课程设计报告

科目：python程序设计与数据分析

时间：2020年6月29日

题目：留言板网站

目录

[一、 题目 3](#_Toc44353107)

[二、 需求分析 3](#_Toc44353108)

[三、 功能实现 3](#_Toc44353109)

[（1） 登录 3](#_Toc44353110)

[（2） 注册 4](#_Toc44353111)

[（3） 首页 5](#_Toc44353112)

[（4） 留言 5](#_Toc44353113)

[（5） 个人中心（管理留言） 5](#_Toc44353114)

[（6） 查找 6](#_Toc44353115)

[（7） 注销 7](#_Toc44353116)

[四、 工程结构和核心代码 7](#_Toc44353117)

[（1） 工程结构 7](#_Toc44353118)

[（2） 核心代码 8](#_Toc44353119)

[五、 改进和不足 10](#_Toc44353120)

[六、 参考 11](#_Toc44353121)

1. 题目

开发设计一个留言板系统；用户登陆后，能发布留言，能对自己发布的留言进行管理；能查看留言列表，分页显示，能根据留言的发布者，和留言内容，做模糊查询。

基于flask框架

1. 需求分析

需要的功能：登录、注册、查看留言、发布留言、管理自己的留言、查找留言。

登录：通过邮箱和密码进行登录，密码经过加密后与数据库中存储的密码进行比较，一致即登录成功。

注册：使用邮箱、用户名、密码、确认密码进行注册。确保注册邮箱不会重复，密码和确认密码一致，即注册成功。

查看留言：进入首页即可查看所有留言，首页具有分页显示功能。

发布留言：发布内容不得超过150字，只有登录后才可以发布留言。

管理自己的留言：在登录后，可以进入个人中心管理留言。参照腾讯QQ的QQ空间的留言板，留言不可修改，所有留言板的管理功能只有删除自己的留言。

查找：可以进行模糊查询，可以通过留言的发布者名字查询，也可以根据留言的内容查询。

1. 功能实现
2. 登录

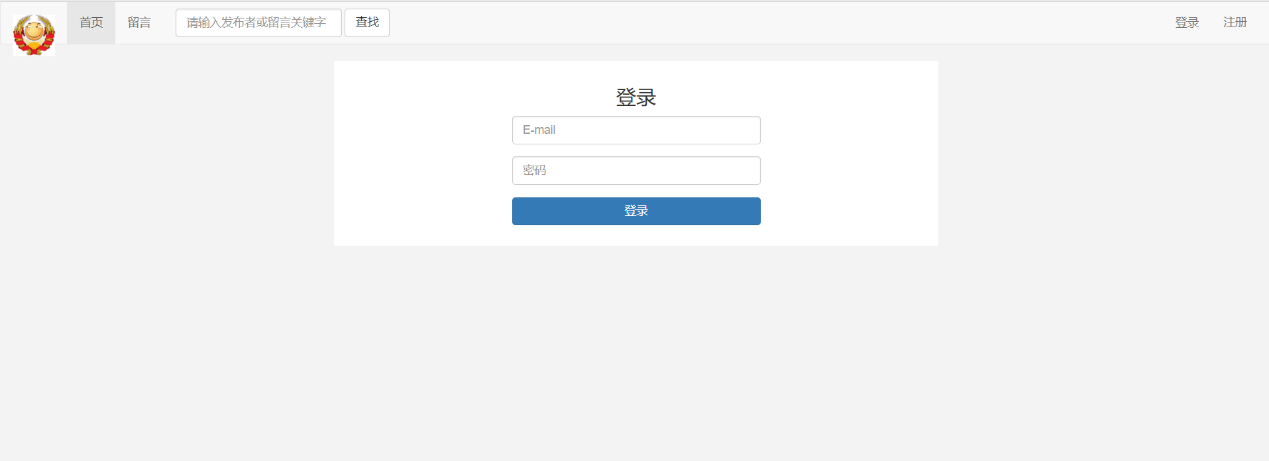
在界面输入邮箱和密码后，点击登录按钮。由login视图函数处理post请求。在数据库中对比邮箱和密码，若数据库中已有且密码一致，则登录成功，并跳转到主页，导航条也发生变化。

