课程作业

课程名称	：	**数据库系统原理**
作业次数	：	**作业#5**
学号	：	**21281280**
姓名	：	**柯劲帆**
班级	：	**物联网2101班**
指导老师	：	**郝爽**
修改日期	：	**2024年5月12日**

---

1. 实验案例描述

本实验我实现的案例是12306系统的用户注册、修改账号信息。

注册需要用户填写：

- **用户名**：设置成功后不可修改
- **登陆密码**：6-20位字母数字或符号
- **确认密码**：须与登陆密码一致
- **姓名**
- **身份证号**：需满足身份证号格式
- **手机号码**：需满足手机号码格式

支持**注销账号**及修改：

- **登陆密码**
- **手机号码**

2. 实验过程

2.1. 项目架构设计

对于前端，我使用 `B/S架构` 和原生的 `HTML5` + `CSS` + `JavaScript` 代码实现。

对于后端，我使用 `Python` + `Flask` 作为Web服务器和应用服务器，并且调用相同服务器中的 `MySQL` 实现任务。

2.2. 项目实现

对于前后端的代码实现，请见附件。

配置环境方面，我新建了 `conda` 环境，并使用 `pip` 安装了项目所需依赖，包括 `Flask` 、 `PyMySQL` 等。依赖列表我已导出至附件中的 `requirements.txt` 。

另外，为了快速启动服务，我编写了启动脚本 `run.sh` ，见附件。

最终目录树如下：

```
$ tree
.
├── db.sql
├── init_db.py
├── main.py
├── requirements.txt
├── run.sh
├── static
|   ├── css
|   |   └── style.css
|   └── js
|       ├── checkInfo.js
|       └── modify.js
└── templates
    ├── index.html
    ├── modify.html
    └── signup.html

4 directories, 11 files
```

3. 实验结果

3.1. 前端

本实现实现了 3 个前端页面：

- 主页
- 注册
- 修改账号信息

3.1.1. 主页

主页主要提供 *注册* 和 *修改账号信息* 两个超链接的跳转。

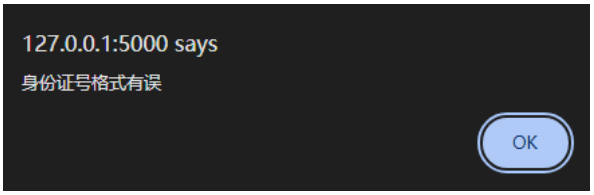
3.1.2. 注册页面

该页面将对输入进行如下检查：

- **检查身份证格式**

```
checkInfo.checkCardCode = function() {
    let cardCode = document.getElementById('cardCode').value
    let regexCardCode = /^([1-6][1-9]|50)\d{4}(18|19|20)\d{2}((0[1-9])|10|11|12)(([0-2][1-9])|10|20|30|31)\d{3}[0-9Xx]$/
    if(!regexCardCode.test(cardCode)) {
        alert('身份证号格式有误')
        return false
    }
    return true
}
```

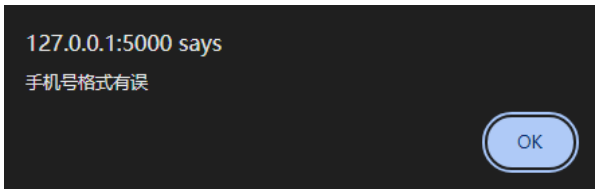
如果格式不正确，将弹出警告：



- **检查手机号格式**

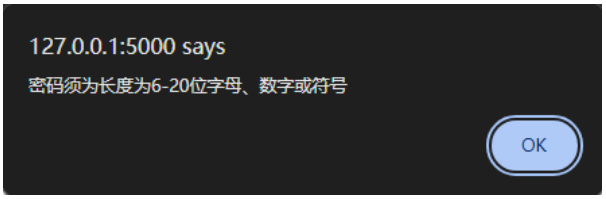
```
checkInfo.checkMobileNo = function() {
    let mobileNo = document.getElementById('mobileNo').value
    let regexMobileNo = /^1[3-9]\d{9}$/
    if (!regexMobileNo.test(mobileNo)) {
        alert('手机号格式有误')
        return false
    }
    return true
}
```

