轻量级 J2EE 框架应用

E 5　A Simple Controller with DAO pattern

学号：SA17225052

姓名：戴赛

报告撰写时间：2017/12/30

1.主题概述

代码逻辑整理:

SimpleController 的 BaseDAO 中:

为了能够重复使用,该抽象类中的属性应都与数据库属性有关,与 Entity 相关的属性应该在实现该抽象类时再定义.

属性(都有 set 与 get 方法):

数据库驱动

数据库路径

数据库用户名

数据库密码

方法:

打开数据库连接(返回 Connection 类)

关闭数据库连接(返回布尔型,其实用 void 应该也可以)

增删改查(四个抽象方法)

UseSC 中:

UserDAO(继承自 SC 中的 BaseDAO):

在 BaseDAO 的基础上增加与 Entity 相关的部分

增加属性(都有 set 与 get 方法):

用户名

用户密码

用户 ID

构造方法:

初始化数据库驱动 数据库路径 数据库用户名 数据库密码(利用 set 和 get)

实现方法:

增查(具有完整逻辑)

删改(空方法)

增加方法:

判断某表是否存在(布尔型)

创建某表(void)

UserBean:

将 Action 的信息转发至 DAO

属性:

用户名

用户密码

用户 ID

数据库地址 数据库驱动 数据库账号 数据库密码(有 set 与 get 方法)

构造方法:

初始化用户名 用户密码 用户 ID

方法:

登陆方法 signIn()

注册方法 signUp()

2.假设

上次作业 E4 及 MySQL

3.实现或证明

1.

SimpleController 中定义 BaseDAO

(属性及 setter 方法)

```
protected String db_userName; // 数据库的用户和密码,别与实体类搞混了
protected String db_userPassword;
protected String driver; // 驱动
protected String url; // 数据库地址
```

```
public String getDb_userName() { return db_userName; }

public void setDb_userName(String db_userName) { this.db_userName = db_userName; }

public String getDb_userPassword() { return db_userPassword; }

public void setDb_userPassword(String db_userPassword) { this.db_userPassword = db_userPassword; }

public String getDriver() { return driver; }

public void setDriver(String driver) { this.driver = driver; }

public String getUrl() { return url; }

public void setUrl(String url) { this.url = url; }
```

两个实现方法

```
// 打开数据库连接
protected Connection openDBConnection()
{
    Connection conn =null;
    System.out.println("创建数据库连接,加载驱动:");
    try {
        // 加载驱动
        Class.forName(driver);
        if(conn==null)
        {
            try {
                // System.out.println("000000"+url+";"+userName+";"+userPassword+"000000");
                conn = DriverManager.getConnection(url,db_userName,db_userPassword);
            } catch (SQLException e) {
                System.out.println("    创建链接失败:"+e);
                // e.printStackTrace();
            }
        }else
        {
            System.out.println("    BaseDAO逻辑错误:数据链接对象非空");
        }
    } catch (ClassNotFoundException e) {
        System.out.println("    加载驱动失败:" + e);
        // e.printStackTrace();
    }
    return conn;
}
```

```java
// 关闭数据库连接
protected boolean closeDBConnection(Connection connection, PreparedStatement pstmt, ResultSet rs)
{
    // conn数据库链接对象
    // rs数据返回结果,有可能为空
    // pstmt数据库命令执行对象
    System.out.println("关闭数据库连接");
    try {
        if (rs!=null)
        {
            rs.close();
            // rs = null;
        }
        if (pstmt!=null)
        {
            pstmt.close();
            // pstmt = null;
        }
        if (connection!=null)
        {
            connection.close();
            // connection = null;
        }
        System.out.println("    数据库连接关闭成功");
    }catch (SQLException e)
    {
        System.out.println("    数据库关闭失败,错误为:"+e);
        return false;
    }
    return true;
}
```

四个抽象方法

```java
// 增删改查,四个抽象方法
public abstract Object query(String sql);
public abstract boolean insert(String sql,String tableName);
public abstract boolean update(String sql);
public abstract boolean delete(String sql);
```

2.

