轻量级 J2EE 框架应用

E 5　A Simple Controller with DAO pattern

学号：SA18225433

姓名：杨帆

报告撰写时间：2018/12/01

1.主题概述

1、 DAO 层。

DAO 层全称为 Data Access Object（数据访问层），一般处于 Service 层（业务逻辑层）与数据库之间，主要功能是进行与数据库交互的数据操作，将对数据库的操作封装为增删改查的方法接口来供 Service 层调用完成相应的业务逻辑。

2、 JDBC。

JDBC 的概念为：JDBC（Java Data Base Connectivity,java 数据库连接）是一种用于执行 SQL 语句的 Java API，可以为多种关系数据库提供统一访问，它由一组用 Java 语言编写的类和接口组成。JDBC 提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序。其基本层次为应用程序、JDBC、各种数据库驱动、对应的数据库，我们需要使用厂商给出的数据库驱动连接数据库，使用 JDBC 定义的数据库 Connection 对象与数据库建立连接，使用 Statement 接口执行静态的 SQL 语句并返回它所生成结果的结果集 ResultSet，从结果集中获得 SQL 语句的执行结果。

3、 MYSQL 与 SQLITE

本次作业使用到了 MYSQL 数据库与 SQLITE 数据库，MySQL 是一个关系型数据库，是完善的独立的数据库服务器，MYSQL 使用标准的 SQL 数据库语言，实现了比较全面的数据库功能与特性，而且具有很高的效率，能处理千万条数据规模的数据，在平时学习开发中 mysql 数据是数据库学习的首选，而相比于 MYSQL 数据库，SQLITE 数据库是轻量级的小型数据库，功能简化，追求最大磁盘效率，整个数据库都存储在一个文件中，能够很方便的进行数据转储，但是其没有用户的划分，在性能优化上也略显不足。

2.假设

本次作业能够使用 IDEA 实现 Controller 具有 dao 层，且能够使用 dao 层访问数据库的能力，即使用 SimpleController 定义数据库 dao 层接口，使用 UseSC 实现接口，同时创建 bean 类，在 bean 类中直接使用 dao 层接口进行数据库的查询操作，来进行用户登录的用户名和密码验证的简单功能。

3.实现或证明

1. 实现成果

e5： https://github.com/saaaaaail/J2eee5

2. 基于 E4。在 SimpleController 工程中新建 package 名为 sc.ustc.dao，在该包中新建一个抽象类为 BaseDAO，该类中有以下 protected 属性及其 Setter 方法：driver（数据库驱动类），url（数据库访问路径），userName（数据库用户名），userPassword（数据库用户密码）；

新建 dao 层包，同时建立 BaseDAO 抽象类，具有 driver（数据库驱动类），url（数据库访问路径），userName（数据库用户名），userPassword（数据库用户密码）属性，

```
public abstract class BaseDAO {
    protected String driver;
    protected String url;
    protected String userName;
    protected String userPassword;
    protected Connection connection;
```

BaseDAO 实现以下方法：

openDBConnection(): Connection，负责打开数据库连接

加载数据库驱动程序，通过反射创建 driver 对象，然后使用 driverManager 获得数据连接对象，建立数据库连接，

```
public Connection openDBConnectionMYSQL() {
    try {
        Class.forName(driver);
        connection = DriverManager.getConnection(url,userName,userPassword);
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return connection;
}
```

closeDBConnection(): boolean，负责关闭数据库连接

判断数据库连接对象是否为空，若不为空，则关闭连接，

```
public boolean closeDBConnection(){
    try {
        if(connection!=null){
            connection.close();
        }
        System.out.println("close Connection");
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}
```

BaseDAO 定义以下抽象方法：

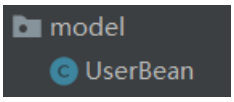
query(String sql): Object，负责执行 sql 语句，并返回结果对象
insert(String sql): boolean，负责执行 sql 语句，并返回执行结果
update(String sql): boolean，负责执行 sql 语句，并返回执行结果
delete(String sql): boolean， 负责执行 sql 语句，并返回执行结果

定义增删改查的抽象方法，

```
public abstract Object query(String sql);

public abstract boolean insert(String sql);

public abstract boolean update(String sql);

public abstract boolean dalete(String sql);
```