如果格式不正确，将弹出警告：



- **检查密码格式**

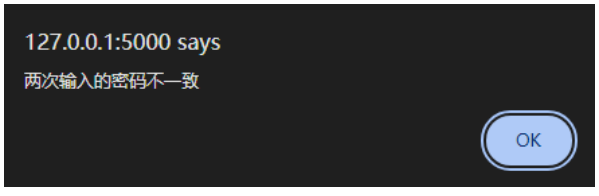
  如果格式不正确，将弹出警告：



- **检查确认密码是否与第一次输入的密码匹配**

```
checkInfo.checkPassword = function() {
    let password = document.getElementById('password')
    let regexPassword = /^[A-Za-z0-9\W_]{6,20}$/
    if (!regexPassword.test(password.value)) {
        alert("密码须为长度为6-20位字母、数字或符号")
        return false
    }
    let confirmPassword = document.getElementById('confirmPassword')
    if (password.value !== confirmPassword.value) {
        alert("两次输入的密码不一致")
        return false
    }
    return true
}
```

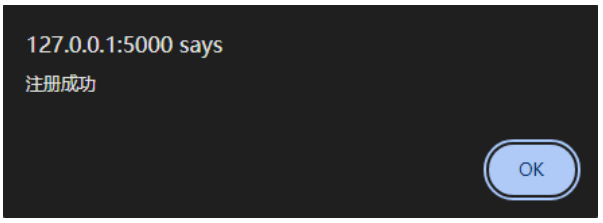
如果不匹配，将弹出警告：



页面会对密码使用 `md5码` 加密传输至后端（效果见 `3.2.1.2. 注册页面请求处理函数` 部分的数据库查询结果截图）。

```
document.getElementById('encryptedPassword').value = md5(
    document.getElementById('password').value
)
```

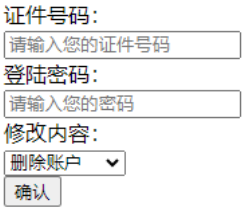
注册成功后，页面跳转回主页，并弹出注册成功通知：



3.1.3. 修改账户信息页面

修改账户信息页面可以选择三种修改方式，通过点击下拉框即可选择：

- 删除账户：



- 修改密码：

- 修改手机号码：

这里使用了 Javasript 代码操作列表的 `display` 属性，从而控制列表项的显示：

```
modify.showModifyPassword = function() {
    let modifyType = document.getElementById('modifyType').value
    let info = {
        modifyPasswordLis: document.getElementsByClassName('modifyPassword'),
        modifymobileNoLis: document.getElementsByClassName('modifymobileNo'),
    }
```

```
    // 遍历隐藏所有元素
    for (let key in info) {
        let elements = info[key];
        for (let item of elements) {
            item.style.display = 'none'; // 确保所有相关元素被隐藏
        }
    }
    // 根据 modifyType 显示相关元素
    if (modifyType === "2") {
        for (let item of info.modifyPasswordLis) {
            item.style.display = 'block';
        }
    } else if (modifyType === "3") {
        for (let item of info.modifymobileNoLis) {
            item.style.display = 'block';
        }
    }
}
```

上述修改都会在后端检查证件号码和登录密码是否匹配（见后端部分实验结果）。

修改密码将会做密码二次匹配检查、密码格式检查，修改手机号将会做手机号格式化检查。弹出的提示和注册页面提示一样，这里不作赘述。

3.2. 后端

3.2.1. Flask 框架

Flask 框架负责作为 Web 服务器、应用服务器的实现方式。

我在代码代码中实现了一个函数，用于连接数据库，以便后续获取 cursor 以及关闭连接。

```
def get_db():
    return pymysql.connect(
        host='localhost', user='kejingfan',
        password='PASSWORD', database='TESTDB'
    )
```

对于每一个前端页面，都需要在 Flask 的调用代码中编写一个函数来处理该页面的 GET / POST 请求。

3.2.1.1. 主页请求处理函数

主页目前只有 GET 请求，因此只需要返回 `index.html` 即可。

```
@app.route("/")
def index():
    return render_template("index.html")
```