LoginAction 中增加 UserBean 内容

```java
public class LoginAction {
    public String handleLogin(String name,String pwd) {
        //  Login中根据传入的name与pwd进行判断,跳转,判断成功则返回success,失败则返回failure
//        if (name.equals("tom") && pwd.equals("123")) {
//            System.out.println("****************************************");
//            System.out.println("LoginAciton结果是success");
//            System.out.println("****************************************");
//            return "success";
//        } else {
//            System.out.println("****************************************");
//            System.out.println("LoginAciton结果是failure");
//            System.out.println("****************************************");
//            return "failure";
//        }
        String id = "112"; // ID写死
        UserBean userBean = new UserBean(id, name, pwd);
        userBean.setUrl("jdbc:mysql://127.0.0.1:3306/USTC");
        userBean.setDriver("com.mysql.jdbc.Driver");
        userBean.setDb_userName("USTC");
        userBean.setDb_userPass("123");
        if(userBean.signIn())
        {
            return "success";
        }else {
            return "failure";
        }
```

UserBean

```java
public class UserBean {
    private String userID;
    private String userName;
    private String userPass;

    private String url; // 需要在Action中单独设置
    private String driver; // 需要Action中单独设置
    private String db_userName; // 同上
    private String db_userPass;



    public UserBean(String uID, String name, String pass)
    {
        this.userID = uID;
        this.userName = name;
        this.userPass = pass;
    }

    public boolean signIn()
```

```java
public boolean signIn()
{
    UserDAO userDAO = new UserDAO(db_userName, db_userPass, url, driver);
    userDAO.setUserName(userName);
    userDAO.setUserPassword(userPass);
    String sql = "SELECT *   FROM USER   WHERE  USERNAME=? and USERPASS=? and USERID="+userID;
    System.out.println("-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-");
    System.out.println("执行signIn()");
    // 判断userDAO的query方法,并取出ID与PASS,考虑到可能会有重名的情况,同时要匹配ID
    // HashMap queryResult = (HashMap) userDAO.query(sql);
    ArrayList queryResultList = (ArrayList) userDAO.query(sql);
    if (queryResultList == null)
    {
        System.out.println("    signIn()中query()的返回值为空,signIn失败");
        System.out.println("-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-");
        return false;
    }else {
        // 遍历List
        for (Object queryresult:queryResultList) {
            HashMap queryresult_hm = (HashMap) queryresult;
            // 取出userID和PASS
            String queryUserID = (String) queryresult_hm.get("userID");
            String queryUserPass = (String) queryresult_hm.get("userpass");

            if (userID.equals(queryUserID) && userPass.equals(queryUserPass))
            {
                System.out.println("            ID匹配成功,signIn成功");
                System.out.println("-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-");
                return true;
            }
        }

        System.out.println("            ID匹配失败~页面ID为:"+userID+";query返回ID为:");
        System.out.println("            ID匹配失败~页面pass为:"+userPass);
        System.out.println("-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-");
        return false;
    }
}
```

3.

UserDAO

```
public UserDAO(String dbun, String dbpwd, String url, String driver)
{
    // 在构造方法中初始化父类域(利用Setter方法)
    // this.setUserName(un);
    // this.setUserPassword(pwd);
    this.setDb_userName(dbun);
    this.setDb_userPassword(dbpwd);
    this.setUrl(url);
    this.setDriver(driver);
}

// 用户的账号和密码
private String userName;
private String userPassword;
```

Query 方法

```
// 重写query
@Override
public Object query(String sql) {
    // "SELECT *   FROM USER   WHERE  sname=? and pwd=?"
    System.out.println("对语句'"+sql+"'执行query():");
    // 打开连接
    Connection connection = this.openDBConnection();
    // 返回结果
    ArrayList<HashMap<String,String>> resultHashMapList = new ArrayList<~>();

    try {
        PreparedStatement preparedStatement = connection.prepareStatement(sql); // 预编译
        preparedStatement.setString( parameterIndex: 1,userName);
        preparedStatement.setString( parameterIndex: 2,userPassword);

        ResultSet resultSet = preparedStatement.executeQuery();
        System.out.println("        返回的resultSet结果为:" + resultSet.toString());

        while (resultSet.next())
        {
            // 获取表单中所有结果
            HashMap<String, String> queryResult = new HashMap<>();

            String result1 = resultSet.getString( columnLabel: "USERNAME");
            String result2 = resultSet.getString( columnLabel: "USERPASS");
            String result3 = resultSet.getString( columnLabel: "USERID");
            System.out.println("        返回结果中的用户名:"+result1+"密码为:"+result2+"ID为:"+result3);
            queryResult.put("username",result1);
            queryResult.put("userpass",result2);
            queryResult.put("userID",result3);

            resultHashMapList.add(queryResult);
        }

        // 关闭连接
        boolean closeResult = this.closeDBConnection(connection, preparedStatement, resultSet);
        if (closeResult) System.out.println("    query()关闭成功");
    } catch (SQLException e) {
        System.out.println("    query()方法出错:" + e);
    }

    return resultHashMapList;
}
```

