    <span class="sr-only">上一个</span>
                </a>
                <a class="right carousel-control" href="#focusslide"
role="button" data-slide="next" rel="nofollow">
                    <span class="glyphicon glyphicon-chevron-right" aria-
hidden="true"></span>
                    <span class="sr-only">下一个</span>
                </a>
            </div>

            {# 推荐内容 #}
            {% for art in recommend_article %}
                <article class="excerpt-minic excerpt-minic-index">
                    <h2>
                        <span class="red">【推荐】</span>
                    {# 文章标签 #}
                    <a target="_blank" href="#" title="{{ art.title }}>{{
art.title }}</a>
                    </h2>
                    {# 文章简介 #}
                    <p class="note">{{ art.desc }}</p>
                </article>
            {% endfor %}

            {# 最新发布的文字，包括分类的文字 #}
            <div class="title">
                <h3>最新发布</h3>
                <div class="more">
                    {% for cate in many_category %}
                        <a href="{% url 'list' cate.category__name %}" title="{{
cate.category__name }}>{{ cate.category__name }}</a>
                    {% endfor %}
                </div>
            </div>
            {# 所有文章 #}
            {% for art in page_obj.object_list %}
                <article class="excerpt excerpt-1" style="">

```

```

        <a class="focus" href="#" title="{{ art.title }}"
target=_blank>
    {# 文章图片 #-}
    
</a>
<header>
    {# 文章分类的名字 #-}
    <a class="cat" href="#" title="{{ art.category.name }}"
target=_blank>{{ art.category.name }}<i></i></a>
    {# 文章名字 #-}
    <h2><a href="#" title="{{ art.title }}" target=_blank>
{{ art.title }}</a></h2>
</header>
<p class="meta">
    {# 文章添加时间 #-}
    <time class="time"><i class="glyphicon glyphicon-time"></i> {{ art.add_time }}</time>
    {# 点击数 #-}
    <span class="views"><i class="glyphicon glyphicon-eye-open"></i> {{ art.click_num }}</span>
    {# 留言数 #-}
    <a class="comment" title="留言">
        <i class="glyphicon glyphicon-comment"></i> {{ art.cont_num }}</a>
    </p>
    {# 文章简介 #-}
    <p class="note">{{ art.desc }}</p>
</article>
{%- empty %}
    <div class="no-post">暂无博客</div>
{%- endfor %}
    {# 分页器 #-}
    {%- include 'pagination.html' %}</div>
{%- endblock %}

{%- block TongJi %}
    {# 统计区域，只有主页才有 #-}
    <div class="widget widget-tabs">
        <ul class="nav nav-tabs" role="tablist">
            <li role="presentation" class="active"><a href="#notice" aria-controls="notice" role="tab" data-toggle="tab">统计信息</a></li>
            <li role="presentation"><a href="#contact" aria-controls="contact" role="tab" data-toggle="tab">联系站长</a></li>
        </ul>
        <div class="tab-content">
            {# 统计信息 #-}
            <div role="tabpanel" class="tab-pane contact active" id="notice">
                <h2>文章总数:</h2>
                {{ article_total }}篇
            </div>
            <h2>网站运行:</h2>
            <span id="sitetime">{{ all_day }}天 </span></h2>

```

```

        </div>
        {# 联系站长 #-}
        <div role="tabpanel" class="tab-pane contact" id="contact">
            <h2>WX:Allen
                <a href="" target="_blank" rel="nofollow" data-
                    toggle="tooltip" data-placement="bottom" title=""
                    data-original-title="QQ:853307553"></a>
            </h2>
            <h2>Email:449837498@qq.com
                <a href="#" target="_blank" data-toggle="tooltip"
                    rel="nofollow" data-placement="bottom" title=""
                    data-original-title="#"></a>
            </h2>
        </div>
    </div>
</div>
{% endblock %}

```

9.3.3 list.html

- 调整 `list.html` 的代码

```

{% extends 'base.html' %}
{% load static %}

{% block title %}
    <title>列表页-{{ cate }}</title>
{% endblock %}

{% block content %}
    {# 中间文章的区域 #-}
    <div class="content">
        <div class="title">
            <h3 style="line-height: 1.3">{{ cate_obj.name }}主题</h3>
        </div>

        {% for art in page_obj.object_list %}
            <article class="excerpt excerpt-{{ forloop.counter }}"><a
                class="focus" href="#" title="{{ art.title }}" target="_blank"></a>
                <header><a class="cat" title="{{ cate_obj.name }}">{{
                    cate_obj.name }}<i></i></a>
                    <h2><a href="#" title="{{ art.title }}">{{ art.title }}</a>
                </h2>
            </header>
            <p class="meta">
                <time class="time"><i class="glyphicon glyphicon-time">
                </i> {{ art.add_time }}</time>
                <span class="views"><i class="glyphicon glyphicon-eye-
                    open"></i>{{ art.click_num }}</span>
            </p>
        {% endfor %}
    </div>
</div>

```

```

        {# 留言数 #}
        <a class="comment" title="留言">
            <i class="glyphicon glyphicon-comment"></i> {{ art.cont_num }}
        </a>
    </p>
    <p class="note">{{ art.desc }}</p>
</article>
{% endfor %}

    {% include 'pagination.html' %}
</div>
{% endblock %}

```

- 调整完后，刷新主页和列表页，查看是否和之前页面一样

10.详情页

- 根据详情页图，能够看到该文章的内容在整个页面中间，同时，在下方可以提交留言并展示留言信息

10.1 页面展示

- 先将静态页面展示