3.2.1.2. 注册页面请求处理函数

注册页面中，收到 GET 请求，返回 `signup.html` ；

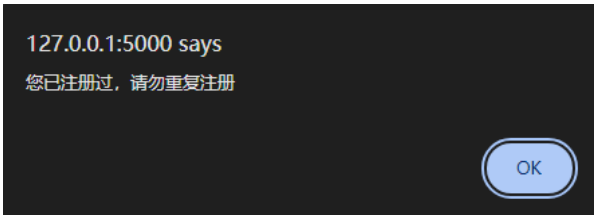
```python
@app.route("/signup.html", methods=('GET', 'POST'))
def signup():
    if request.method == 'GET':
        return render_template('signup.html')
```

收到 POST 请求，需要处理返回的表单。表单一共有 4 项：`'cardCode', 'name', 'mobileNo', 'encryptedPassword'`。

```python
if request.method == 'POST':
    id = request.form['cardCode']
    name = request.form['name']
    phone_number = request.form['mobileNo']
    password = request.form['encryptedPassword']

    db = get_db()
    cursor = db.cursor()
```

首先使用身份证号 `'cardCode'` 项，对数据库进行查询，检查用户是否正在重复注册。

若重复注册，弹出：




```python
sql = """
SELECT COUNT(*) FROM passengers \
WHERE ID = %s;
"""
try:
    cursor.execute(sql, (id,))
    id_exist = cursor.fetchall()[0][0]
except Exception as e:
    flash("数据库异常，查询失败")
    print(e)
    return redirect(url_for('signup'))
if (id_exist != 0):
    flash("您已注册过，请勿重复注册")
    db.close()
    return redirect(url_for('index'))
```

如果没有检索到已注册信息，则插入注册信息：

```python
sql = '''
INSERT INTO passengers (ID, `Name`, Phone_number, `Password`) \
VALUES (%s, %s, %s, %s); \
'''
try:
    cursor.execute(sql, (id, name, phone_number, password))
    db.commit()
    flash("注册成功")
```

```
except Exception as e:
    db.rollback()
    print(e)
    flash("数据库异常，注册失败")
db.close()
return redirect(url_for('index'))
```

此时数据库可以查询到：

```
mysql> select * from passengers;
+--------------------+-------------+--------------+----------------------------------+
| ID                 | Name        | Phone_number | Password                         |
+--------------------+-------------+--------------+----------------------------------+
| 440902200307130011 | '柯劲帆'    |  19120711787 | 'e10adc3949ba59abbe56e057f20f883e' |
+--------------------+-------------+--------------+----------------------------------+
1 row in set (0.00 sec)
```

3.2.1.3. 修改账户信息页面请求处理函数

修改账户信息页面中，收到 GET 请求，返回 `modify.html` ；

```
@app.route("/modify.html", methods=('GET', 'POST'))
def modify():
    if request.method == 'GET':
        return render_template('modify.html')
```

收到 POST 请求，需要处理返回的表单。表单一共有 5 项：

`'cardCode', 'encryptedPassword', 'modifyType', ''encryptedNewPassword'', 'mobileNo'`

。

首先需要验证用户是否存在，且密码正确。这里需要做两个查询操作，分别查询 `ID` 的数量以及 `Password` 的值。

将验证过程封装成函数：

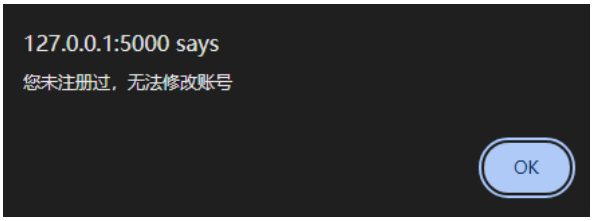
```
def verify_user(cursor:Cursor, id:str, password:str) -> str:
    # 检查已有用户
    sql = """
    SELECT COUNT(*) FROM passengers \
    WHERE ID = %s;
    """
    try:
        cursor.execute(sql, (id,))
        id_exist = cursor.fetchall()[0][0]
    except Exception as e:
        flash("数据库异常，查询失败")
        print(e)
        return redirect(url_for('signup'))
    if (id_exist == 0):
        return "NO_USER"

    # 检查密码
    sql = """
    SELECT `Password` FROM passengers \
    WHERE ID = %s;
    """
    try:
```

```
        cursor.execute(sql, (id,))
        record_password = cursor.fetchall()[0][0]
    except Exception as e:
        flash("数据库异常，查询失败")
        print(e)
        return redirect(url_for('modify'))
    if (record_password != password):
        return "WRONG_PASSWORD"

    return "USER_VERIFIED"
```

