

**综合实验报告**

( 2019-- 2020年度第2学期)

实验名称： python课程设计

题 目： 留言板系统

院 系：控制与计算机工程学院

班 级： 软件1802班

学 号： 120181080608

姓 名： 王义

指导教师： 熊建国

成 绩：

日期：2020年6月 24 日

题目名称： 开发设计一个留言板系统

姓名 (学号)：王义 120181080608

一、课程设计目的：

1．进一步掌握和利用python语言进行编程设计的能力；

2．进一步理解和运用结构化程序设计的思想和方法；

3．初步掌握开发一个小型实用系统的基本方法；

4．学会调试一个较长程序的基本方法；

5．掌握书写程序设计开发文档的能力（书写课程设计报告）

二、题目描述

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）

三、功能分析

1、用户注册

2、用户登录

3、登录后分页显示留言信息

4、发布留言

5、根据用户名或留言内容进行模糊查询

6、管理我的留言，能够对留言进行更改和删除操作

四、文件/数据库设计

1．数据库设计

表1 login表的结构

|  |  |  |
| --- | --- | --- |
| 列名 | 类型 | 说明 |
| username | varchar | 存储用户名 |
| password | varcahr | 存储密码 |

表2 mymess表的结构

|  |  |  |
| --- | --- | --- |
| 列名 | 类型 | 说明 |
| id | int | 序号，主键 |
| name | varcahr | 用户名 |
| body | varchar | 用户留言 |
| timestamp | datetime | 留言时间 |

2．界面设计

HTML文件:

login.html ：

用户登录的界面，新用户可以注册账号，登录留言板系统。

register.html ：

新用户注册的界面。

home.html ：

用户登录后的主界面。在此界面分页显示全部留言，用户可以发送留言， 可以进行模糊查询，可以跳转到‘我的留言’界面管理我的留言。

message.html ：

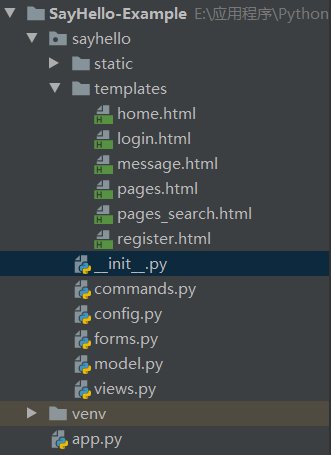
‘我的留言’的界面，在此界面分页显示我的全部留言，用户可以对自 己的留言进行更改和删除的操作。

pages.html、pages\_search.html:

实现界面底部分页按钮的功能

五、系统实现

1. 项目结构



static 静态文件

templates 存模板，包括html文件

\_\_init\_\_.py 初始化操作，构造文件

commands.py 创建数据库表，并生成虚拟数据

config.py 配置文件

forms.py 存放表单

models.py 数据库模型

views.py 视图函数

app.py 主程序

1. 配置文件和创建实例

利用Flask框架，首先需要创建Flask实例，初始化，完成配置等操作。

\_\_init\_\_.py: 创建Flask实例，进行初始化

from flask import Flask  
from flask\_bootstrap import Bootstrap  
from flask\_sqlalchemy import SQLAlchem  
app = Flask('sayhello', static\_folder='static')  
app.config.from\_pyfile('config.py')  
app.jinja\_env.trim\_blocks = True  
app.jinja\_env.lstrip\_blocks = True  
db = SQLAlchemy(app)  
bootstrap = Bootstrap(app)

config.py: 配置文件

import os  
DIALECT = 'mysql' # mysql数据库  
DRIVER = 'mysqlconnector'  
USERNAME = 'root' # root用户  
PASSWORD = 'wangyi200801' # 密码  
HOST = '127.0.0.1'  
PORT = '3306'  
DATABASE = 'test'  
SQLALCHEMY\_DATABASE\_URI = "{}+{}://{}:{}@{}:{}/{}?charset=utf8mb4".format \  
 (DIALECT, DRIVER, USERNAME, PASSWORD, HOST, PORT, DATABASE)  
SQLALCHEMY\_TRACK\_MODIFICATIONS = False  
SECRET\_KEY = os.getenv('SECRET\_KEY', 'secret string')