图1.登录界面

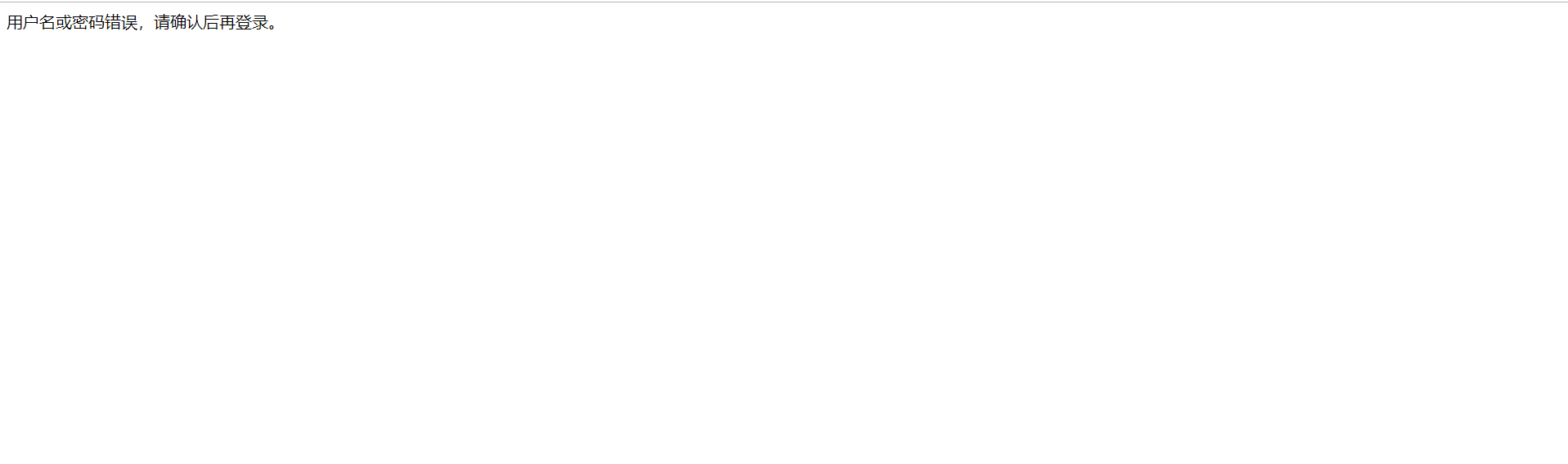


图2.登录失败



图3.登录成功，跳转到首页

登录成功后，页面跳转到首页。导航条右边的“登录”、“注册”标签也会变成“‘用户名’+的个人中心”和“注销”。

点击注销后，用户会注销，并返回登录界面。导航条的右边又会变回“登录”和“注销”。

在登录之前，导航栏的“留言”是不可点击的，而且不会跳转。而“首页”和“查找”功能不受影响。

1. 注册

用户使用邮箱、用户名、密码、确认密码进行注册。确保注册邮箱不会重复，密码和确认密码一致，即注册成功。注册成功后跳转到登录界面，完成登录后，即可正常使用。由视图函数regist实现功能。