```

# articles\urls.py文件

from django.urls import path

from articles import views

urlpatterns = [
    path('', views.index, name='index'), # 主页
    path('list/<str:cate>', views.list, name='list'), # 分类信息
    path('detail/<int:id>', views.detail, name='detail'), # 文章详情页
]

```

```

# articles\views.py文件

def detail(request, id):
    """详情页"""
    return render(request, 'detail.html')

```

- 在 `templates` 下面创建 `detail.html` 文件

```

{# detail.html文件 #-}

{% load static %}
<!doctype html>
<html lang="zh-CN">
<head>

```

```
<meta name="360-site-verification" content="85326d9c1b0d512826605334e6eb1d5c">
<meta charset="utf-8">
<meta name="renderer" content="webkit">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="baidu_union_verify" content="6c3c4420bcc5cb0d05563cc88180cd88">
<title>详情页</title>
<meta name="keywords" content="Django博客">
<link rel="stylesheet" type="text/css" href="{% static 'css/bootstrap.min.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'css/nprogress.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'css/shang.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'css/fontawesome.min.css' %}">
<link rel="apple-touch-icon-precomposed" href="{% static 'images/icon.png' %}">
<link rel="shortcut icon" href="{% static 'images/favicon.ico' %}">
<script src="{% static 'js/jquery-2.1.4.min.js' %}"></script>
<script src="{% static 'js/nprogress.js' %}"></script>
<script src="{% static 'js/jquery.lazyload.min.js' %}"></script>
</head>
<body class="user-select">
<header class="header">
<nav class="navbar navbar-default" id="navbar">
<div class="container">
{# 头部最上层 #}
<div class="header-topbar hidden-xs link-border">
<ul class="site-nav topmenu">
{# 如果用户是登录则显示名字#}
{% if request.user.is_authenticated %}
<li><a href="#" @{{ request.user.username }}></a>
</li>
<li><a href="#" style="border-left: 1px solid black">退出</a></li>
{# 如果未登录，则显示登录或注册 #}
{% else %}
<li><a href="{% url 'user_login' %}">登录</a></li>
<li><a href="{% url 'user_register' %}">注册</a></li>
{# endif #}
</ul>
人生苦短，我用Python
</div>
{# 头部下面左 #}
<div class="navbar-header">
<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#header-navbar" aria-expanded="false">
<span class="sr-only"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
</button>
```

```
<h1 class="logo hvr-bounce-in">
    <a href="{% url 'index' %}" title="首页">
        
    </a>
</h1>
</div>
{# 头部下面右 #}
<div class="collapse navbar-collapse" id="header-navbar">
    <ul class="nav navbar-nav navbar-right">
        <li>
            <a data-cont="首页" title="首页" href="{% url 'index'
%}">首页
            </a>
        </li>
        {# 所有的分类 #}
        <li>
            <a data-cont="Python" title="Python"
href="#">Python</a>
        </li>
        <li>
            <a data-cont="Java" title="Java" href="#">Java</a>
        </li>
        <li>
            <a data-cont="Java" title="Java" href="#">Java</a>
        </li>
        <li>
            <a data-cont="Python" title="Python"
href="#">Python</a>
        </li>
        <li>
            <a data-cont="Python" title="Python"
href="#">Python</a>
        </li>
        </ul>
    </div>
</nav>
</header>
{# 中间区域， 轮播图+文章+标签等所有 #}
<section class="container">
    {# 左边部分 #}
    <div class="content-wrap">
        {# 中间文章的区域 #}
        <div class="content">
            <header class="article-header">
                <h1 class="article-title">
                    {# 文章标题 #}
                    <a href="#" title="Django">Django</a>
                </h1>
                <div class="article-meta">
                    <span class="item article-meta-time">
                        {# 发布时间 #}
                        <time class="time" data-toggle="tooltip" data-
placement="bottom" title="" data-original-title="发表时间: 2022-7-25">
                            <i class="glyphicon glyphicon-time"></i> 2022-7-25
                        </time>
                    </span>
                </div>
            </header>
            <div class="article-content">
                {# 文章内容 #}
                <p>这篇文章主要介绍了 Django 的基本框架和一些常见问题。Django 是一个开源的 Python Web 框架，它提供了许多强大的功能，如自动化的模型管理、视图处理、模板引擎、数据库抽象层等。通过使用 Django，开发者可以快速地构建复杂的 Web 应用程序，同时保持代码的可读性和可维护性。本文将简要介绍 Django 的核心概念，并通过一个简单的示例来演示如何使用 Django 来开发一个博客系统。
            </div>
        </div>
    </div>
</section>
```

```

        <span class="item article-meta-category" data-
toggle="tooltip" data-placement="bottom" title="" data-original-
title="Python">
            <i class="glyphicon glyphicon-list"></i>
            {# 分类 #-}
            <a href="#" title="Python">Python</a>
        </span>
        <span class="item article-meta-views" data-
toggle="tooltip" data-placement="bottom" title="" data-original-title="浏览
量">
            {# 点击数 #-}
            <i class="glyphicon glyphicon-eye-open"></i> 1
        </span>
        <span class="item article-meta-comment" data-
toggle="tooltip" data-placement="bottom" title="" data-original-title="留言
量">
            {# 留言数 #-}
            <i class="glyphicon glyphicon-comment"></i> 1
        </span>
    </div>
</header>
<article class="article-content">
    <p>
        