1. 构造数据库

定义表Message和Login，然后创建数据库表并生成虚拟数据。

model.py:

from datetime import datetime  
from sayhello import db  
# 存储留言信息的表  
class Message(db.Model):  
 \_\_tablename\_\_ = 'mymess'  
 id = db.Column(db.Integer, primary\_key=True)  
 name = db.Column(db.String(50))  
 body = db.Column(db.String(200))  
 timestamp = db.Column(db.DateTime, default=datetime.now, index=True)  
# 存储登录信息的表  
class Login(db.Model):  
 \_\_tablename\_\_ = 'login'  
 username = db.Column(db.String(50, 'utf8\_unicode\_ci'), primary\_key=True)  
 password = db.Column(db.String(50, 'utf8\_unicode\_ci'), nullable=False)

commands.py:创建数据库表并生成虚拟数据

from sayhello import app, db  
from sayhello.model import Message  
from faker import Faker  
try:  
 db.create\_all()  
except Exception as e:  
 print(e)  
# 生成数据，初始化数据库  
fake = Faker()  
for i in range(40):  
 message = Message(  
 name=fake.name(),  
 body=fake.sentence(),  
 timestamp=fake.date\_time\_this\_year()  
 )  
 db.session.add(message)  
db.session.commit()

1. 编写表单类

forms.py:

from flask\_wtf import FlaskForm  
from wtforms import StringField, SubmitField, TextAreaField, PasswordField  
from wtforms.validators import DataRequired, Length  
# 提交留言的表单  
class HelloForm(FlaskForm):  
 body = TextAreaField('Message', validators=[DataRequired(), Length(1, 200)], id='contentcode') #定义id后，控制组件大小  
 submit = SubmitField('发送')  
# 登录的表单  
class LoginForm(FlaskForm):  
 username = StringField("username", validators=[DataRequired(), Length(1, 20)])  
 password = PasswordField("password", validators=[DataRequired(), Length(1, 20)])  
# 注册的表单  
class RegisterForm(FlaskForm):  
 username = StringField("username", validators=[DataRequired(), Length(1, 20)])  
 password = PasswordField("password", validators=[DataRequired(), Length(1, 20)])  
 againP = PasswordField("password", validators=[DataRequired(), Length(1, 20)])

1. 编写视图函数

views.py:

1.登录功能的实现

@app.route('/', methods=['GET', 'POST'])  
def login():  
 loginform = LoginForm()  
 if request.method == 'POST':  
 username = loginform.username.data  
 password = loginform.password.data  
 try:  
 exist = db.session.query(Login).filter(Login.username == username, Login.password == password)  
 user\_exists = db.session.query(exist.exists()).scalar()  
 if user\_exists == True: # 用户名和密码正确  
 return redirect(url\_for('home', username=username, page=int(1)))  
 return '''<a type="submit" href="/">用户名或密码错误，请重新登录！</a>'''  
 except Exception:  
 return '''<a type="submit" href="/">登录失败，请重新登录！</a>'''  
 return render\_template('login.html', form=loginform)

2.注册功能的实现

@app.route('/register', methods=['GET', 'POST'])  
def register():  
 registerform = RegisterForm()  
 if request.method == 'POST':  
 username = registerform.username.data  
 password = registerform.password.data  
 againP = registerform.againP.data  
 try:  
 exist = db.session.query(Login).filter(Login.username == username)  
 name\_exists = db.session.query(exist.exists()).scalar()  
 if name\_exists == True: # 用户名已存在  
 return render\_template('register.html', form=registerform, umessage='账号已存在！')  
 elif againP != password: # 确认密码错误  
 return render\_template('register.html', form=registerform, pmessage='密码错误，请再次输入！')  
 else:  
 user = Login(username=username, password=password)  
 db.session.add(user)  
 db.session.commit()  
 return '''  
 <a type="submit" href="/">返回登录界面</a>  
 '''  
 except Exception as e:  
 return render\_template('register.html', form=registerform)  
 return render\_template('register.html', form=registerform)

