第五节课

1.路径修改 2.验证码, 3.发送邮件,4.注册逻辑

修改apps的位置,新建一个apps的python package 把所有的app都放在一个文件下面

```
#pycharm 修改apps的属性    修改这个可以在pycharm里运行
#apps-->Mark Directory as--> Sources Root


#  修改setting.py 项目可以在linux里运行
import sys
sys.path.insert(0, os.path.join(BASE_DIR, 'apps'))
```

修改 html 文件里面的url配置

配置urls.py

{% url 'name' %}

也把static的文件进行修改

{{ load staticfiles }}

{% static 'css' %}路径地址的转换

利用正则表达式:进行快速操作

ctrl + r

选择要替换的一个内容,然后再进行正则匹配

替换的需要的替换的内容用()括起来 注意是不是需要贪婪匹配

然后在利用正则的$1来替换

```
ctrl + r
../static/(.*?)"
{% static '$1' %}"
```

~~验证码~~

django有一个安装的验证码的

在github上找到这个包

然后看使用的教程,按照教程进行操作.

找到django-simple-captcha 安装这包

github 网站上的地址： https://github.com/mbi/django-simple-captcha

```
1.Install django-simple-captcha via pip: pip install  django-simple-captcha

2.Add captcha to the INSTALLED_APPS in your settings.py

3.Run python manage.py migrate

4.Add an entry to your urls.py:

urlpatterns += [
    url(r'^captcha/', include('captcha.urls')),
]
```

在forms.py里面添加好字段

```
from captcha.fields import CaptchaField

class RegisterForm(forms.Form):
    email = forms.EmailField(required=True)
    password = forms.CharField(required=True, min_length=6)
    captcha = CaptchaField(error_messages={"invalid": u"验证码错误"})
```

在views.py里面写入自己的逻辑

```
from django.contrib.auth.hashers import make_password

class RegisterView(View):
    def get(self, request):
        register_form = RegisterForm()
        return render(request, "register.html", {'register_form': register_form})
```

配置url.py

```
url(r'^register/$', RegisterView.as_view(), name='register'),
```

html里面修改自己的验证码的位置,去掉不要的内容

```
{{ register_form.captcha }}  #添加captcha

{% csrf_token %}   #添加csrf验证表单
```

在提交表单的时候 action'url地址' method='post'

~~验证码~~

然后在view.py里面书写逻辑

```
 # views.py

class RegisterView(View):

    def post(self, request):
        register_form = RegisterForm(request.POST)
        if register_form.is_valid():
            email = request.POST.get("email", "")
            if UserProfile.objects.filter(email=user_name):
                return render(request, "register.html", {
                    "register_form": register_form,
                    "msg": "用户已经存在!",
                })
            pass_word = request.POST.get("password", "")
            user_profile = UserProfile()
            user_profile.username =email
            user_profile.email = email
            user_profile.is_active = False
            user_profile.password = make_password(pass_word)
            user_profile.save()
```

发送邮箱的py文件配置

```
# setting.py  # 在配饰setting的
#  使用新浪来发送邮件
EMAIL_HOST = "smtp.sina.com"
EMAIL_PORT = 25
EMAIL_HOST_USER = "sina@sina.com"
EMAIL_HOST_PASSWORD = "密码"
EMAIL_USE_TIS = False
EMAIL_FROM = "sina@sina.com"


#  使用QQ来发送邮件
EMAIL_HOST = "smtp.qq.com"
EMAIL_PORT = 465  # SSL  # 第三种配置方式
# EMAIL_PORT = 587  #  TSL  # 第二种配置方式
# EMAIL_PORT = 25   #第一种配置方式
EMAIL_HOST_USER = "qq@qq.com"
EMAIL_HOST_PASSWORD = "秘钥"    '企业qq':"pqwpilzfhyrgdcia"
# EMAIL_USE_TLS = True  #第一种配置方式 # 第二种配置方式
EMAIL_USE_SSL = True   #第三种配置方式
EMAIL_FROM = "qq@qq.com"
```

在app.py里面新建一个 python.package (utils)

里面新建一个py文件 send_email.py 在里面设置好发送的字段 头,文件,内容,url

```
#  -*- coding:utf-8 -*-
from random import Random

from django.core.mail import send_mail

from users.models import EmailVerify
from jichuDjango.settings import EMAIL_FROM

  # 定义一个随机字符串的方法
def random_str(randomlenght=8):
    str = ''
    chars = 'AaBbCcDdEeFfGgHhIiJjKkLlNnMmOoPpQqRrSsTtUuVvWwXxYyZz012346789'
    length = len(chars) - 1
    random = Random()
    for i in range(randomlenght):
        str += chars[random.randint(0, length)]
    return str

  # 定义了一个发送邮件的
def send_register_email(email, send_type="register"):
    email_record = EmailVerify()
    code = random_str(16)
    email_record.code = code
    email_record.email = email
    email_record.send_type = send_type
    email_record.save()

    email_title = ""
    email_body = ""

    if send_type == "register":
        email_title = u"在线注册激活链接"
        email_body = u"请点击下面的链接激活你的账号：
http://192.168.83.128:8000/active/{0}".format(code)

        send_status = send_mail(email_title, email_body, EMAIL_FROM, [email])
        if send_status:   # 需要对这个状态进行dubug,
            pass
```

新建url的链接方式.

在urls.py文件下添加

```
url(r'^active/(?P<active_code>.*)/$', ActiveUserView.as_view(), name='active_code'),
```

在user.views.py写逻辑

```
# 在post的逻辑后面添加  userprofile.save()之后发送
#  发送邮件,返回页面 发送错误就返回一个

            send_register_email(user_name,'register')
            return render(request, 'login.html')
        else:
            return render(request, "register.html",{"register_form":register_form})
```

在urls.py 里面对激活的代码进行验证.

```
    url(r'^active/(?P<active_code>.*)/$', ActiveUserView.as_view(), name='active_code'),
```

在views.py 对登录逻辑里面对, 用户的是否激活进行一个验证

```
# 在用户验证有没有只有要对is_active进行判断，然后在返回之后的页面
#返回的页面也应该用返回重定向的方式返回

            if user is not None:
                if user.is_active:
                    login(request, user)
                    return HttpResponseRedirect(reverse("index"))
                else:
                    return render(request, "login.html", {"msg": "用户未激活！"})
            else:
                return render(request, 'login.html', {"msg": "用户或密码错误!"})
        return render(request, "login.html", {"login_form": login_form})
```

激活 views.py

```
# 激活
class ActiveUserView(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
                user = UserInfo.objects.get(email=email)
                user.is_active = True
                user.save()
                return render(request, 'success_activate.html')
            return render(request, 'login.html')
```