    </p>
    <p style="white-space: normal;">
        
    </p>
    <p>
        {# 文章内容 #-}
        <p>Flask是一个使用 <a
href="https://baike.baidu.com/item/Python" target="_blank">Python</a>&nbsp;
编写的轻量级 web 应用框架。其&nbsp;<a
href="https://baike.baidu.com/item/WSGI"
target="_blank">WSGI</a>&nbsp;工具箱采用 werkzeug , <a
href="https://baike.baidu.com/item/%E6%A8%A1%E6%9D%BF%E5%BC%95%E6%93%8E/907
667" target="_blank">模板引擎</a>则使用 jinja2 。Flask使用 BSD 授权。</p>
        <p>Flask也被称为 &ldquo;microframework&rdquo; , 因为它使用简单
的核心, 用 extension 增加其他功能。Flask没有默认使用的数据库、窗体验证工具。<sup>&nbsp;
[1]</sup><a name="ref_[1]_9910417">&nbsp;</a>
    </p>
        <p>Flask是一个轻量级的可定制框架, 使用Python语言编写, 较其他同类型框
架更为灵活、轻便、安全且容易上手。它可以很好地结合<a
href="https://baike.baidu.com/item/MVC%E6%A8%A1%E5%BC%8F/713147"
target="_blank">MVC模式</a>进行开发, 开发人员分工合作, 小型团队在短时间内就可以完成功能
丰富的中小型网站或<a
href="https://baike.baidu.com/item/web%E6%9C%8D%E5%8A%A1/2837593"
target="_blank">web服务</a>的实现。另外, Flask还有很强的定制性, 用户可以根据自己的需求
来添加相应功能, 在保持核心功能简单的同时实现功能的丰富与扩展, 其强大的插件库可以让用户实
现个性化的网站定制, 开发出功能强大的网站。
    </p>

```

```
<pre class="prettyprint lang-cs"></pre>
<script>
    window._bd_share_config = {
        "common": {
            "bdSnsKey": {},
            "bdText": "",
            "bdMini": "2",
            "bdMiniList": false,
            "bdPic": "",
            "bdStyle": "1",
            "bdSize": "32"
        }, "share": {}
    };
    with (document) 0[(getElementsByTagName('head')[0] ||
body).appendChild(createElement('script')).src =
'http://bdimg.share.baidu.com/static/api/js/share.js?v=0.js?cdnversion=' + ~
(-new Date() / 36e5)];
</script>
<p style="white-space: normal;"></p>
<p>请随手给个star, 谢谢! </p>
</p>
</article>

<div class="article-shang">
    <p>
        <a href="javascript:void(0)" rel="nofollow"
style="color: #fff" onclick="dashangToggle()" class="dashang" title="如文章有
帮助到你, 支持一下"
            draggable="false">打赏</a>
    </p>
</div>
<div class="hide_box" style="display: none;"></div>
<div class="shang_box" style="display: none;">
    <a class="shang_close" href="javascript:void(0)"
onclick="dashangToggle()" title="关闭" draggable="false">
        
    </a>
    
    <div class="shang_tit">
        <p>
            感谢您的支持, 我会继续努力的!</p>
    </div>
    <div class="shang_payimg">
        
    </div>
    <div class="pay_explain">
        扫码打赏, 您说多少就多少
    </div>
    <div class="shang_payselect">
        
    </div>
    <div class="shang_info">
        <p>
```

打开微信扫一扫，即可进行
扫码打赏哦

```
</p>
<p>
    分享从这里开始，精彩与您同在
</p>
</div>
</div>
<script>
    function dashangToggle() {
        console.log(1)
        $('.hide_box').fadeToggle();
        $('.shang_box').fadeToggle();
    }
</script>
<a href="javascript:;" onclick="repeat()"></a>
{# 留言的内容 #}
<div id="postcomments" class="addcomment">
<ol id="comment_list" class="commentlist">
    <li class="comment-content">
        <span class="comment-f">#1楼 ( admin ) </span>
        <div class="comment-main">
            <p>
                <a class="address" href="#" rel="nofollow"
target=_blank>admin</a>
                <span class="time">(2022-7-25)</span><br>
                测试留言
            </p>
        </div>
    </li>
    <li class="comment-content">
        <span class="comment-f">#2楼 ( admin ) </span>
        <div class="comment-main">
            <p>
                <a class="address" href="#" rel="nofollow"
target=_blank>admin</a>
                <span class="time">(2022-7-25)</span><br>
                测试留言
            </p>
        </div>
    </li>
    <li class="comment-content">
        <span class="comment-f">#3楼 ( admin ) </span>
        <div class="comment-main">
            <p>
                <a class="address" href="#" rel="nofollow"
target=_blank>admin</a>
                <span class="time">(2022-7-25)</span><br>
                测试留言
            </p>
        </div>
    </li>
</ol>
</div>

