华北电力大学

留言板系统开发设计说明文档

（ 2019--2020 年度第二学期）

名　　称：Python 期末大作业

题　　目：留言板系统

院　　系：控制与计算机工程学院

班　　级：软件 1801

学　　号：120181080416

学生姓名：王子轩

指导教师：熊建国

成　　绩：

日期： 2020 年 6 月 25 日

一、 题目描述

题目 2：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。

二、 正文

1 需求分析

1.1 功能需求

- 用户注册（用户名用于登录和唯一标识发布留言的用户，对用户名和密码数据类型和长度进行限制）
- 用户登录
- 修改密码
- 留言发布（对留言长度和分行上限做了限制）
- 个人留言管理（修改留言、删除留言）
- 留言展示与分页（展示页内容按时间降序排列）
- 分页跳转
- 留言作者与内容的模糊查询

1.2 数据要求

- 数据使用 MySQL 数据库存储
- 数据库存储密码时存储加密后的密码
- 用户名限制在 1 到 15 字，可以包含中文、英文、数字
- 密码限制在 5 到 20 字，可以包含英文、数字

2 系统总体设计

2.1 系统结构与模块设计概述

bbs-demo\

app\

__init__.py

- 创建FLASK对象(flask服务，提供给app\routes.py)
- 创建LoginManager对象(用于用户登录，提供给module\model.py)
- 创建SQLAlchemy对象(数据库相关，提供给module\model.py和module\SQL.py)

config.py 配置flask app、数据库、用户登录秘钥、分页展示等

routes.py 程序运行入口

module\

__init__.py 创建SQL对象，用于app\routes.py中的部分数据库操作

model.py 创建Message和User类，对应数据库中的留言和用户

SQL.py 创建SQL类，提供经过异常处理后的部分数据库操作(如用户注册、插入留言等)

static\

js\

index.js 提供index.html的搜索、跳转页面、基本登录数据校验操作

register.js 提供register.html的注册表单数据校验函数

util.js 提供常用的空字符串校验等函数

search.png 用于美化界面的搜索框图标

templates\

change_password.html 修改密码页

index.html 主页面

profile.html 个人信息页面（包含该用户发布的留言和一些功能按钮）

register.html 用户注册页面

release.html 留言发布页面

search.html 搜索结果展示页面

util\

__init__.py 对原始密码进行加密的函数

2.2 数据库设计

说明：

（1）数据库名：message_board_db

（2）表名：留言表 message_board，用户表 users

（3）数据库配置文件：app\config.py

（4）数据库转储文件：message_board_db.sql

数据库表结构说明：

表 1　留言表结构

字段名称	字段类型	字段约束	备注与说明
id	int	主键、非空、自动递增	留言序号
content	varchar	长度小于 511、utf8 编码	留言内容
username	varchar	长度小于 511、utf8 编码	留言者用户名
time	datetime		留言的创建或更新时间

表 2　用户表结构

字段名称	字段类型	字段约束	备注
user_id	int	主键、非空、自动递增	用户序号
username	varchar	长度小于 255	用户名
password	varchar	长度小于 255	密码，md5 加密字符串
register_date	datetime		用户注册日期

2.3 界面设计

界面为 HTML 页面，均放在 bbs-demo\templates\下。界面包含：

- 主页：index.html，功能包括：分页留言展示、登录、搜索；
- 注册页：register.html，实现用户注册；
- 个人信息页：profile.html，包含该用户发布的留言和功能按钮；
- 留言发布页：release.html，填写留言内容并发布；
- 搜索结果展示页：search.html，展示搜索结果；
- 密码修改页：change_password.html，实现密码修改功能；

3　开发与运行环境

3.1 开发环境

Pycharm Professional 2020.1.2，项目根目录为 bbs-demo.依赖第三方库及版本号已导出至 requirements.txt。

3.2 运行环境

浏览器为 Chrome 浏览器，版本号为 83.0.4103.97。

主机为 Windows 64 位操作系统，基于 x64 处理器。8GB 内存，Intel® Core™ i7-8750H CPU @ 2.20GHz

4　系统部分具体实现

4.1 开发技术

服务端基于 flask 框架，使用 flask_sqlalchemy 进行数据库操作，使用 flask_login 进行用户登录；

4.2 系统界面示意

4.2.1　主页如图 1 所示

功能描述：

- 左下角输入页码可以进行跳转，已经做了数据验证，不合法数据会进行提示；
- 右下角是翻页按钮，当到达边界时自动设置为不可点击；
- 每页展示 12 条留言，在 app\config.py 中的 POSTS_PER_PAGE 进行配置；
- 右上角可以登录，未登录时只有一个登录按钮，点击后可以在弹出的模态框中登录。登录后可以发布留言或进入个人主页；
- 搜索框可以直接搜索无需登录。搜索后弹出新窗口展示搜索结果。对空字符串进行了判断；

