## 简介

开发文档主要包含的功能是注册、登陆（绑定邮箱）、查看、修改个人信息，查看、修改、删除自己所发的帖子。在个人中心中也有我的推荐，日常推荐一些活动，若学生感兴趣可以点击查看详情。除此之外还有联系我们页面，包含了我们的联系方式。

User模块的后端代码储存在apps\user文件夹下，前端代码储存在templates\users文件夹下。

## 模块

django的数据库模型在models.py 中构建，本模块的模型有两个，是有关用户的类UserProfile和有关登陆注册认证邮箱的类EmailVerifyRecord。如下所示：

class UserProfile(AbstractUser):

gender\_choices = (

('male','男'),

('female','女'))

nick\_name = models.CharField('昵称',max\_length=50,default='')

birthday = models.DateField('生日',null=True,blank=True)

gender = models. CharField ('性别',max\_length=10,choices=gender\_choices,default='female')

adress = models. CharField ('地址',max\_length=100,default='')

mobile = models.CharField('手机号', max\_length=11,null= True,blank=True)

image = models.I mageField(upload\_to ='image/%Y%m', default='image/default.png', max\_length =100)

class Meta:

verbose\_name = '用户信息'

verbose\_name\_plural = verbose\_name

def \_\_str\_\_(self):

return self.username

class EmailVerifyRecord(models.Model):

send\_choices = (

('register','注册'),

('forget','找回密码'),

('update\_email','修改邮箱'))

code = models.CharField('验证码',max\_length=20)

email = models.EmailField('邮箱',max\_length=50)

send\_type = models.CharField(choices=send\_choices,max\_length=30)

send\_time = models.DateTimeField(default=datetime.now)

class Meta:

verbose\_name = '邮箱验证码'

verbose\_name\_plural = verbose\_name

通过python manage.py makemigrations 和python manage.py migrate 进行数据库迁移

## 视图

Django的视图在views.py中构建，本模块的视图有以下个，通过views完成前后端数据的传值和操作。 django中模板多样的写法使前端的展现更加灵活，而这往往需要后端控制函数传递大量数据。Views文件中的函数匹配了对应的URL后会决定使用哪个模板，传递什么样的参数到模板。 数据交互是后端功能最为重要的一部分。Views文件引入模型模块，可以进行数据过滤分析提取等数据库操作功能。此外，views文件还能够获取前端的请求，得到表单数据进行比对分析插入数据库等。

# 邮箱和用户名都可以登录

# 基与ModelBackend类，因为它有authenticate方法

class CustomBackend(ModelBackend):

def authenticate(self, request, username=None, password=None, \*\*kwargs):

try:

# 不希望用户存在两个，get只能有一个。两个是get失败的一种原因 Q为使用并集查询

user = UserProfile.objects.get(Q(username=username) | Q(email=username))

# django的后台中密码加密：所以不能password==password

# UserProfile继承的AbstractUser中有def check\_password(self, raw\_password):

if user.check\_password(password):

return user

except Exception as e:

return None

# 登录

class LoginView(View):

'''用户登录'''

def get(self, request):

return render(request, 'login.html')

def post(self, request):

# 实例化

login\_form = LoginForm(request.POST)

if login\_form.is\_valid():

# 获取用户提交的用户名和密码

user\_name = request.POST.get('username', None)

pass\_word = request.POST.get('password', None)

# 成功返回user对象,失败None

user = authenticate(username=user\_name, password=pass\_word)

# 如果不是null说明验证成功

if user is not None:

if user.is\_active:

# 只有注册激活才能登录

login(request, user)

return HttpResponseRedirect(reverse('index'))

else:

return render(request, 'login.html', {'msg': '用户名或密码错误', 'login\_form': login\_form})

# 只有当用户名或密码不存在时，才返回错误信息到前端

else:

return render(request, 'login.html', {'msg': '用户名或密码错误', 'login\_form': login\_form})

# form.is\_valid（）已经判断不合法了，所以这里不需要再返回错误信息到前端了

else:

return render(request, 'login.html', {'login\_form': login\_form})

# 激活用户

class ActiveUserView(View):

def get(self, request, active\_code):

# 查询邮箱验证记录是否存在

all\_record = EmailVerifyRecord.objects.filter(code=active\_code)

if all\_record:

for record in all\_record:

# 获取到对应的邮箱

email = record.email

# 查找到邮箱对应的user

user = UserProfile.objects.get(email=email)

user.is\_active = True

user.save()

# 验证码不对的时候跳转到激活失败页面

else:

return render(request, 'user/active\_fail.html')

# 激活成功跳转到登录页面

return render(request, "login.html", )

class LogoutView(View):

'''用户登出'''

def get(self, request):

logout(request)

return HttpResponseRedirect(reverse('index'))

# 注册

class RegisterView(View):

'''用户注册'''

def get(self, request):

register\_form = RegisterForm()

return render(request, 'register.html', {'register\_form': register\_form})

def post(self, request):

register\_form = RegisterForm(request.POST)

if register\_form.is\_valid():

user\_name = request.POST.get('email', None)