<div class="title" id="comment">
    <h3>留言</h3>
</div>
```

```
<div id="respond">
    <div class="comment">
        <div class="comment-box">
            <textarea placeholder="您的留言或留言（必填）"
name="comment_textarea" id="comment-textarea" cols="100%" rows="3"
tabindex="3"></textarea>
            <div class="comment-ctrl">
                <div class="comment-prompt" style="display:
none;"><i class="fa fa-spin fa-circle-o-notch"></i>
                    <span class="comment-prompt-text">留言正在提交
中...请稍后</span>
                </div>
                <div class="comment-success" style="display:
none;">
                    <i class="fa fa-check"></i>
                    <span class="comment-prompt-text">留言提交成
功...</span>
                </div>
                <button type="submit" name="comment-submit"
id="comment-submit" tabindex="4">留言</button>
            </div>
        </div>
    </div>
<div class="widget widget_sentence">
    <h3>标签云</h3>
    <div class="widget-sentence-content">
        <ul class="plinks ptags">
            {# 只拿最新的前11个 #}
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
            <li>
                <a href="#" title="Django" draggable="false">Django
                    <span class="badge">1</span>
                </a>
            </li>
        </ul>
    </div>
</div>
```

```
</div>
{# 右边部分 #-}
<aside class="sidebar">
    <div class="fixed">

        <div class="widget widget_search">
            <form class="navbar-form" action="#">
                <div class="input-group">
                    <input type="text" name="keyword" class="form-control" size="35" placeholder="请输入关键字" maxlength="15" autocomplete="off">
                    <span class="input-group-btn">
                        <button class="btn btn-default btn-search" type="submit">搜索</button>
                    </span>
                </div>
            </form>
        </div>
        {# 右下的标签云 #-}
        <div class="widget widget_sentence">
            <h3>标签云</h3>
            <div class="widget-sentence-content">
                <ul class="plinks ptags">
                    {# 只将分类当中的后10个拿出来 #-}
                    <li>
                        <a href="#" title="Django" draggable="false">Django
                            <span class="badge">1</span>
                        </a>
                    </li>
                    <li>
                        <a href="#" title="Django" draggable="false">Django
                            <span class="badge">1</span>
                        </a>
                    </li>
                    <li>
                        <a href="#" title="Django" draggable="false">Django
                            <span class="badge">1</span>
                        </a>
                    </li>
                    <li>
                        <a href="#" title="Django" draggable="false">Django
                            <span class="badge">1</span>
                        </a>
                    </li>
                    <li>
                        <a href="#" title="Django" draggable="false">Django
                            <span class="badge">1</span>
                        </a>
                    </li>
                    <li>
                        <a href="#" title="Django" draggable="false">Django
                            <span class="badge">1</span>
                        </a>
                    </li>
                </ul>
            </div>
        </div>
    </div>
</div>
```

```
{# 最新留言文章 #}
<div class="widget widget_hot">
    <h3>热门推荐</h3>
    <ul>
        <li>
            <a title="Django框架" href="#" target="_blank">
                <span class="thumbnail">
                    {# 文章的图片 #}
                    
                </span>
                {# 文章的名字 #}
                <span class="text">Flask框架</span>
                <span class="muted">
                    <i class="glyphicon glyphicon-time"></i>
                    2022-7-25 {# 文章的发布时间 #}
                </span>
                <span class="muted">
                    <i class="glyphicon glyphicon-eye-open"></i>
                    1 {# 文章的观看数 #}
                </span>
            </a>
        </li>
        <li>
            <a title="Django框架" href="#" target="_blank">
                <span class="thumbnail">
                    {# 文章的图片 #}
                    
                </span>
                {# 文章的名字 #}
                <span class="text">Flask框架</span>
                <span class="muted">
                    <i class="glyphicon glyphicon-time"></i>
                    2022-7-25 {# 文章的发布时间 #}
                </span>
                <span class="muted">
                    <i class="glyphicon glyphicon-eye-open"></i>
                    1 {# 文章的观看数 #}
                </span>
            </a>
        </li>
        <li>
            <a title="Django框架" href="#" target="_blank">
                <span class="thumbnail">
                    {# 文章的图片 #}
                    
                </span>
                {# 文章的名字 #}
                <span class="text">Flask框架</span>
                <span class="muted">
                    <i class="glyphicon glyphicon-time"></i>
                    2022-7-25 {# 文章的发布时间 #}
                </span>
                <span class="muted">
                    <i class="glyphicon glyphicon-eye-open"></i>
                    1 {# 文章的观看数 #}
                </span>
            </a>
        </li>
    </ul>

```

```
<span class="muted">
    <i class="glyphicon glyphicon-eye-open"></i>
    1  {# 文章的观看数 #}
</span>
</a>
</li>
<li>
    <a title="Django框架" href="#" target="_blank">
        <span class="thumbnail">
            {# 文章的图片 #}
            
        </span>
        {# 文章的名字 #}
        <span class="text">Flask框架</span>
        <span class="muted">
            <i class="glyphicon glyphicon-time"></i>
            2022-7-25 {# 文章的发布时间 #}
        </span>
        <span class="muted">
            <i class="glyphicon glyphicon-eye-open"></i>
            1 {# 文章的观看数 #}
        </span>
    </a>
</li>
<li>
    <a title="Django框架" href="#" target="_blank">
        <span class="thumbnail">
            {# 文章的图片 #}
            
        </span>
        {# 文章的名字 #}
        <span class="text">Flask框架</span>
        <span class="muted">
            <i class="glyphicon glyphicon-time"></i>
            2022-7-25 {# 文章的发布时间 #}
        </span>
        <span class="muted">
            <i class="glyphicon glyphicon-eye-open"></i>
            1 {# 文章的观看数 #}
        </span>
    </a>
</li>
</ul>
</div>

{# 广告区域 #}
<div class="widget widget_sentence">
    <a href="https://www.aliyun.com" rel="nofollow" title="阿里云"
target=_blank">
        </a>
    </div>
    <div class="widget widget_sentence">
```