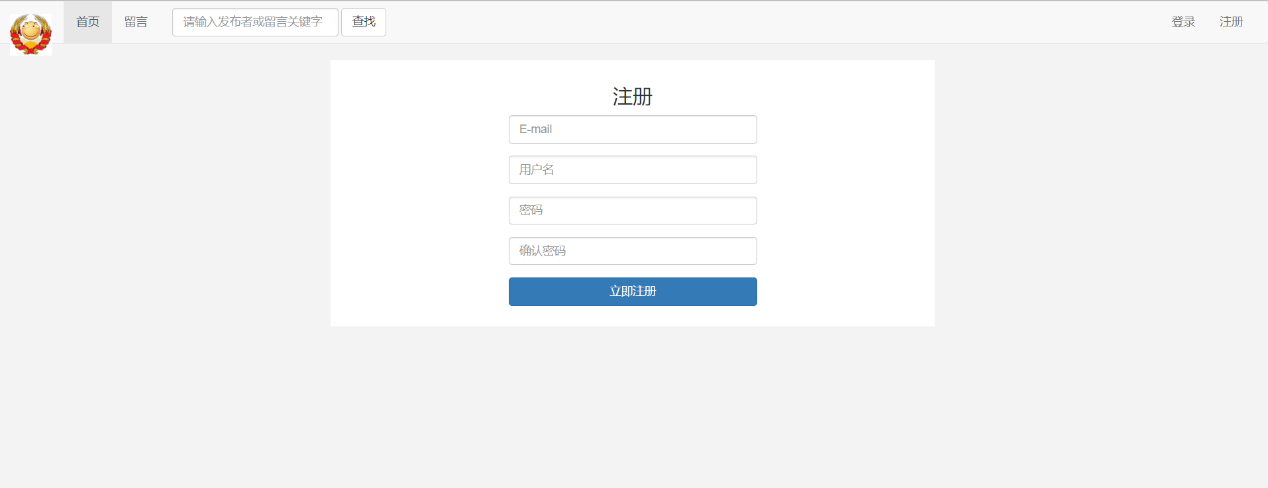


图4.注册界面

1. 首页

一进入网站即网站首页。首页显示出网站所有的留言，并且分页展示，每页显示五条留言。留言展示顺序按照发布时间排序，发布最新的留言展示在最前面。登录和未登录的首页功能并无变化，而页面只有导航条不同。由视图函数index实现功能。

1. 留言

在登录后，用户才能使用留言功能。在文本域中输入留言内容，内容不能位空或者超过150字。留言发布成功后，跳转进入首页。由视图函数leave实现功能。

图5.留言发布成功



图6.留言不能超过150字



图7.留言不能为空

1. 个人中心（管理留言）

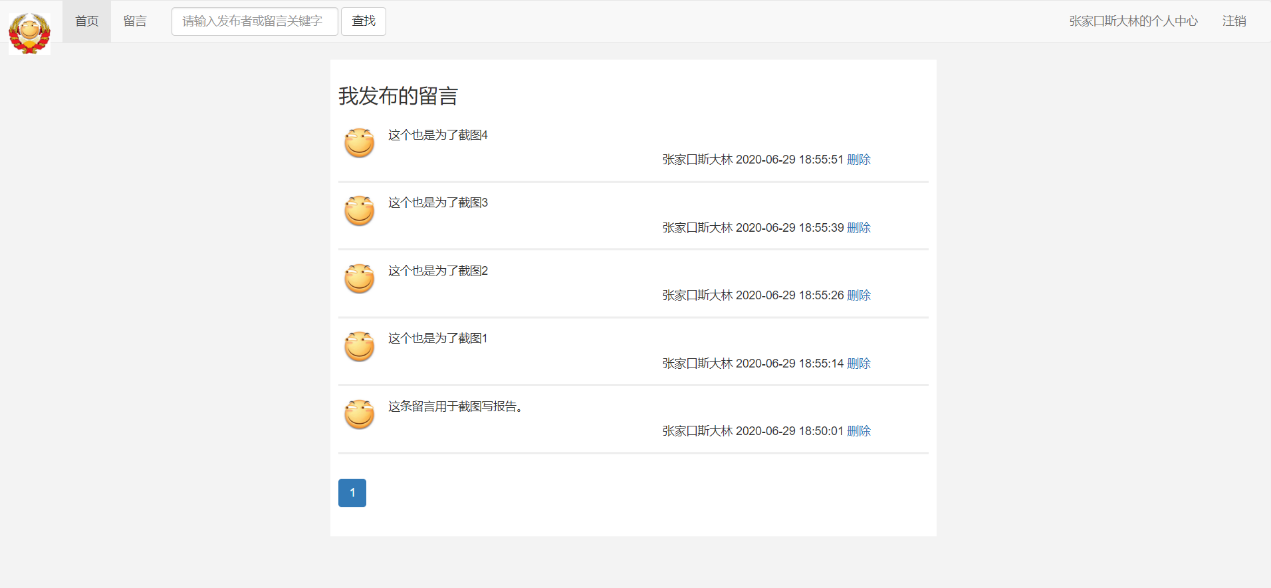
只有登录后，导航条上才会展示出“个人中心”的标签。点击进入后，会展示出用户发布的留言，依然是最新发布的排在最前面，分页展示，每页展示5条。其中与首页不同的是，每条留言都有一个删除标签，点击删除，即可删除留言，然后跳转回本页。该功能由视图函数MyCenter和delete实现。