4.

为了能够实现完整的代码逻辑,改写 RegisterAction,在 UserBean 中增加注册 signUp 方法,signUp 调用 UserDAO 中的 insert 方法(也可能会用到 createTable 方法创建表格)将注册的用户信息输入数据库

Action

```
public class RegisterAction {

    public String handleRegister(String userName, String pwd) {

        String id = "112"; // ID写死
        UserBean userBean = new UserBean(id, userName, pwd);
        userBean.setUrl("jdbc:mysql://127.0.0.1:3306/USTC");
        userBean.setDriver("com.mysql.jdbc.Driver");
        userBean.setDb_userName("USTC");
        userBean.setDb_userPass("123");
        if (userBean.signUp())
        {
            return "success";
        }
        else {
            return "handleRegister()出错";
        }
    }
}
```

UserBean 的 signUp

```
public boolean signUp()
{
    // 注册
    // 具体逻辑:在UserDAO中完成插入
    // 插入的内容为name,pass,id
    System.out.println("-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-");
    System.out.println("执行signUp()");
    String tableName = "USER";
    String sql="INSERT INTO "+tableName+"(USERNAME,USERPASS,USERID)  VALUES(?,?,"+userID+")";
    UserDAO userDAO = new UserDAO(db_userName, db_userPass, url, driver);
    userDAO.setUserName(userName);
    userDAO.setUserPassword(userPass);
    if(userDAO.insert(sql,tableName))
    {
        System.out.println("    signUp()执行成功");
        System.out.println("-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-");
        return true;
    }
    else {
        System.out.println("   signUp()执行失败");
        System.out.println("-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-");
        return false;
    }
}
```

UserDAO 的 insert

```java
@Override
public boolean insert(String sql,String tableName) {
    // 插入,在验证(登陆)时使用
    System.out.println("对语句'"+sql+"'执行insert():");

    // 判断表格是否存在,不存在则建表,存在则直接插入
    if (isTableExist(tableName))
    {
        System.out.println(tableName+"表格存在");
    }else
    {
        System.out.println(tableName+"表格不存在,创建表格");
        // 写死了,回来再改
        createTable();
    }
    Connection connection = this.openDBConnection();
    try {
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString( parameterIndex: 1,userName);
        preparedStatement.setString( parameterIndex: 2,userPassword);
        int rowNum = preparedStatement.executeUpdate();
        System.out.println("        插入成功,插入行数为:"+rowNum);

        this.closeDBConnection(connection,preparedStatement, rs: null);
        return true;
    } catch (SQLException e) {
        System.out.println("        insert()方法出错:"+e);
        // e.printStackTrace();
    }
    return false;
}
```

判断表格是否存在的方法 isTableExist

```java
private boolean isTableExist(String tableName)
{
    // 判断表是否存在
    System.out.println("开始执行表格监测方法");
    boolean flag = false;
    Connection connection = this.openDBConnection();
    try {
        DatabaseMetaData metaData = connection.getMetaData();
        String type [] = {"TABLE"};
        ResultSet rs = metaData.getTables( catalog: null,  schemaPattern: null, tableName, type);
        flag = rs.next();

        this.closeDBConnection(connection, pstmt: null,rs);
    } catch (SQLException e) {
        System.out.println("    *表格检测方法失败:"+e);
        // e.printStackTrace();
    }

    System.out.println("表格监测方法执行结束:"+flag);
    return flag;
}
```

表格不存在则还需新建表格(创建的内容写死了)

```
private void createTable()
{
    System.out.println("开始创建表格");
    Connection connection = this.openDBConnection();
    try {
        Statement statement = connection.createStatement();
        String sql = "CREATE TABLE USER " +
                "(USERID VARCHAR(255) not NULL, " +
                " USERNAME VARCHAR(255), " +
                " USERPASS VARCHAR(255), " +
                " PRIMARY KEY ( USERID ))";
        statement.executeUpdate(sql);
        statement.close();
        this.closeDBConnection(connection, pstmt: null, rs: null);
    } catch (SQLException e) {
        System.out.println("创建表格失败"+e);
    }

}
```

4.结论

1.

