第六课

关于密码这一块所有内容

逻辑: 在密码忘记的时候需要发一个网页+验证码给用户进行,进行修改密码.

修改密码之后可以把修改的密码之后存到数据库里面.

还有一个就是直接修改密码:

在user_infomation页面有修改密码的页面. 需要写一个验证用户是不是已经登录的view,如果没有登录就跳转到login页面

直接修改密码

在url设置两个url配置.

```
from users.views import FrogetPwdView, ResetPwd

# 忘记密码
url(r'^forget_pwd/$', FrogetPwdView.as_view(), name="forget_pwd"),
```

在from.py里面设置需要验证的字段和验证码

```
# 忘记密码的验证字段
class ForgetForm(forms.Form):
    email = forms.EmailField(required=True)
    captcha = CaptchaField(error_messages={"invalid": u"验证码错误"})
```

在view.py设置忘记密码的逻辑

```
# 忘记密码
class FrogetPwdView(View):
    def get(self, request):
        forget_form=ForgetForm()  #定义一个form 里面需要有一个返回验证码的字段
        return render(request, 'forgetpwd.html',{"forget_form":forget_form})

    def post(self, request):
        forget_from = ForgetPwdForm(request.POST)
        if forget_from.is_valid():
            email = request.POST.get("email", "")
            if UserInfo.objects.filter(email=email):  #查询有没有这个emial
                send_register_email(email, 'forget')
                return render(request, 'success_activate.html', {'email': email})
            return render(request, 'forgetpwd.html', {'msg':u"用户不存在!!"})
        else:
            return render(request, 'forgetpwd.html', {'forget_from': forget_from})
```

在发送邮件的里类型里面加上一个forget的类型

```

```

在forgetpwd.html 里修改前段的字段, 在回填和要显示的字段显示出来

```
 action="{% url 'forget_pwd' %}"
 {% if forget_form.errors.email %}errorput{% endif %}
 value="{% if forget_from.name.errors %}{% else %}{{ forget_from.email.value }}{% endif  %}"
 {% for key,error in forget_form.errors.items %}{{ error }}{% endfor %}{{ msg }}
```

在发送验证码之后, 需要对验证码进行验证,然后返回到一个可以修改密码的页面,然后修改密码后保存到数据库.

```
# 找回密码
url(r'^reset/(?P<active_code>.*)/$', ResetPwd.as_view(), name='reset_pwd'),

# 确认修改密码
url(r'^modify_pwd/$', ModifyPwdView.as_view(), name="modify_pwd"),
```

因为我们在提交的时候不需要把验证提交进去,提交进去之后有错误 ,我们就需要对提交表单重新写一个逻辑

```

# 修改密码
class ResetView(View):

    def get(self, request, active_code):
        all_records = EmailVerify.objects.filter(code=active_code)
        if all_records:
            for record in all_records:
                email = record.email
                return render(request, "password_reset.html", {"email": email})
        else:
            return render(request, "send_success.html")
        return render(request, "login.html")


# 确认修改密码
class ModifyPwdView(View):
    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, "password_reset.html", {"email": email, "msg": "密码不一
致"})
            user = UserInfo.objects.get(email=email)
            user.password = make_password(pwd2)
            user.save()
            return render(request, 'reset_success.html')

            # return HttpResponseRedirect(reverse("login"))
```

```
        else:
            email = request.POST.get("email", "")
            return render(request, "password_reset.html", {"email": email, "modify_form":
modify_form})
```

在url里面把user.url都设置进来

```
    # 关于用户
    url(r'^i/',include('users.urls',namespace='i')),
```

在user.urls.py设置需要的url 先设置两个url 一个用户信息 和一个修改密码的url, 帮两url配置好之后 把view的逻辑写好

```
    from django.conf.urls import url
from users.views import UserInfoView, ChangePwdView

urlpatterns = [
    # 用户信息
    url(r'info/$', UserInfoView.as_view(), name='info'),

    # 修改密码
    url(r'change_pwd/$', ChangePwdView.as_view(), name='change_pwd'),
]
```

获取info页面和password页面

```
# 关于用户的信息
class UserInfoView(LoginRequiredMixin, View):
    def get(self, request):
        return render(request, 'my_info.html', {})


# 关于用户密码修改
class ChangePwdView(LoginRequiredMixin, View):
    def get(self, request):
        email = request.user.email
        return render(request, 'my_paasword.html', {'email': email})

    def post(self, request):
        change_form = ChangePwdForm(request.POST)
        email = request.user.email

        # 验证前段传的数据
        if change_form.is_valid():
            pwdold = request.POST.get("passwordold", "")
            email = request.POST.get("email", "")
            user = authenticate(username=email, password=pwdold)
            if user is not None:
                pwd1 = request.POST.get("password1", "")

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

```
                if pwd1 != pwd2:
                    return render(request, "my_paasword.html", {"email": email, "msg": u"密码不
一致"})
                user = UserInfo.objects.get(email=email)
                user.password = make_password(pwd2)
                user.save()

                return render(request, 'reset_success.html')
            else:
                return render(request, "my_paasword.html", {"email": email, "msg": u"旧密码不
对"})

        return render(request, 'my_paasword.html', {'change_form': change_form, 'email':
email})
```