图8.个人中心

1. 查找

查找功能实现了模糊查找，按照发布者名字查找和按照留言内容查找。该功能受是否登录影响，由视图函数search实现。



图9.根据留言内容查找



图10.根据发布者名字查找

1. 注销

登录后才能点击注销，注销后，会跳转到登录界面。由视图函数logout实现。

1. 工程结构和核心代码
2. 工程结构

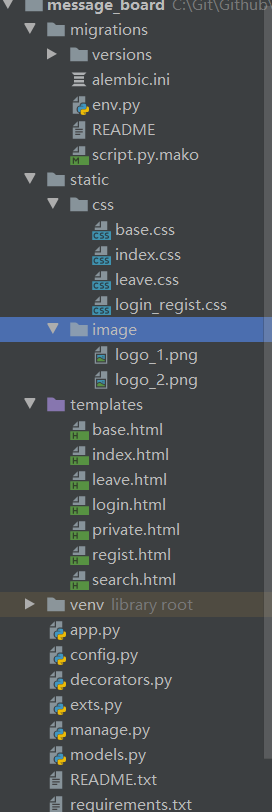


图11.工程结构

Migrations存放数据库迁移文件。Static存放css代码和网页用到的图片。Templates存放jinja2模板。Venv存放虚拟环境。App.py是所有视图函数，config.ppy是配置文件，decorators.py是所有装饰器函数，manage.py数据库迁移脚本，models.py数据库的模型，包括用户模型和留言模型。

