邮箱设置-->账户-->开启IMAP/SMTP服务-->手机验证-->授权码

**注意**：千万不要把账户密码直接写入脚本；为了保护账户信息，让脚本从环境变量中导入敏感信息。

设置环境变量：

Linux：export MAIL\_USERNAME=< username >

Windows：set MAIL\_USERNAME=< username >

* 在程序中集成发送电子邮件功能

**from** flask\_mail **import** Message  
app.config[**'FLASKY\_MAIL\_SUBJECT\_PREFIX'**] = **'[Flasky]'**app.config[**'FLASKY\_MAIL\_SENDER'**] = **'Flasky Admin <flasky@example.com>'  
def** send\_email(to, subject, template, \*\*kwargs):  
 *# to为接收方,subject为邮件主题,template为渲染邮件正文的模板* msg = Message(app.config[**'FLASKY\_MAIL\_SUBJECT\_PREFIX'**] + **' '** + subject,  
 sender=app.config[**'FLASKY\_MAIL\_SENDER'**], recipients=[to])  
 msg.body = render\_template(template + **'.txt'**, \*\*kwargs)  
 msg.html = render\_template(template + **'.html'**, \*\*kwargs)  
 mail.send(msg)

index()视图修改为：

app.config[**'FLASKY\_ADMIN'**] = os.environ.get(**'FLASKY\_ADMIN'**)  
@app.route(**'/'**, methods=[**'GET'**, **'POST'**]) *# 支持GET和POST***def** index():  
 form = NameForm()  
 **if** form.validate\_on\_submit(): *# 提交表单,数据被验证函数接受* user = User.query.filter\_by(username=form.name.data).first() *# 数据库中查询* **if** user **is None**: *# 用户不存在于数据库,加入* user = User(username=form.name.data)  
 db.session.add(user)session[**'known'**] = **False** *# known记录是不是新用户* **if** app.config[**'FLASKY\_ADMIN'**]:  
 send\_email(app.config[**'FLASKY\_ADMIN'**], **'New User'**, **'mail/new\_user'**, user=user)  
 **else**:  
 session[**'known'**] = **True** session[**'name'**] = form.name.data *# 获取字段data属性存入session* form.name.data = **''** *# 清空字段* **return** redirect(url\_for(**'index'**))  
 **return** render\_template(**'index.html'**, form=form, name=session.get(**'name'**), known=session.get(**'known'**, **False**))

电子邮件的收件人保存在环境变量FLASKY\_ADMIN中，程序启动加载到同名配置变量中。要创建两个模板文件(templates文件夹下的mail文件夹下)，分别用于渲染纯文本和HTML版本的邮件正文。

电子邮件的模板中要有一个模板参数是user，因此调用send\_mail()函数时要以

关键字参数的形式传入user。

注意：上面代码未亲自测试，不一定靠谱！

* 异步发送电子邮件

**from** threading **import** Thread  
**def** send\_async\_email(app, msg):  
 **with** app.app\_context():  
 mail.send(msg)  
**def** send\_email(to, subject, template, \*\*kwargs):  
 msg = Message(app.config[**'FLASKY\_MAIL\_SUBJECT\_PREFIX'**] + **' '** + subject,  
 sender=app.config[**'FLASKY\_MAIL\_SENDER'**], recipients=[to])  
 msg.body = render\_template(template + **'.txt'**, \*\*kwargs)  
 msg.html = render\_template(template + **'.html'**, \*\*kwargs)  
 t = Thread(target=send\_async\_email, args=[app, msg])  
 t.start()  
 **return** t

很多Flask扩展都假设已经存在激活的程序上下文和请求上下文。 Flask-Mail中的send()函数使用current\_app，因此必须激活程序上下文。但在不同线程中执行 send()函数时，程序上下文要使用app.app\_context()人工创建。