邮箱验证

1、首先需要一个 QQ 邮箱授权码： 进入QQ 邮箱点击设置

1. 示例图

2、开启服务，并且生成授权码

1. 示例图

3、setting.py 配置

1. 示例代码

```
EMAIL_HOST = 'smtp.qq.com'
EMAIL_PORT = 25 #如果不好使 就换成 465
EMAIL_HOST_USER = 'xxx@qq.com' # 你的QQ账号
EMAIL_HOST_PASSWORD = '授权码,不是qq邮箱密码'
EMAIL_USE_TLS = True # 这里必须是 True，否则发送不成功
EMAIL_FROM = 'xxx@qq.com' # 你的 QQ 账号
```

4、业务逻辑

1. 说明
    1. 处理用户注册数据，存入数据库，is_active字段设置为False，用户未认证之前不允许登陆
    2. 产生token，生成验证连接URL
    3. 发送验证邮件
    4. 用户通过认证邮箱点击验证连接，设置is_active字段为True，可以登陆
    5. 若验证连接过期，删除用户在数据库中的注册信息，允许用户重新注册（username、email字段具有唯一性）
2. 邮件验证连接主要有两步
    - 一是产生token，发送邮件
    - 二是处理验证链接。这里采用base64加密，及itsdangerous序列化（自带时间戳）

```
from itsdangerous import URLSafeTimedSerializer as utsr
import base64
from django.conf import settings as django_settings

class Token:
    def __init__(self, security_key):
        self.security_key = security_key
        self.salt =
base64.encodebytes(security_key.encode('utf8'))

    def generate_validate_token(self, username):
        serializer = utsr(self.security_key)
        return serializer.dumps(username, self.salt)

    def confirm_validate_token(self, token, expiration=3600):
        serializer = utsr(self.security_key)
```

```
        return serializer.loads(token, salt=self.salt,
max_age=expiration)

    def remove_validate_token(self, token):
        serializer = utsr(self.security_key)
        print(serializer.loads(token, salt=self.salt))
        return serializer.loads(token, salt=self.salt)


token_confirm = Token(django_settings.SECRET_KEY)    # 定义为全局
变量
```

3. 注册发送邮箱

```
def register_view(request):
        if request.method == 'POST':
        try:
            username = request.POST.get('username')
            password = request.POST.get('password')

            # 验证用户是否存在
            user = authenticate(username=username,
password=password)
            if user:
                # 用户已经存在
                return render(request, 'register.html', {'msg':
'用户名已存在'})
            else:
                # 保存用户
                user =
User.objects.create_user(username=username,
                                                password=password
                                               )
                user.is_active = False
                # 发送邮件验证
                token =
token_confirm.generate_validate_token(user.username)
                link = reverse("App:active",kwargs=
{'token':token})

                link = "http://"+request.get_host() + link
                print(link)
```

```
                html =
loader.get_template('active.html').render({'link':link})
                send_mail('账户激活','',EMAIL_FROM,
['landmark_csl@126.com'],html_message=html)
            return render(request, 'message.html', {'message':
"请登录到注册邮箱中验证用户，有效期为1个小时",

 'username':username})
        except Exception as e:
            print(e)
            return render(request, 'register.html', {'msg': '注册
失败,用户名或密码错误'})
    else:
        return render(request, 'register.html')
```

4. 激活用户

```
def active_user(request, token):
        try:
        username = token_confirm.confirm_validate_token(token)
    except:
        username = token_confirm.remove_validate_token(token)
        users = User.objects.filter(username=username)
        for user in users:
            user.delete()
        return render(request, 'message.html', {
            'message': "对不起，验证链接已经过期，请重新<a
href='/register/'>注册</a>"})
    try:
        user = User.objects.get(username=username)
    except User.DoesNotExist:
        return render(request, 'message.html', {'message': u"对不
起，您所验证的用户不存在，请重新注册"})
    user.is_active = True
    user.save()
    message = "验证成功，请进行<a href='/login/'>登录</a>操作"
    return render(request, 'message.html', {'message': message})
```