首先直接进行登陆(密码是 555)

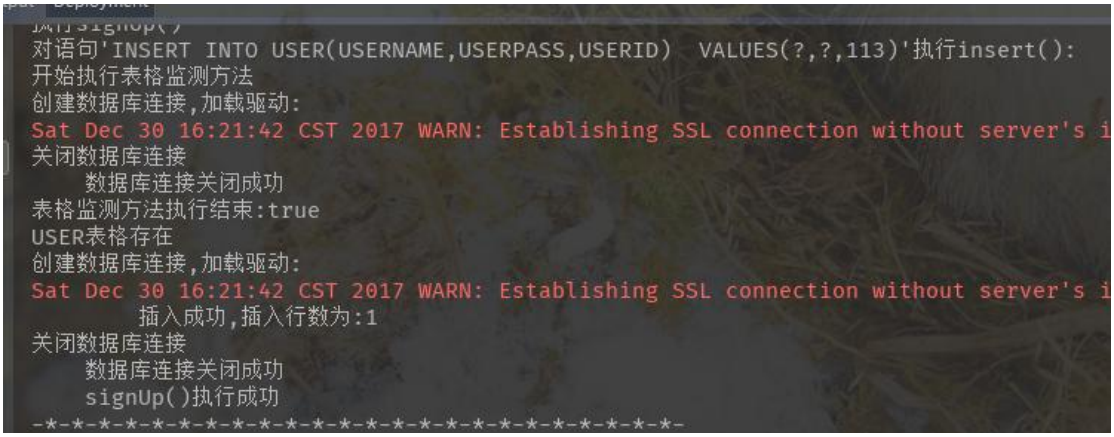
结果是登陆失败

可以看到在数据库中没有查到结果

2.

现在点击注册

注册完毕后直接到了登陆界面

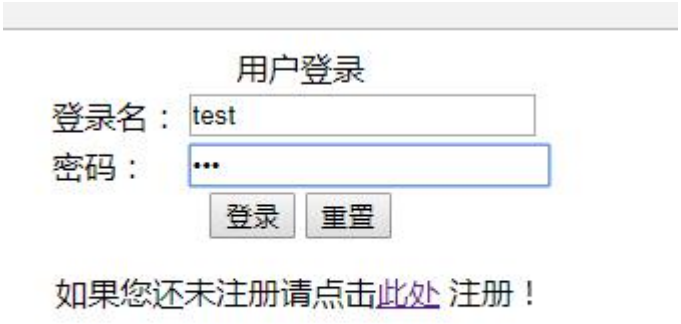


因为之前已经试验过,所以表格已经存在了

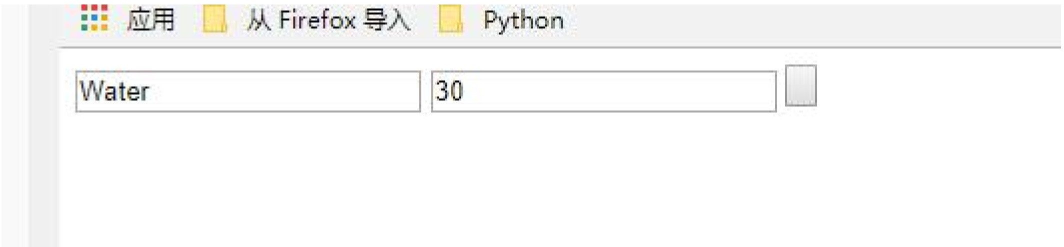
该用户信息已经被录入在了系统中

3.

现在再登陆



登陆成功



拦截器等操作都正常执行

5.参考文献

以上内容的理论知识点或技术点如果参考了网上或印刷制品，请在这里罗列出来