3. 在 UseSC 工程中新建 UserBean，UserBean 具有域 userId,userName,userPass。修改 LoginAction 源码，当调用 login 方法时，将请求分发至 UserBean；使用 UserBean 中的 signIn(): boolean 方法负责处理登录业务。

新建 bean 类，具有 userId,userName,userPass 三个属性，




```
public class UserBean {
    private String  userId;
    private String userName;
    private String userPass;
```

修改 LoginAction 的 handleLogin 方法，添加用户名和密码到传入参数上，然后新建该用户名和密码的 bean 类，调用 bean 类的 signIn 方法根据返回值判断是否验证成功，

```
public class LoginAction {
    private UserBean userBean;
    public String handleLogin(String name,String password) {
        userBean = new UserBean(name, password);
        System.out.println("执行handleLogin...");
        if (userBean.signIn(name,password)) {
            return "success";
        } else {
            return "failure";
        }
    }
}
```

4.在 UseSC 工程中新建 UserDAO 继承 SimpleController 的 BaseDAO，在 UserDAO 的构造方法中初始化父类域：driver（数据库驱动类），url（数据库访问路径），userName（数据库用户名），userPassword（数据库用户密码）。UserBean 中的 signIn(): boolean 方法使用域 userName 构造用户查询语句，调用 UserDAO 的 query (String sql): Object 查询该用户是否存在。如果存在，从数据库中取出 password 属性，构造一个新的 UserBean 对象，否则返回 NULL。

新建 UserDAO 类继承 SimpleController 中的 BaseDAO 抽象类，实现数据库属性字段在构造函数中的初始化，包括数据库驱动，url，用户名，密码等，同时打开数据库连接，

```
private UserDAO() {
    userPassword = "123456";
    userName = "root";
    driver = "com.mysql.jdbc.Driver";
    url = "jdbc:mysql://localhost:3306/sc?useSSL=false&serverTimezone=UTC";
    /*
    driver="org.sqlite.JDBC";
    url="jdbc:sqlite:G:\\sqliteDB\\identifier.sqlite";
    */
    connection = openDBConnectionMYSQL();
    //connection = openDBConnectionSQLITE();
}
```

然后定义 UserDAO 的 query 方法，构建条件查询的 SQL 语句，创建 statement 对象执行 SQL 语句访问数据库，获得 ResultSet，从 ResultSet 中获得查询密码，构建 UserBean 对象，并返回 UserBean 对象，如果没有查询结果则返回 null，

```
public Object query(String s) {
    String pass="";
    System.out.println(s);
    String FIELD = "password";
    String SQL_QUERY = "select "+FIELD+" from user where name = "+"'"+s+"'";
    System.out.println(SQL_QUERY);
    try {
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(SQL_QUERY);
        while (resultSet.next()){
            pass= resultSet.getString( columnLabel: "password");
            System.out.println(pass);
        }
        UserBean userBean = new UserBean();
        userBean.setUserPass(pass);
        return userBean;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return null;
}
```

5.UserBean 中的 signIn(): boolean 接收 UserDAO 的 query()方法返回的结果对象，并对结果对象判断是否为 NULL。如果结果为 NULL，signIn()返回 false；否则，对返回结果对象取 userPass 属性，并与当前对象中的 userPass 属性进行是否相同判断。如果相同，login()返回 true；否则，返回 false。

定义 UserBean 中的 signIn 方法，通过 DAO 类的 query 方法查询对应用户名的密码，与传入的密码进行比较，若相等返回 true 表示登录成功，若不等，或者没有返回 UserBean 对象，则返回 false 表示登录失败，

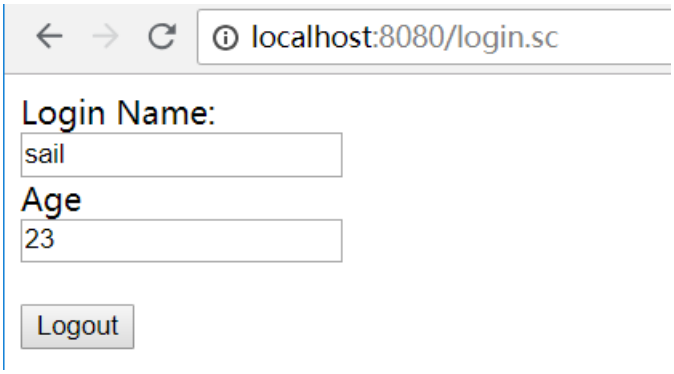
```
public boolean signIn(String name,String password){
    UserDAO userDAO=UserDAO.getInstance();
    UserBean userBean = (UserBean) userDAO.query(name);
    if (userBean!=null){
        if(userBean.getUserPass().equals(password)) {
            return true;
        }
    }
    return false;
}
```