```
        <a href="https://cloud.tencent.com/" rel="nofollow" title="腾讯
云" target="_blank">
            </a>
        </div>
        <div class="widget widget_sentence">
            <h3>友情链接</h3>
            <div class="widget-sentence-link">
                <a href="https://www.python.org/" title="Python">Python官网
            </a>
                <a href="https://docs.djangoproject.com/zh-hans/3.2/" title="Django官网">Django官网</a>
                <a href="https://github.com/" title="github">github</a>
            </div>
        </div>
    </aside>
    {# 向上的按钮 #}
    <ul class="sidebar">
        <li class="totop"></li>
    </ul>
</section>
{# 底部信息 #}
<footer class="footer">
    <div class="container">
        <p>
            Copyright © 2022.知吾所想 予吾所好-Alen
        </p>
    </div>
    <div id="gotop"><a class="gotop"></a></div>
</footer>

<script src="{% static 'js/bootstrap.min.js' %}"></script>
<script src="{% static 'js/jquery.ias.js' %}"></script>
<script src="{% static 'js/scripts.js' %}"></script>
<script>
    var _hmt = _hmt || [];
    (function () {
        var hm = document.createElement("script");
        hm.src = "https://hm.baidu.com/hm.js?
e8ae61fbc1aa8b44823aae4cd23690b0";
        var s = document.getElementsByTagName("script")[0];
        s.parentNode.insertBefore(hm, s);
    })();
</script>
<script>
    $(function replyBox(box) {
        $('#comment-submit').click(function () {
            var nick_name = $('#nick_name').val();
            var content = $('#comment-textarea').val();
            var div = document.createElement('div');
            div.className = 'addcomment';
            $.ajax({
                type: 'POST',
                url: '#',
                data: {
                    csrfmiddlewaretoken: '{{ csrf_token }}',
                    nick_name: nick_name,
                    content: content,
                }
            })
        })
    })
</script>
```

```

        },
        success: function (cb) {
            if (cb.status == 'ok') {
                window.location.reload();
            } else {
                alert(cb.msg)
            }
        }
    });
});

</script>
</body>
</html>

```

10.2 模板继承

- 将 `detail.html` 文件，继承至 `base.html`

```

{# detail.html文件 #}
{% extends 'base.html' %}
{% load static %}

{% block content %}
    {# 中间文章的区域 #}
    <div class="content">
        <header class="article-header">
            <h1 class="article-title">
                {# 文章标题 #}
                <a href="#" title="Django">Django</a>
            </h1>
            <div class="article-meta">
                <span class="item article-meta-time">
                    {# 发布时间 #}
                    <time class="time" data-toggle="tooltip" data-
placement="bottom" title="" data-original-title="发表时间: 2022-7-25">
                        <i class="glyphicon glyphicon-time"></i> 2022-7-25
                    </time>
                </span>
                <span class="item article-meta-category" data-
toggle="tooltip" data-placement="bottom" title="" data-original-
title="Python">
                    <i class="glyphicon glyphicon-list"></i>
                    {# 分类 #}
                    <a href="#" title="Python">Python</a>
                </span>
                <span class="item article-meta-views" data-
toggle="tooltip" data-placement="bottom" title="" data-original-title="浏览
量">
                    {# 点击数 #}
                    <i class="glyphicon glyphicon-eye-open"></i> 1
                </span>
            </div>
        </header>
        <div class="article-content">
            {# 文章内容 #}
            <p>Django是一个开源的Python Web框架，由视图、模型、模板三部分组成。它遵循Django哲学，强调“不要让你自己写的代码占了太多的比例”。Django的视图层使用了MVT（Model-View-Template）模式，使得开发者可以专注于业务逻辑而不用担心视图的实现。Django的模型层提供了ORM（Object-Relational Mapping）功能，使得开发者可以方便地操作数据库。Django的模板层则提供了强大的模板引擎，使得开发者可以轻松地生成HTML输出。
    </div>
</div>

```

```

        <span class="item article-meta-comment" data-
toggle="tooltip" data-placement="bottom" title="" data-original-title="留言
量">
            {# 留言数 #}
            <i class="glyphicon glyphicon-comment"></i> 1
        </span>
    </div>
</header>
<article class="article-content">
    <p>
        
    </p>
    <p style="white-space: normal;">
        
    </p>
    <p>
        {# 文章内容 #}
        <p>Flask是一个使用 <a href="https://baike.baidu.com/item/Python" target="_blank">Python</a>&nbsp;编写的轻量级 web 应用框架。其 <a href="https://baike.baidu.com/item/wsgi" target="_blank">wsgi</a>&nbsp;工具箱采用 werkzeug , <a href="https://baike.baidu.com/item/%E6%A8%A1%E6%9D%BF%E5%BC%95%E6%93%8E/907667" target="_blank">模板引擎</a>则使用 jinja2 。Flask使用 BSD 授权。</p>
        <p>Flask也被称为 &ldquo;microframework&rdquo; , 因为它使用简单的核心, 用 extension 增加其他功能。Flask没有默认使用的数据库、窗体验证工具。<sup>&nbsp;[1]</sup><a name="ref_[1]_9910417">&nbsp;</a>
    </p>