4.2.2 个人主页如图 2 所示

图 2 个人信息页

包含所有该用户发布的留言，并能对留言进行编辑修改和删除。上方是功能按键。点击删除后弹出确认删除的模态框，如图 2.1 所示

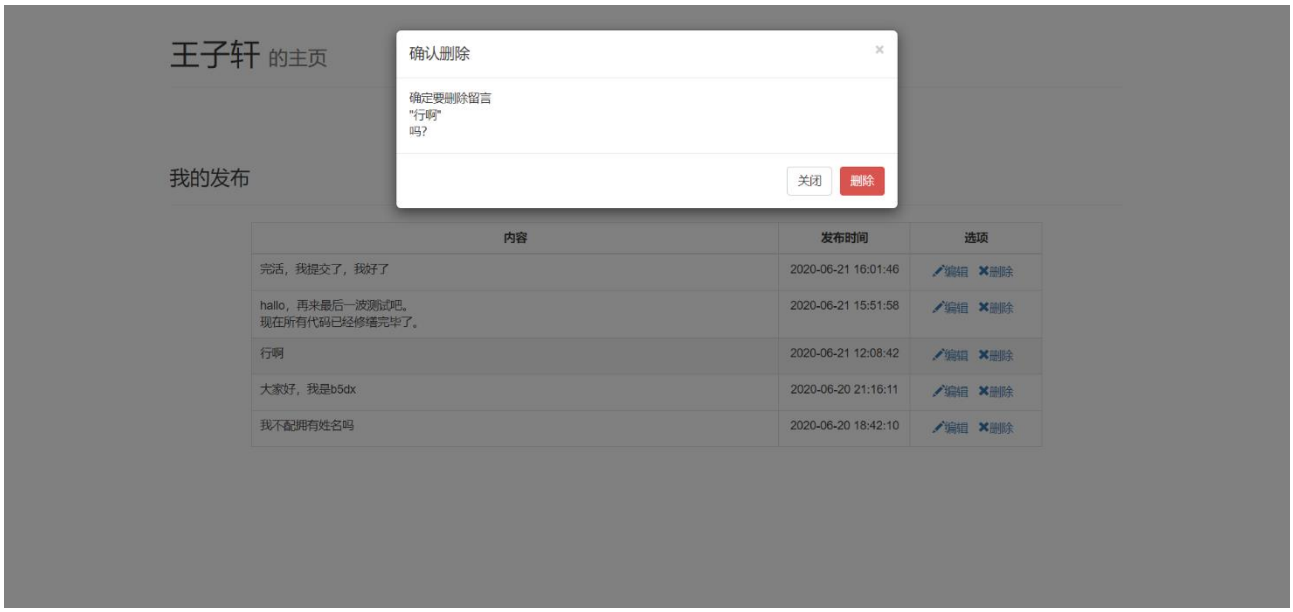

图 2.1 确认删除

4.2.3 登录模态框如图 3 所示

图 3 登录模态框

4.2.4　注册页如图 4 所示

图 4　注册页

输入框中有占位符进行输入内容提示，用 js 进行数据类型和长度的初步验证，通过后传送至服务端和数据库交互，注册成功后回到主页，否则提示错误信息，仍然留在注册页等待再次注册。