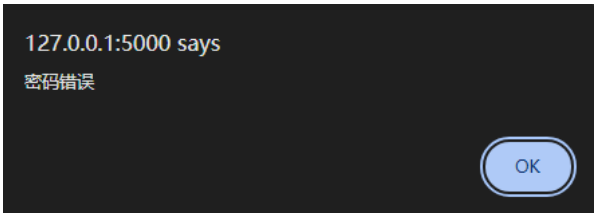
判断验证结果，作出响应：

```
id = request.form['cardCode']
password = request.form['encryptedPassword']
db = get_db()
cursor = db.cursor()

verify_info = verify_user(cursor, id, password)
if (verify_info == "NO_USER"):
    flash("您未注册过，无法修改账号")
    db.close()
    return redirect(url_for('signup'))
elif (verify_info == "WRONG_PASSWORD"):
    flash("密码错误")
    db.close()
    return redirect(url_for('modify'))
```

如果用户没有注册：



如果密码错误：



验证通过后，需要根据请求对数据库进行更改。

编写了一个类专门用来处理修改内容、数据库修改指令和提示信息：

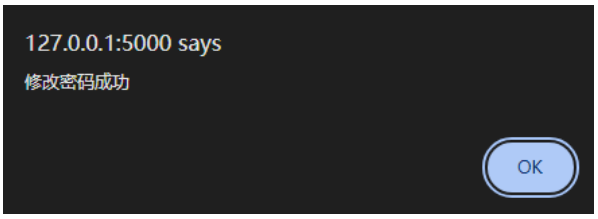
```
class ModifyInfo:
    def __init__(self, form:Dict[str, str]):
        self.id = form['cardCode']
        modifyType = form['modifyType']
        self.new_password = form['encryptedNewPassword']
        self.phone_number = form['mobileNo']
        modifyType2command = {
```

```
            '1':'delete account',
            '2':'modify Password',
            '3':'modify Phone_Number'
        }
        self.sql_dict = {
            'delete account': 'DELETE FROM passengers WHERE ID = %s;',
            'modify Password': 'UPDATE passengers SET `Password` = %s WHERE ID = %s;',
            'modify Phone_Number': 'UPDATE passengers SET Phone_number = %s WHERE ID = %s;'
        }
        self.sql_args_dict = {
            'delete account': (self.id,),
            'modify Password': (self.new_password, self.id),
            'modify Phone_Number': (self.phone_number, self.id)
        }
        self.ok_message_dict = {
            'delete account': "删除账户成功",
            'modify Password': "修改密码成功",
            'modify Phone_Number': "修改手机号成功"
        }
        self.fail_message_dict = {
            'delete account': "数据库异常，删除账户失败",
            'modify Password': "数据库异常，修改密码失败",
            'modify Phone_Number': "数据库异常，修改手机号失败"
        }
        self.command = modifyType2command[modifyType]
    def get_sql(self):
        return self.sql_dict[self.command]
    def get_args(self):
        return self.sql_args_dict[self.command]
    def get_ok_message(self):
        return self.ok_message_dict[self.command]
    def get_fail_message(self):
        return self.fail_message_dict[self.command]
```