3.主页，分页显示留言

@app.route('/home', methods=['GET', 'POST'])  
def home():  
 helloform = HelloForm()  
 username = request.args.get('username')  
 page = int(request.args.get('page'))  
 searchname = request.args.get('searchname')  
 searchcontent = request.args.get('searchcontent')  
 try:  
 if helloform.validate\_on\_submit(): # 发送信息  
 body = helloform.body.data  
 message = Message(body=body, name=username)  
 db.session.add(message)  
 db.session.commit()  
 return redirect(url\_for('home', username=username, page=1))  
 if request.method == 'POST': # 模糊查询  
 searchname = request.form.get('username')  
 searchcontent = request.form.get('content')  
 return redirect(url\_for('home', username=username, page=1, searchcontent=searchcontent, searchname=searchname))  
 if searchcontent!=None and searchname!=None: # 模糊查询处理  
 if searchname == '':  
 if searchcontent != '':  
 blogs = Message.query.filter(Message.body.like('%{}%'.format(searchcontent))).order\_by(  
 Message.timestamp.desc()).paginate(page=page, per\_page=10)  
 else:  
 blogs = Message.query.order\_by(Message.timestamp.desc()).paginate(page=page, per\_page=10)  
 elif searchcontent == '':  
 blogs = Message.query.filter(Message.name.like('%{}%'.format(searchname))).order\_by(  
 Message.timestamp.desc()).paginate(page=page, per\_page=10)  
 else:  
 blogs = Message.query.filter(and\_(Message.body.like('%{}%'.format(searchcontent)), \  
 Message.name.like('%{}%'.format(searchname)))).order\_by(  
 Message.timestamp.desc()).paginate(page=page, per\_page=10)  
 return render\_template('home.html', blogs=blogs.items, pagination=blogs, form=helloform, username=username,\  
 searchname=searchname, searchcontent=searchcontent)  
 except Exception:  
 return redirect(url\_for('home', username=username, page=1))  
 blogs = Message.query.order\_by(Message.timestamp.desc()).paginate(page=page, per\_page=10)  
 return render\_template('home.html', blogs=blogs.items, pagination=blogs, form=helloform, username=username)

4.管理我的留言

@app.route('/home/mymessage', methods=['GET', 'POST'])  
def mymess():  
 username = request.args.get('username')  
 page = int(request.args.get('page'))  
 mess\_id = request.args.get('messid')  
 dele = request.args.get('dele')  
 try:  
 if request.method == 'POST': # 提交修改后的信息  
 content = request.form.get('content')  
 update\_id = db.session.query(Message).filter(Message.id == mess\_id).one()  
 update\_id.body = content  
 db.session.commit()  
 return redirect(url\_for('mymess', username=username, page=page))  
 if dele: # 删除该条信息  
 exist = db.session.query(Message).filter(Message.id == mess\_id)  
 id\_exists = db.session.query(exist.exists()).scalar()  
 if id\_exists == True:  
 del\_id = exist.one()  
 db.session.delete(del\_id)  
 db.session.commit()  
 return redirect(url\_for('mymess', username=username, page=page))  
 except:  
 return redirect(url\_for('mymess', username=username, page=page))  
 blogs = Message.query.filter(Message.name == username).order\_by \  
 (Message.timestamp.desc()).paginate(page=page, per\_page=10)  
 return render\_template('message.html', messages=blogs.items, pagination=blogs, username=username, page=page)