        <p>Flask是一个轻量级的可定制框架, 使用Python语言编写, 较其他同类型框架更为灵活、轻便、安全且容易上手。它可以很好地结合<a href="https://baike.baidu.com/item/MVC%E6%A8%A1%E5%BC%8F/713147" target="_blank">MVC模式</a>进行开发, 开发人员分工合作, 小型团队在短时间内就可以完成功能丰富的中小型网站或<a href="https://baike.baidu.com/item/web%E6%9C%8D%E5%8A%A1/2837593" target="_blank">web服务</a>的实现。另外, Flask还有很强的定制性, 用户可以根据自己的需求来添加相应功能, 在保持核心功能简单的同时实现功能的丰富与扩展, 其强大的插件库可以让用户实现个性化的网站定制, 开发出功能强大的网站。
    </p>

    <pre class="prettyprint lang-cs"></pre>
    <script>
        window._bd_share_config = {
            "common": {
                "bdSnsKey": {},
                "bdText": "",
                "bdMini": "2",
                "bdMiniList": false,
                "bdPic": "",
                "bdStyle": "1",
                "bdSize": "32"
            }, "share": {}
        };
    </script>

```

```
        with (document) 0[(getElementsByTagName('head')[0] ||
body).appendChild(createElement('script')).src =
'http://bdimg.share.baidu.com/static/api/js/share.js?v=0.js?cdnversion=' + ~
(-new Date() / 36e5)];
    </script>
    <p style="white-space: normal;"></p>
    <p>请随手给个star, 谢谢! </p>
    </p>
</article>

<div class="article-shang">
    <p>
        <a href="javascript:void(0)" rel="nofollow" style="color:
#fff" onclick="dashangToggle()" class="dashang" title="如文章有帮助到你, 支持一下"
            draggable="false">打赏</a>
    </p>
</div>
<div class="hide_box" style="display: none;"></div>
<div class="shang_box" style="display: none;">
    <a class="shang_close" href="javascript:void(0)"
onclick="dashangToggle()" title="关闭" draggable="false">
        
    </a>
    
    <div class="shang_tit">
        <p>
            感谢您的支持, 我会继续努力的!</p>
    </div>
    <div class="shang_payimg">
        
    </div>
    <div class="pay_explain">
        扫码打赏, 您说多少就多少
    </div>
    <div class="shang_payselect">
        
    </div>
    <div class="shang_info">
        <p>
            打开<span id="shang_pay_txt">微信</span>扫一扫, 即可进行扫码打
赏哦
        </p>
        <p>
            分享从这里开始, 精彩与您同在
        </p>
    </div>
</div>
<script>
    function dashangToggle() {
        console.log(1)
        $('.hide_box').fadeToggle();
        $('.shang_box').fadeToggle();
    }
</script>
```

```

        }
    </script>
    <a href="javascript:;" onclick="repeat()"></a>
    {# 留言的内容 #}
    <div id="postcomments" class="addcomment">
        <ol id="comment_list" class="commentlist">
            {% for com in comment_list %}
                <li class="comment-content">
                    <span class="comment-f">#{{ forloop.counter }}楼 ({{ com.comment_user }}) </span>
                    <div class="comment-main">
                        <p>
                            <a class="address" href="#" rel="nofollow"
                                target="_blank">{{ com.comment_man.nick_name }}</a>
                            <span class="time">({{ com.add_time }})</span>
                        <br>
                            {{ com.content }}
                        </p>
                    </div>
                </li>
            {% endfor %}
        </ol>
    </div>

    <div class="title" id="comment">
        <h3>留言</h3>
    </div>
    <div id="respond">
        <div class="comment">
            <div class="comment-box">
                <textarea placeholder="您的留言或留言（必填）"
                    name="comment_textarea" id="comment-textarea" cols="100%" rows="3"
                    tabindex="3"></textarea>
                <div class="comment-ctrl">
                    <div class="comment-prompt" style="display: none;"><i
                        class="fa fa-spin fa-circle-o-notch"></i>
                    <span class="comment-prompt-text">留言正在提交中...
                        请稍后</span>
                    </div>
                    <div class="comment-success" style="display: none;">
                        <i class="fa fa-check"></i>
                        <span class="comment-prompt-text">留言提交成功...
                    </div>
                </div>
                <button type="submit" name="comment-submit"
                    id="comment-submit" tabindex="4">留言</button>
            </div>
        </div>
    </div>
    {% endblock %}

    {% block myjs %}
    <script>
        $(function replyBox(box) {
            $('#comment-submit').click(function () {
                var nick_name = $('#nick_name').val();

```

```

        var content = $('#comment-textarea').val();
        var div = document.createElement('div');
        div.className = 'addcomment';
        $.ajax({
            type: 'POST',
            url: '#',
            data: {
                csrfmiddlewaretoken: '{{ csrf_token }}',
                nick_name: nick_name,
                content: content,
            },
            success: function (cb) {
                if (cb.status == 'ok') {
                    window.location.reload();
                } else {
                    alert(cb.msg)
                }
            }
        });
    });
</script>
{% endblock %}

```

10.3 功能实现

- 通过模板继承已经能够将详情页的内容展示出来,但是都只是静态的,接下来将详情页中真正要展示的数据获取到,然后传递到详情页即可
- 根据观察知道,列表页需要的数据有以下

1. 推荐分类
 2. 所有标签云
 3. 热门推荐8个
- # 4. 拿当前文章对应的留言
5. 对当前文章阅读+1

10.3.1 调整视图

- 调整视图文件,将数据获取到并传递模型

```

from django.shortcuts import get_object_or_404
from comments.models import Comment

```