在处理函数中创建该类对象，并调用该对象方法获取操作参数：

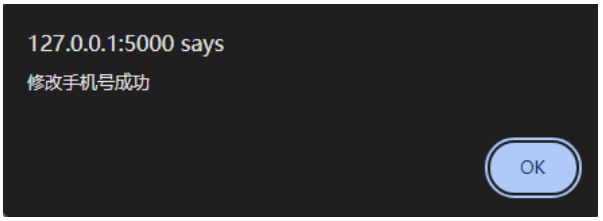
```
modifyInfo = ModifyInfo(request.form)
try:
    cursor.execute(modifyInfo.get_sql(), modifyInfo.get_args())
    db.commit()
    flash(modifyInfo.get_ok_message())
except Exception as e:
    db.rollback()
    print(e)
    flash(modifyInfo.get_fail_message())
db.close()
return redirect(url_for('index'))
```

修改密码成功后：



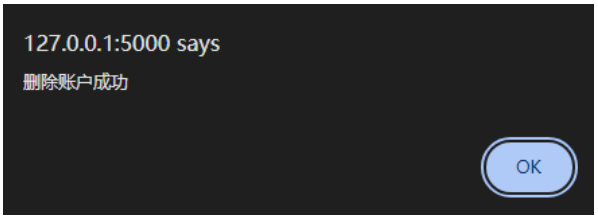

```
mysql> select * from passengers;
+-------------------+----------+--------------+----------------------------------+
| ID                | Name     | Phone_number | Password                         |
+-------------------+----------+--------------+----------------------------------+
| 440902200307130011 | 柯劲帆    |  19120711787 | e10adc3949ba59abbe56e057f20f883e |
+-------------------+----------+--------------+----------------------------------+
1 row in set (0.00 sec)

mysql> select * from passengers;
+-------------------+----------+--------------+----------------------------------+
| ID                | Name     | Phone_number | Password                         |
+-------------------+----------+--------------+----------------------------------+
| 440902200307130011 | 柯劲帆    |  19120711787 | fcea920f7412b5da7be0cf42b8c93759 |
+-------------------+----------+--------------+----------------------------------+
1 row in set (0.00 sec)
```

修改手机号成功后：




```
mysql> select * from passengers;
+-------------------+----------+--------------+----------------------------------+
| ID                | Name     | Phone_number | Password                         |
+-------------------+----------+--------------+----------------------------------+
| 440902200307130011 | 柯劲帆    |  19120711787 | fcea920f7412b5da7be0cf42b8c93759 |
+-------------------+----------+--------------+----------------------------------+
1 row in set (0.00 sec)

mysql> select * from passengers;
+-------------------+----------+--------------+----------------------------------+
| ID                | Name     | Phone_number | Password                         |
+-------------------+----------+--------------+----------------------------------+
| 440902200307130011 | 柯劲帆    |  15218302829 | fcea920f7412b5da7be0cf42b8c93759 |
+-------------------+----------+--------------+----------------------------------+
1 row in set (0.00 sec)
```

删除账号成功后：




```
mysql> select * from passengers;
+-------------------+----------+--------------+----------------------------------+
| ID                | Name     | Phone_number | Password                         |
+-------------------+----------+--------------+----------------------------------+
| 440902200307130011 | 柯劲帆    |  15218302829 | fcea920f7412b5da7be0cf42b8c93759 |
+-------------------+----------+--------------+----------------------------------+
1 row in set (0.00 sec)

mysql> select * from passengers;
Empty set (0.01 sec)
```

3.2.2. 数据库服务器

本实验使用 MySQL 作为数据库。

编写了一个 SQL 脚本和一个 Python 文件用于初始化测试数据库：

```sql
DROP TABLE IF EXISTS passengers;

CREATE TABLE passengers (
    ID BIGINT PRIMARY KEY,
    `Name` VARCHAR (255) NOT NULL,
    Phone_number BIGINT UNIQUE NOT NULL,
    `Password` VARCHAR (255) NOT NULL,
    CHECK (ID REGEXP '^\\d{18}$'),
    CHECK (Phone_number REGEXP '^\\d{11}$')
);
```

```python
import pymysql

db = pymysql.connect(
    host='localhost', user='kejingfan',
    password='PASSWORD', database='TESTDB'
)

cursor = db.cursor()

with open('db.sql', 'r') as f:
    sql_commands = f.read().split(';')
    for command in sql_commands:
        if command.strip():  # 确保不执行空命令
            cursor.execute(command)

db.close()
```