# 如果用户已存在，则提示错误信息

if UserProfile.objects.filter(email=user\_name):

return render(request, 'register.html', {'register\_form': register\_form, 'msg': '用户已存在'})

pass\_word = request.POST.get('password', None)

# 实例化一个user\_profile对象

user\_profile = UserProfile()

user\_profile.username = user\_name

user\_profile.email = user\_name

user\_profile.is\_active = False

# 对保存到数据库的密码加密

user\_profile.password = make\_password(pass\_word)

user\_profile.save()

send\_register\_eamil(user\_name, 'register')

return render(request, 'login.html')

else:

return render(request, 'register.html', {'register\_form': register\_form})

class ForgetPwdView(View):

'''找回密码'''

def get(self, request):

forget\_form = ForgetPwdForm()

return render(request, 'user/forgetpwd.html', {'forget\_form': forget\_form})

def post(self, request):

forget\_form = ForgetPwdForm(request.POST)

if forget\_form.is\_valid():

email = request.POST.get('email', None)

send\_register\_eamil(email, 'forget')

return render(request, 'user/send\_success.html')

else:

return render(request, 'user/forgetpwd.html', {'forget\_form': forget\_form})

class ModifyPwdView(View):

'''修改用户密码'''

def post(self, request):

modify\_form = ModifyPwdForm(request.POST)

if modify\_form.is\_valid():

pwd1 = request.POST.get("password1", "")

pwd2 = request.POST.get("password2", "")

email = request.POST.get("email", "")

if pwd1 != pwd2:

return render(request, "user/password\_reset.html", {"email": email, "msg": "密码不一致！"})

user = UserProfile.objects.get(email=email)

user.password = make\_password(pwd2)

user.save()

return render(request, "login.html")

else:

email = request.POST.get("email", "")

return render(request, "user/password\_reset.html", {"email": email, "modify\_form": modify\_form})

class UserinfoView(LoginRequiredMixin, View):

""" 用户个人信息 """

def get(self, request):

return render(request, 'user/usercenter-info.html', {})

def post(self, request):

user\_info\_form = UserInfoForm(request.POST, instance=request.user)

if user\_info\_form.is\_valid():

user\_info\_form.save()

return HttpResponse('{"status":"success"}', content\_type= 'application/json')

else:

return HttpResponse (json.dumps(user\_info\_form.errors ), content\_type ='application/json')

class UploadImageView(LoginRequiredMixin, View):

'''用户图像修改'''

def post(self, request):

# 上传的文件都在request.FILES里面获取，所以这里要多传一个这个参数

image\_form = UploadImageForm(request.POST, request.FILES)

if image\_form.is\_valid():

image = image\_form.cleaned\_data['image']

request.user.image = image

request.user.save()

return HttpResponse('{"status":"success"}', content\_type='application/json')

else:

return HttpResponse('{"status":"fail"}', content\_type='application/json')

class SendEmailCodeView(LoginRequiredMixin, View):

'''发送邮箱修改验证码'''

def get(self, request):

email = request.GET.get('email', '')

if UserProfile.objects.filter(email=email):

return HttpResponse('{"email":"邮箱已存在"}', content\_type='application/json')

send\_register\_eamil(email, 'update\_email')

return HttpResponse('{"status":"success"}', content\_type='application/json')

class MycommendView(LoginRequiredMixin, View):

'''我的推荐'''

def get(self, request):

return render(request, "user/usercenter-commend.html", )

class MyproblemsView(LoginRequiredMixin, View):

'''常见问题'''

def get(self, request):

return render(request, "user/usercenter-problems.html", )

def ad\_list(request):

# 取出所有网站帖子

ads = AdPost.objects.all()

# 需要传递给模板（templates）的对象

context = {'ads': ads}

# render函数：载入模板，并返回context对象

return render(request, 'user/usercenter-post-ad.html', context)

def lostfound\_list(request):

# 取出所有网站帖子

lostfounds = LostfoundPost.objects.all()

# 需要传递给模板（templates）的对象

context = {'lostfounds': lostfounds}

# render函数：载入模板，并返回context对象

return render(request, 'user/usercenter-post-lostfound.html', context)

def seekhelp\_list(request):

# 取出所有网站帖子

seekhelps = SeekhelpPost.objects.all()

# 需要传递给模板（templates）的对象

context = {'seekhelps': seekhelps}

# render函数：载入模板，并返回context对象

return render(request, 'user/usercenter-post-seekhelp.html', context)

def salebuy\_list(request):

# 取出所有网站帖子

salebuys = SalebuyPost.objects.all()

# 需要传递给模板（templates）的对象

context = {'salebuys': salebuys}

# render函数：载入模板，并返回context对象

return render(request, 'user/usercenter-post-salebuy.html', context)

class IndexView(View):

"""首页"""

def get(self, request):

return render(request, 'index.html', )

def page\_not\_found(request, exception):

# 全局404处理函数

from django.shortcuts import render\_to\_response

response = render\_to\_response('404.html', {})

response.status\_code = 404

return response

def page\_error(request):

# 全局500处理函数

from django.shortcuts import render\_to\_response

response = render\_to\_response('500.html', {})

response.status\_code = 500

return response