1. 测试

此系统使用mysql数据库，使用时需要到config.py文件中修改密码。

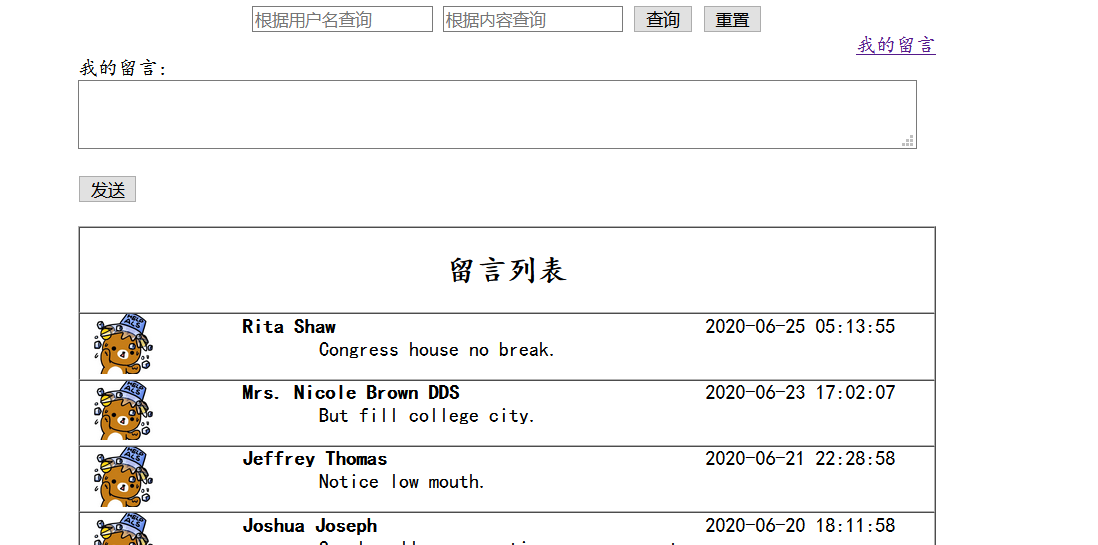
1. 输入网址‘http://127.0.0.1:5000/’进入留言板登录界面



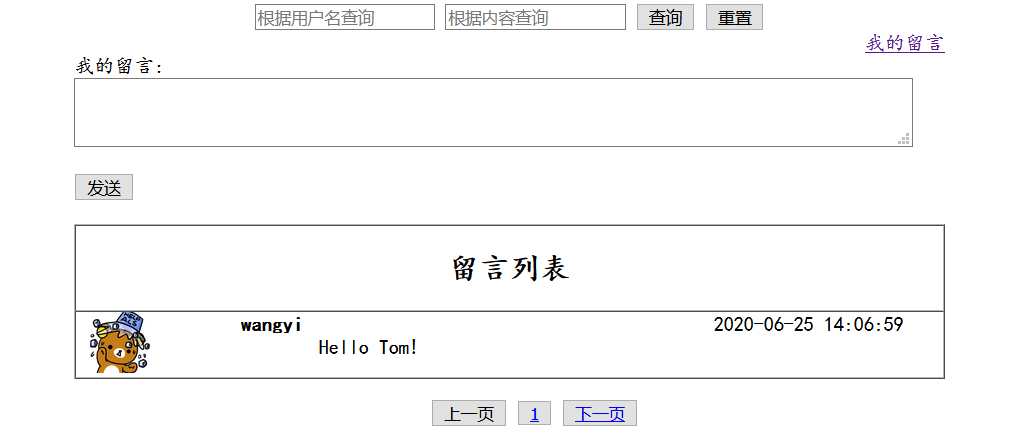
1. 注册一个新的账号



1. 注册成功后，登录留言板系统，进入主页



1. 发送数据‘Hello Tom！’，并查询，结果如下：



1. 点击‘我的留言’，管理我的留言，修改留言时，先点击‘更改’，再 在留言修改区写入修改后的留言，删除时直接点击‘删除’即可。