1. 核心代码
2. # 主页
3. @app.route('/')
4. **def** index():
5. #获取页码数
6. page = request.args.get('page',1)
7. # 分页器对象
8. query = Message.query.order\_by(Message.create\_time.desc())
9. paginate = query.paginate(page=int(page), per\_page=per\_page)
10. **return** render\_template('index.html', paginate=paginate)
12. # 登录
13. @app.route('/login/', methods=['GET','POST'])
14. **def** login():
15. **if** request.method == 'GET':
16. **return** render\_template('login.html')
17. **else**:
18. Email = request.form.get('Email')
19. password = request.form.get('password')
20. user = User.query.filter(User.Email == Email).first()
21. **if** user **and** user.check\_password(password):
22. # 记录进入cookie
23. session['user\_id'] = user.id
24. # 31天内不用再登录
25. session.permanent = True
26. **return** redirect(url\_for('index'))
27. **else**:
28. **return** u"用户名或密码错误，请确认后再登录。"
30. # 注册
31. @app.route('/regist/', methods=['GET','POST'])
32. **def** regist():
33. **if** request.method == 'GET':
34. **return** render\_template('regist.html')
35. **else**:
36. Email = request.form.get('Email')
37. username = request.form.get('username')
38. password1 = request.form.get('password1')
39. password2 = request.form.get('password2')
41. #如果Email被注册了，就不能再注册。
42. user = User.query.filter(User.Email == Email).first()
43. **if** user:
44. **return** u"该Email已经被注册，请更换Email。"
45. **else**:
46. **if** password1!=password2:
47. **return**  u"两次密码输入不一致。"
48. **else**:
49. user = User(Email=Email, username=username, password=password1)
50. db.session.add(user)
51. db.session.commit()
52. # 注册成功，返回登录界面
53. **return** redirect(url\_for('login'))
55. #注销
56. @app.route('/logout/')
57. **def** logout():
58. session.pop('user\_id')
59. **return** redirect(url\_for('login'))
61. # 留言
62. @app.route('/leave/' , methods=['GET', 'POST'])
63. @login\_required
64. **def** leave():
65. **if** request.method == 'GET':
66. **return** render\_template('leave.html')
67. **else**:
68. content = request.form.get('message')
69. num = len(content)
70. **if** num <= 0:
71. **return** u"留言内容不能为空。请重新填写。"
72. **elif** num > 150:
73. **return** u"留言内容不能超过150个字符。请重新填写。"
74. **else**:
75. user\_id = session.get('user\_id')
76. user = User.query.filter(User.id == user\_id).first()
77. message = Message(message=content, author=user)
78. db.session.add(message)
79. db.session.commit()
80. **return** redirect(url\_for('index'))
81. # 查找
82. @app.route('/search/')
83. **def** search():
84. q = request.args.get('q')
85. page = request.args.get('page', 1)
86. # 根据username和message查找
87. users = User.query.filter(User.username.contains(q)).all()
88. users\_id =[]
89. **for** user **in** users:
90. users\_id.append(user.id)
91. **if** len(users\_id)==0:
92. users\_id = [0]
93. messages = Message.query.filter(or\_(Message.message.contains(q), Message.author\_id.contains(users\_id))).order\_by(Message.create\_time.desc())
94. paginate = messages.paginate(page=int(page), per\_page=per\_page)
95. **return** render\_template('search.html', paginate=paginate,q=q)
97. # 个人中心、管理自己的留言
98. @app.route('/MyCenter/')
99. **def** MyCenter():
100. page = request.args.get('page',1)
101. user\_id = session.get('user\_id')
102. messages = Message.query.filter(Message.author\_id.contains(user\_id)).order\_by(
103. Message.create\_time.desc())
104. paginate = messages.paginate(page=int(page), per\_page=per\_page)
105. **return** render\_template('private.html', paginate=paginate)
107. # 删除自己的留言
108. @app.route('/delete/')
109. @center\_to\_index
110. **def** delete():
111. message\_id = request.args.get('message\_id')
112. message = Message.query.filter(Message.id == message\_id).first()
113. db.session.delete(message)
114. db.session.commit()
115. **return** redirect(url\_for('MyCenter'))
117. # 钩子函数，上下文
118. @app.context\_processor
119. **def** my\_context\_processor():
120. user\_id = session.get('user\_id')
121. **if** user\_id:
122. user = User.query.filter(User.id == user\_id).first()
123. **if** user:
124. **return** {'user': user}
125. **return** {}
126. 改进和不足

（1）在登录界面和注册界面应当由页面提示错误信息，并且在输入e-mail应当有格式检查。但是由于时间有限，没有再去学习这方面的内容，所以没有实现这些功能。

（2）作为一个独立的网站，应当有管理员，可以管理整个网站，有权限删除其他用户发布的留言。本程序只是留言板网站的简单实现，所以没有添加这个功能。在以后的版本改进中可以添加这个功能。

（3）本程序只是一个开发版本，对于网站在服务器上的部署，没有实现。

1. 参考

[1]Flask教程W3Cschool https://www.w3cschool.cn/flask/