```

def detail(request, id):
    """
    详情页
    1. 推荐分类
    2. 所有标签云
    3. 热门推荐8个
    # 4. 拿当前文章对应的留言
    5. 对当前文章阅读+1
    """

    # 1. 推荐分类
    all_category = Category.objects.filter(is_tab=True).order_by('add_time')
    # 2. 所有的标签云
    all_tags = TagInfo.objects.all()
    # 3. 热门推荐前8篇(点击量最多的)
    hot_articles = ArticleInfo.objects.order_by('-click_num')[:8]
    # 4. 拿当前文章对应的留言
    # comment_list = Comment.objects.filter(comment_article_id=id)

    art_obj = get_object_or_404(ArticleInfo, id=id)
    # 5. 对当前文章阅读 + 1
    art_obj.click_num += 1
    art_obj.save()

    return render(request, 'detail.html', {
        'all_category': all_category,
        'all_tags': all_tags,
        'hot_articles': hot_articles,
        'art_obj': art_obj,
        # 'comment_list': comment_list,
    })

```

10.2 模板传递

- 使用Django模板语言, 将视图当中的数据传递给到模板当中
- 同时在首页和列表页的模板中, 修改详情页的点击路由, 即可实现点击文章来跳转到详情页

11. 留言

- 文章的详情页已经能够展示了, 接下来对留言功能进行完善

11.1 创建留言app, 并注册

- 打开pycharm的Terminal使用命令创建app

```
python manage.py startapp comments
```

- 注册comments

```
31     INSTALLED_APPS = [
32         'django.contrib.admin',
33         'django.contrib.auth',
34         'django.contrib.contenttypes',
35         'django.contrib.sessions',
36         'django.contrib.messages',
37         'django.contrib.staticfiles',
38         'users.apps.UsersConfig', # 注册用户app
39         'articles.apps.ArticlesConfig', # 注册文章app
40         'comments.apps.CommentsConfig', # 注册留言app|  
41         'ckeditor', # 富文本编辑器
42         'ckeditor_uploader', # 富文本编辑器上传图片模块
43     ]
```

CSDN @__Mr徐

- 创建子路由, 将并其配置到总路由中

```
# comments\urls.py文件

from django.urls import path

urlpatterns = [


from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('users.urls')), # 导入users路由
    path('', include('articles.urls')), # 导入articles路由
    path('', include('comments.urls')), # 导入comments路由

    path('ckeditor', include('ckeditor_uploader.urls')), # 富文本编辑器
]
```

11.2 创建模型类

- 每篇文章下面都会有留言, 所以得构建一个留言的模型

```
from datetime import datetime

from django.db import models

from articles.models import ArticleInfo
from users.models import UserProfile
```

```

class Comment(models.Model):
    """所有的留言"""
    comment_user = models.ForeignKey(UserProfile, verbose_name="评论人",
                                     on_delete=models.DO_NOTHING)
    comment_article = models.ForeignKey(ArticleInfo, verbose_name="评论文章",
                                         on_delete=models.DO_NOTHING)
    content = models.CharField(max_length=300, verbose_name="评论内容")
    add_time = models.DateTimeField(default=datetime.now, verbose_name="评论时间")

    def __str__(self):
        return self.comment_user.email

    class Meta:
        db_table = 'Comment'
        verbose_name = '用户评论信息表'
        verbose_name_plural = verbose_name

```

- Terminal中输入命令来生成迁移文件

```

python manage.py makemigrations
python manage.py migrate

```

11.3 留言处理

- 对于详情页, 可以编写留言, 当我们点击提交时, 应该将留言数据保存起来, 所以接下来对整个过程实现功能
 1. 修改详情页模板的留言提交功能
 2. 编写留言app子路由
 3. 获取留言数据保存时使用表单来处理
 4. 编写留言的视图函数
 5. 修改详情页留言展示数据

1. 打开 `detail.html` 文件, 在最下面的js代码中修改urls对应的路由

```

url: '{% url 'comment' art_obj.id %}',

```

2. 打开 `comments\urls.py` 文件, 编写路由

```

from django.urls import path

from comments import views

urlpatterns = [
    path('comment/<int:id>', views.comment, name='comment'), # 提交留言
]

```

3. 在 `comments` app中创建 `forms` 表单, 来对留言数据校验

```
from django import forms

class CommentForm(forms.Form):
    """留言的表单"""
    content = forms.CharField(min_length=3, max_length=200)
```

4. 打开 `comments\views.py` 文件编写视图

```
from django.http import JsonResponse

from .models import Comment, ArticleInfo
from .forms import CommentForm

def comment(request, id):
    """
    评论信息
    1. 拿到前端的评论
    2. 将评论保存到数据库中去
    """

    # 判断当前用户是否登录
    if request.user.is_authenticated:
        # 1. 拿到前端发送过来的数据通过表单进行校验
        comment_form = CommentForm(request.POST)
        if comment_form.is_valid():
            # 如果校验成功则拿数据并且保存到数据库
            # 评论内容
            content = comment_form.cleaned_data.get('content')
            # 评论人
            user = request.user
            # 2. 对评论的文章数据进行保存
            article = ArticleInfo.objects.get(id=id)
            article.cont_num += 1
            article.save()
            # 对留言数据进行保存
            comment = Comment()
            comment.comment_user = user
            comment.comment_article = article
            comment.content = content
            comment.save()

            return JsonResponse({'status': 'ok', 'msg': '评论成功',
'content': content})
        return JsonResponse({'status': 'false', 'msg': '内容长度不符'})
    return JsonResponse({'status': 'false', 'msg': '请登录再评论'})
```

5. 打开 `articles\views.py` 获取留言并传递给模板文件, 同时打开 `detail.html` 文件, 展示出留言信息