4.2.5　留言发布页面如图 5 所示

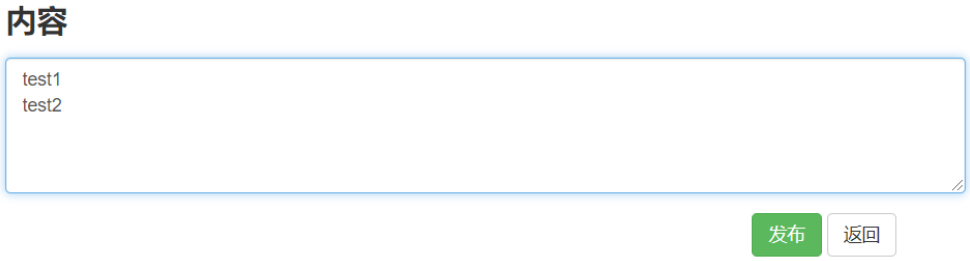

图 5　留言发布页

内容限制 255 字以内，最多手动回车分行 3 次。

4.3 分页留言展示功能的详细设计与实现

```python
@app.route('/index')
def index():
    """
    Homepage
    """
    page = request.args.get('page', 1, type=int)
    message_length = sql.get_message_num()
    total_page = ceil(message_length / app.config['POSTS_PER_PAGE'])
    messages = Message.query.order_by(Message.time.desc()).paginate(
        page, app.config['POSTS_PER_PAGE'], False
    )
    next_url = url_for('index', page=messages.next_num) if messages.has_next else None
    pre_url = url_for('index', page=messages.prev_num) if messages.has_prev else None

    return render_template('index.html', message_list=messages.items, next_url=next_url,
                           pre_url=pre_url, cur_page=page.numerator, total_page=total_page)
```

首先用 request.args.get 获得 url 中的所请求的页面的信息，然后调用 sql 对象（从 module/__init__.py 中 import 进来的 SQL 类的对象）的 get_message_num()获取当前留言的总数，并计算总页面数（用于判断分页边界，控制按钮 disabled 属性和跳转页面的合法范围）。利用 BaseQuery 对象的 paginate 方法获取 Paginate 对象，将 next_url（下一页的请求链接）和 pre_url（上一页的请求链接）生成。最后使用 index.html 模板。传入参数 message_list 消息列表（由 Message 对象组成的 list，Message 对象在 module\model.py 中），也即本页要展示的消息，然后传入两个 url，用于点击 button 时触发翻页事件切换到对应页面。cur_page 和 total_page 用于主页的页码展示。

4.4 用户登录功能的详细设计与实现

```python
@login.user_loader
def load_user(user_id):
    """
    Used for flask_login.
    """
    return User.query.filter_by(user_id=user_id).first()
```

在 module\model.py 中使用 login.user_loader 装饰器装饰 load_user 函数来让 flask_login 中的 login_user(User)正确找到 user。为了使用 user_id 获取 user，需要重载继承了 UserMixin 类的 User 类中的 get_id 方法，如下图所示

```python
class User(UserMixin, db.Model):
    """
    Used for flask_login and SQL command.
    """
    __tablename__ = 'users'

    user_id = db.Column(db.INTEGER(), primary_key=True)
    username = db.Column(db.String(255))
    password = db.Column(db.String(255))
    register_date = db.Column(db.String())

    def __repr__(self):...

    def check_password(self, password):...

    def get_id(self):
        """
        Override the function in UserMixin.
        """
        return self.user_id
```

最后，在 app\routes.py 文件中进行服务端的登录操作，包含获取登录表单信息，验证信息，返回提示信息：

```python
@app.route('/login', methods=['POST'])
def login():
    """
    Route for login.
    Use a modal in index.html to visit.
    :return:
    Redirect to 'index.html'.
    If failed, flash an alert message and return templates 'index.html'.
    """
    if current_user.is_authenticated:
        return redirect(url_for('index'))

    username = request.form['username']
    password = request.form['password']
    user = User.query.filter_by(username=username).first()

    if user is None:
        flash('用户名不存在')
        return redirect(url_for('index'))

    if not user.check_password(password):
        flash('用户名或密码错误')
        return redirect(url_for('index'))

    login_user(user)
    return redirect(url_for('index'))
```

4.5 删除指定留言功能的详细设计与实现

删除留言需要将 message_id 提供给 app\routes.py 中的 delete 函数，通过访问 delete/<message_id> 路径来将数据传送给 flask。delete 函数如图

```python
@app.route('/delete/<message_id>')
def delete(message_id):
    """
    Delete a message by message_id.
    Only the user of the message can delete.
    """
    if not guarantee_user_correct(message_id):
        return '非法访问，你不是这条留言的主人！'

    sql.delete_message(message_id)

    return redirect(url_for('profile'))
```

访问链接前检验是否为留言的拥有者，否则为用户非法访问。确认后调用 sql 的 delete_message 方法通过 message_id 删除留言，然后跳转回个人主页。接下来的问题就是如何把 message_id 从前端传入。

```html
<div class="modal-dialog">
    <div class="modal-content">
        <div class="modal-header">
            <button type="button" class="close" data-dismiss="modal"
                    aria-hidden="true">×</button>
            <h4 class="modal-title" id="myModalLabel">确认删除</h4>
        </div>
        <div class="modal-body">确定要删除留言<br>
            {%- for line in ('"'+i.content+'"').split('\n') -%}
            {{line}}<br>
            {%- endfor -%}
            吗？
        </div>
        <div class="modal-footer">
            <button type="button" class="btn btn-default"
                    data-dismiss="modal">关闭</button>
            <button type="button" class="btn btn-danger"
                    onclick="confirm_delete({{i.id}})">删除</button>
        </div>
    </div><!-- /.modal-content -->
</div><!-- /.modal -->
```

