#2 基于Flask和Bootstrap实现的匿名聊天站

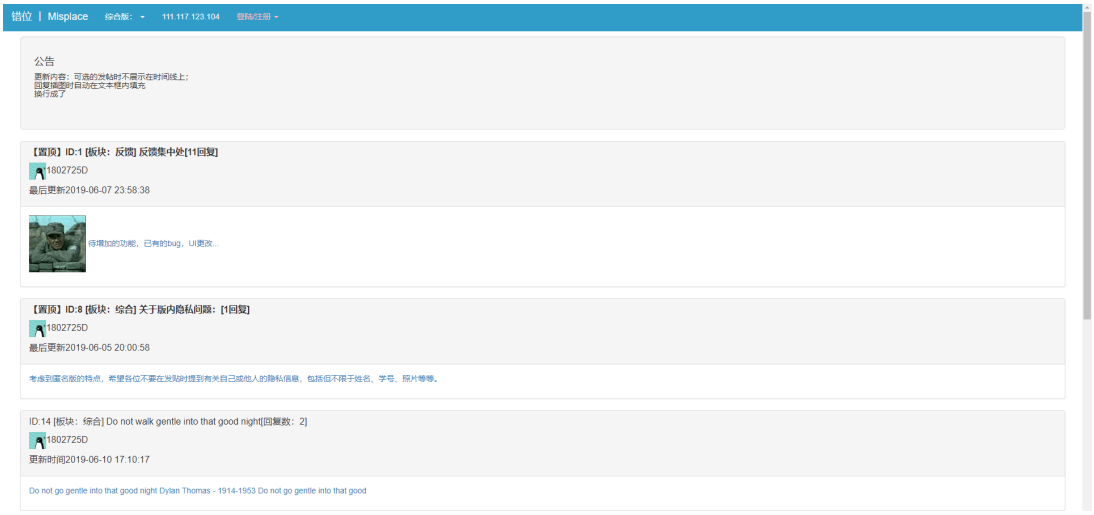
0. 项目描述

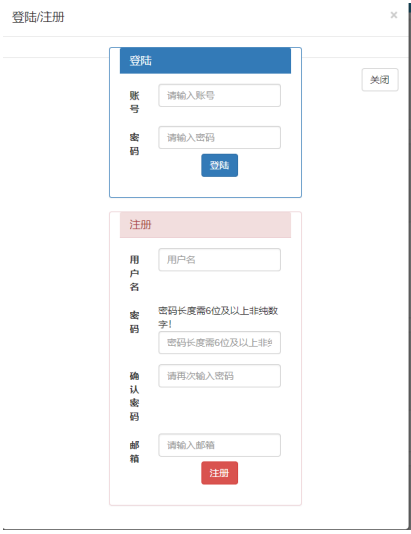
- **纯属兴趣，在课余时间开发，技术水平有限**，功能和一些bug有待完善；
- 需求描述：匿名聊天，保证类似正常论坛聊天功能的同时不在帖子中显示用户真实信息（包括昵称）；
- 开发人员：袁廷洲，赵书彬；
- 项目技术：

> - 语言：Python+SQL，JavaScript+HTML+CSS；
> - 框架：Flask，bootstrap；
> - 库：flask衍生库(flask_sqlalchemy)，pymysql，PIL/pillow，JQuery等；
> - 数据库:MySQL；
> - api：QQsmtp；

- demo网站：154.8.233.189
- 源代码仓库：https://github.com/tzY15368/nmb

1. 基本界面(有bootstrap自带的移动端适配)