6.将 SimpleController、UseSC 工程打包进行用户登录(需要实现数据库访问功能) 测试，如果有错，调试直到预期结果输出。

为了实现用户登录的直观显示，将 welcome.html 页面构建为具有登录和注册两个功能的两个表单，分别向 login.sc 和 register.sc 发起请求，

```
<div>欢迎使用SimpleController！</div>

<div>
    <form action="login.sc" method="post">
        <input type="text" value="sail" name="username"/><br>
        <input type="password" name="userpass"/><br>
        <input type="submit" value="登陆"/><br>
    </form>
    <br>
    <form action="register.sc" method="post">
        <input type="text" value="sail" name="username"/><br>
        <input type="password" name = "userpass"/><br>
        <input type="submit" value="注册"/><br>
    </form>
</div>
```

则请求结果如图所示，点击登录后完成跳转



同时控制台打印用户名、查询语句以及查询到的密码，

```
sail
select password from user where name = 'sail'
123456
```

7.将 UseSC 工程中使用的 DBMS 更改为另一个关系型 DBMS，仅修改 UserDAO 代码，其他代码保持不变，重新将工程打包进行用户登录测试。如将现有工程中的 mysql

更改为 sqlite 或 postgresql。

将 mysql 修改为 sqlite，修改 UserDAO 代码，初始化数据库驱动与数据库的 url 地址，并且获得 sqlite 数据库的连接对象，

```
private UserDAO() {
    /*
    userPassword = "123456";
    userName = "root";
    driver = "com.mysql.jdbc.Driver";
    url = "jdbc:mysql://localhost:3306/sc?useSSL=false&serverTimezone=UTC";
    */
    driver="org.sqlite.JDBC";
    url="jdbc:sqlite:G:\\sqliteDB\\identifier.sqlite";

    //connection = openDBConnectionMYSQL();
    connection = openDBConnectionSQLITE();
}
```

4.结论

对主题的总结，结果评论，发现的问题，或你的建议和看法。
本次作业学习了数据操作的层次 dao，了解了 dao 层存在的作用以及为什么要添加 dao 层，了解了业务逻辑层通过 dao 层访问数据库的优势，然后学习了构建自己的 dao 层，学习了如何使用 JDBC 库连接数据库，并实现与数据库的数据交互，并通过与数据库的简单交互，实现登录验证的功能，最后改变数据库驱动，使用 sqlite 替换 mysql 成功连接到 sqlite。

第一个问题，关于 java.lang.ClassNotFoundException: com.mysql.jdbc.Driver；
出现了两次该问题，第一次出现的原因是因为没有导入 Mysql-Connect 驱动的 jar 包，导致无法解析该数据库驱动地址，导入 jar 包后运行依然报错，查阅资料后发现 jar 包版本较低，使用 IDEA 导入最新版 jar 包，解决该问题。

第二个问题，查询语句报 Unknown column 'sail' in 'where clause'错误；
当构建 SQL 语句查询密码时，无法匹配用户名，由于用户名为字符串所以在 sql 语句匹配时需要添加上单引号（s 表示需要匹配的用户名，需要额外添加单引号），

```
String SQL_QUERY = "select "+FIELD+" from user where name = "+"'"+s+"'";
```

。

5.参考文献

1.参阅 github 学长代码编写风格
2.使用 JDBC 连接数据库（一）
3.关于 java.lang.ClassNotFoundException: com.mysql.jdbc.Driver 的解决方案
4.IntelliJ IDEA 15 设置默认浏览器
5.连接数据库常见问题，Unknown column '张三' in 'where clause'
6.通过 JDBC 进行简单的增删改查（以 MySQL 为例）
7.idea_在 idea 中使用 sqlite 数据库
8.dao 层和 service 层的区别
9.学习数据库 MySQL，与 SQLite 数据库作对比