```
{# detail.html文件 #-}
{# 留言的内容 #}
```

```

<div id="postcomments" class="addcomment">
    <ol id="comment_list" class="commentlist">
        {% for com in comment_list %}
            <li class="comment-content">
                <span class="comment-f">#{{ forloop.counter }}楼 ({{ com.comment_user }}) </span>
                <div class="comment-main">
                    <p>
                        <a class="address" href="#" rel="nofollow" target="_blank">{{ com.comment_man.nick_name }}</a>
                        <span class="time">({{ com.add_time }})</span>
                    <br>
                    {{ com.content }}
                </p>
            </div>
        </li>
    {% endfor %}
    </ol>
</div>

```

12.搜索

- 所有文章已经能够展示出来了,但是如果文章一旦多的情况,我想要找到某一篇就比较麻烦,我们可以通过搜索功能来快速找到我们想要的文章
- 输出文字后可以搜索文章,该页面和列表页面非常相似,我们可以直接继承 `base.html` 来展示文章
- 搜索功能是对文章操作,所以可以直接将该功能放到 `articles` app里面

12.1 编写路由

- 在 `articles` 中添加路由

```

# articles\urls.py文件

from django.urls import path

from articles import views

urlpatterns = [
    path('', views.index, name='index'), # 主页
    path('list/<str:cate>', views.list, name='list'), # 分类信息
    path('detail/<int:id>', views.detail, name='detail'), # 文章详情页
    path('search', views.search, name='search'), # 搜索功能
]

```

12.2 编写视图

- 通过视图来实现搜索功能

```

def search(request):
    """
    查询功能
    1. 拿到前端查询的关键字(有文字和没有文字)
    2. 有文字(有文章和没有文章)
        2.1 有文章
            2.1.1 从文章中查询出当前关键字的文章
            2.1.2 推荐分类(在所有模板当中都会用， 所以放到最前端)
            2.1.3 所有标签云
            2.1.4 热门文章
        2.2 没有文章则通过404.html展示无
    3. 没有文字则通过404.html展示无
    """

# 2.1.2 根据分类时间进行升序拿所有推荐分类
all_category = Category.objects.filter(is_tab=True).order_by('add_time')

# 1. 拿到前端查询的关键字
keyword = request.GET.get('keyword', '')

# 2. 有文字(有文章和没有文章)
if keyword:
    # 2.1.1 从文章中查询出当前关键字的文章，icontains不区分大小写
    all_articles =
ArticleInfo.objects.filter(Q(title_icontains=keyword) |
Q(desc_icontains=keyword)).all()

    if all_articles: # 有文章
        # 2.1.3 所有的标签云
        all_tags = TagInfo.objects.all()
        # 2.1.4 热门推荐前8篇(点击量最多的)
        hot_articles = ArticleInfo.objects.order_by('-click_num')[:8]

        # 分页
        # 获取从前面点击的页面数字， 默认1
        current_page = request.GET.get('page', 1)
        # 每页有5条数据
        paginator = Paginator(all_articles, 5)
        # 生成一页中的文章
        page_obj = paginator.page(current_page)

        return render(request, 'search_list.html', {
            'all_category': all_category, # 推荐分类
            'all_tags': all_tags, # 所有标签
            'hot_articles': hot_articles, # 热门文章
            'page_obj': page_obj, # 分页
            'keyword': keyword # 搜索的内容
        })
    else:
        return render(request, '404.html', {'all_category':
all_category})
    else:
        return render(request, '404.html', {'all_category': all_category})

```

12.3 编写模板

- 通过模板来将视图传递的数据展示出来

```

{% extends 'base.html' %}
{% load static %}

{% block title %}
    <title>搜索内容: {{ keyword }}</title>
{% endblock %}

{% block content %}
    <div class="content">
        <div class="title">
            <h3 style="line-height: 1.3">搜索到相关{{ cont }}内容如下: </h3>
        </div>

        {% for art in page_obj.object_list %}
            <article class="excerpt excerpt-{{ forloop.counter }}"><a
class="focus" href="{% url 'detail' art.id %}" title="{{ art.title }}"
target="_blank"></a>
                <header><a class="cat" href="{% url 'list' art.category.name
%}" title="{{ art.category.name }}">{{ art.category.name }}<i></i></a>
                    <h2><a href="{% url 'detail' art.id %}" title="{{
art.title }}"
                        target="_blank">{{ art.title }}</a></h2>
                </header>
                <p class="meta">
                    <time class="time"><i class="glyphicon glyphicon-time">
</i> {{ art.add_time }}</time>
                    <span class="views"><i class="glyphicon glyphicon-eye-
open"></i>{{ art.click_num }}</span>
                    {# 评论数 #
                    <a class="comment" title="评论">
                        <i class="glyphicon glyphicon-comment"></i> {{
art.cont_num }}
                    </a>
                </p>
                <p class="note">{{ art.desc }}</p>
            </article>
        {% endfor %}
        {% include 'pagination.html' %}
    </div>
{% endblock %}

```

12.4 调整搜索按钮

- 调整 `base.html` 里面的搜索按钮

```
<form class="navbar-form" action="{% url 'search' %}>
```