2. 部分实现功能的 **要点细节和业务逻辑**（基于Python的flask框架写成）：

- 登陆/注册（前端使用弹出的模态框，后端使用了session和md5加密实现）+邮箱验证（使用了QQ邮箱的smtp api）：

```
#后端Python代码
#session+md5登陆
@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('home',logout=1))


@app.route('/login',methods=['GET','POST'])
def login():
    if request.method == 'GET':
        return return500()
    form = loginform()
    if form.validate_on_submit():
        email = form.name.data#其实是email
        password = form.password.data
        print(email,password)
        result = User.query.filter(User.email==email).first()#唯一辨识符
        if result == None:
            return redirect(url_for('home',err=1))
        elif result.password == password:
            session['account']=result.email
            session['kookies']=result.kookies
            session['avatar']=result.avatar
            active_ip=request.remote_addr
            User.query.filter(User.email==email).update({'active_ip':active_ip})
            try:
                db.session.commit()
            except Exception as e:
                print(e)
            return redirect(url_for('homepage'))
        else:
            return redirect(url_for('home',err=1))
    else:
        return redirect(url_for('home',novalidation=1))

@app.route('/signin',methods=['GET','POST'])
def signin():
    if request.method == 'GET':
        return return500()
    form = signinform()
    if form.validate_on_submit():
        username = form.name.data
        psw1 = form.password1.data
        psw2 = form.password2.data
        email = form.email.data
        #print(username, psw1, email)
```

```
        if psw1 != psw2:
            print('密码与确认密码不同')
            return redirect(url_for('home',err=1))
        try:
            newuser =
User(username=username,password=psw1,email=email,active_ip=request.remote_addr,pas
sword_hash=md5(psw1),confirmed=False,admin=False)
            db.session.add(newuser)
            db.session.commit()
            session['account']=email
            session['kookies']='00000000' #注意这是饼干而不是cookie
            psw_hash = User.query.filter(User.email==email).first().password_hash
            if not psw_hash:
                return return500('psw_hash is None')
            try:
                sender(email,'http://www.ftmagic.xyz:6060/api/confirmation?
code='+psw_hash)
            except Exception as e:
                print(e)
                return return500('邮件发送失败')
            return redirect(url_for('homepage'))
        except Exception as e:
            db.session.rollback()
            print(e)
            return redirect(url_for('home',signin_error=1))
    else:
        return redirect(url_for('home',novalidation=1))
```

- 通过邮箱验证api实现限制未验证用户发帖（饼干系统），并通过饼干系统将id匿名化从而实现匿名功能（前端只显示用户饼干名称，而不显示id，也不传递任何含用户信息的数据）：

```
# 制造饼干（md5加密）
import hashlib
import time

def md5(arg):
    md5_pwd = hashlib.md5(bytes('abd',encoding='utf-8'))
    md5_pwd.update(bytes(arg,encoding='utf-8'))
    return md5_pwd.hexdigest()

def md5single(arg):
    return hashlib.md5(arg.encode("utf-8")).hexdigest()#与前端js中方法一致
if __name__ == '__main__':
    print(md5(str(time.time())))
    print(md5('123'))
    print(md5single('123'))
```

```
# 重新制造饼干（匿名id）
@app.route('/newkookie',methods=['GET','POST'])
def newkookie():
    if checklogin():
        return redirect(url_for('home',nologin=1))
    if check_confirmation():
        return redirect(url_for('homepage',no_confirmation=1))
    account = session.get('account')
    kookie = cookie(User.query.filter(User.email==account).first().username)
    result = User.query.filter(User.email == account).update({'kookies': kookie})
    try:
        db.session.commit()
        try:
            oldkookie =
str(User.query.filter(User.email==account).first().oldkookies)
            oldkookie += (kookie+'-')

User.query.filter(User.email==account).update({'oldkookies':oldkookie})#保存历史
kookie
            db.session.commit()
            session['kookie'] = kookie#s设置新kookie
        except Exception as e:
            db.session.rollback()
            print(e)
            return return500()
    except Exception as e:
        db.session.rollback()
        print(e)
        return return500()
    return redirect(request.referrer)
```

- 记录用户IP且形成IP-id对应的限制；
- 不同板块发表帖子，回复帖子；

---

3. 有待完善的功能

- 待增加验证登陆时专门的密码验证加密方式，前端向后端发起ajax请求，后端返回salt，前端加密提交数据与后端比对
- cookie系统的完善

---

4. 主要收获

- 实践了后端的MVC模式（比较简化，M和C合并在一起了）；
- flask_sqlalchemy提供的ORM，可以把复杂的SQL语句给包装成更加面向对象，易于理解的样子；
- 仅用cookie作登陆状态验证是不安全的，而session+md5可以一定程度缓解问题（但是没有彻底解决，存在伪造sessionID的可能）；
- btw，小幅改动即可变成实名版（删除饼干系统，替换为id）；