在 profile.html 的删除确认模态框中，利用 jinja 来将对应的 message 传入 js 函数。其中，i 是一个 Message 对象，i.id 获取被点击的删除按钮对应的 message 的 id。此处有红色下划线报错是因为 Pycharm 不能识别 jinja，但是运行时并不会有问题，因为 jinja 会在用户访问到的 HTML 中动态的将此处替换成静态值。

当点击删除按钮确认删除后，confirm_delete 就是改变当前 url 为 delete/<message_id>，也就意味着调用了 routes.py 中的 delete 函数，如果正常操作删除成功后会立刻被重定向到 profile 页面，也就是说在用户眼中点击了确认删除按钮后仿佛刷新了一下个人主页一般，留言已经消失。

4.6 搜索功能的详细设计与实现

"m"的搜索结果

发布者	内容	发布时间
赵鹏飞	mahay, mayiyahay	2020-06-23 17:49:31
赵鹏飞	大家好，我是mahay	2020-06-20 16:50:28
不淡定的但丁	odk，mahay来了我看见了。	2020-06-20 13:02:53
赵鹏飞	我是mahay，mahayhay	2020-06-19 21:42:35
不淡定的但丁	odk，mahay来了我看见了。	2020-06-19 12:42:41

搜索结果页面如上图所示，输入内容并点击搜索后会打开新窗口展示内容，原窗口的搜索框输入内容被清空。若输入空串或者不输入会有提醒。

```python
@app.route('/search')
def search():
    """
    Search a message by keyword and open a new tab to show the results.
    The query message should satisfy:
    keyword in message.username or keyword in message.content.
    :return: template 'search.html'
    """
    keyword = request.args.get('keyword')
    result = sql.search_message(keyword)
    return render_template('search.html', message_list=result, keyword=keyword)
```

routes.py 中的 search 函数进行搜索处理。直接从 url 中使用 request.args.get 方法获得搜索内容，用 sql 对象的 search_message 来进行模糊搜索，只要发布者的用户名和发布内容中含有 keyword 就可以被检索到，检索结果也是按照时间降序排列的。接下来看前端 index.html 中对搜索的实现。

```html
<!--        搜索功能-->
<!--        表单提交到url_for('search'), 使用get方法-->
    <div class="col-sm-8" id="func">
        <div class="col-sm-3"></div>
        <div class="col-sm-1">
            <label for="search_input"><img src="../static/search.png" style="..."></label>
        </div>
        <form role="form" class="form-inline" action="{{ url_for('search') }}" method="get"
              onsubmit="return open_search_window()" id="search_form" name="search_form">
            <input type="text" id="search_input" class="form-control" name="keyword">
            <button class="btn btn-default" type="submit">搜索</button>
        </form>
    </div>
```

表单的 action 参数意味着想 search 函数对应的 url 发起 GET 请求，通过 onsubmit 属性中的 open_search_window 来判断输入内容是否合法，如果合法则提交表单然后打开用于展示搜索结果的新页面 search.html。

三、 总结

1. 总结及体会

本次课程设计基本是从对 web 开发零了解的情况下进行，同时进行 HTML、JavaScript、CSS 的简单了解，也要了解 bootstrap 框架，同时扩充 flask 的知识。总的来说对不了解 web 开发的我来说难度和任务量不小。从一开始的一头雾水，到能够设计出简洁清爽的界面，对前后端的交互有一定的了解，实现简单的功能。对目前的成果比较满意，但如果有更多时间想要扩展更多功能例如留言回复，找回密码操作，个人信息展示等。

2. 难题及解决方案

遇到的有印象的困难和对应的解决方法如下：

- 刚开始不清楚表单的操作，不清楚如何让前后端交互。通过学习 form 的属性，看教学视频来理解表单提交操作。
- 学习 bootstrap 布局，学习 JavaScript 与 HTML 的交互。
- 不清楚如何让用户登录。看文档，看老师上课分享的教程，学习了 flask_login 解决了登录问题。
- 不清楚如何实现分页。为此学习了 paginate 方法。
- 不清楚如何删除指定留言，不懂如何让用户点击删除后确认操作。学习 bootstrap 的模态框使用，想到用 jinja 直接动态生成 HTML 来传输留言 id，对 jinja 有了更深的了解。