

# 轻量级 J2EE 框架应用

## E 5 A Simple Controller with DAO pattern

学号: SA16225221

姓名: 欧勇

报告撰写时间: 2016/12/25

# 1. 主题概述

简要介绍主题的核心内容

1. 基于 E1。新建一个类为 UserDAO，该类中主要有以下方法：  
openDBConnection(): Connection, 负责打开 MySQL 数据库连接  
closeDBConnection(): boolean, 负责关闭 MySQL 数据库连接  
queryUser(String userName): UserBean, 负责根据参数查询对象表记录  
insertUser(UserBean u): boolean, 负责根据参数增加对象表记录  
updateUser(UserBean u): boolean, 负责根据参数修改对象表记录  
deleteUser(UserBean u): boolean, 负责根据参数删除对象表记录
2. 当 E1 中的控制器，接收到登录请求时，将请求分发至 UserBean；UserBean 中的 login(): boolean 方法负责处理登录业务。
3. UserBean 中的 login(): boolean 方法使用 UserDAO 的 queryUserBean(String userName): UserBean 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。
4. UserBean 中的 login(): boolean 接收 UserDAO 的 queryUserBean 方法返回结果对象，并对结果对象判断是否为 NULL。如果结果为 NULL，login()返回 false；否则，对返回结果对象取 password 属性，并与当前对象中的 password 属性进行是否相同判断。如果相同，login()返回 true；否则，返回 false。
5. 将工程打包进行用户登录测试。
6. 将工程中使用的 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。

## 2. 假设

主题内容所参照的假设条件，或假定的某故事场景，如调试工具或软硬件环境

开发环境：

**Win10**

**Eclipse kepler**

**JDK 1.8**

**Tomcat 7.0**

### 3. 实现或证明

对主题内容进行实验实现，或例举证明，需描述实现过程及数据。如对 MVC 中 Controller 功能的实现及例证（图示、数据、代码等）

注：本次 E5 作业报告是在 E1 作业报告的基础上完成的，同时由于数据库操作不是作业重点，所以本次作业中省略了建立 MySQL、SQLite 两种数据库、表和填入数据的说明。

流程：

为了方便查看，采用 get 方式提交，可以通过浏览器 url 看到输入的用户名和密码（因为若采用 post 方式则无法通过 url 看到用户名和密码，所以采用 get 方式提交）  
若登录成功则跳转 login\_success.jsp 页面，页面显示 Login Success 的字符串  
若登录失败则跳转 login\_fail.jsp 页面，页面显示 Login Fail 的字符串

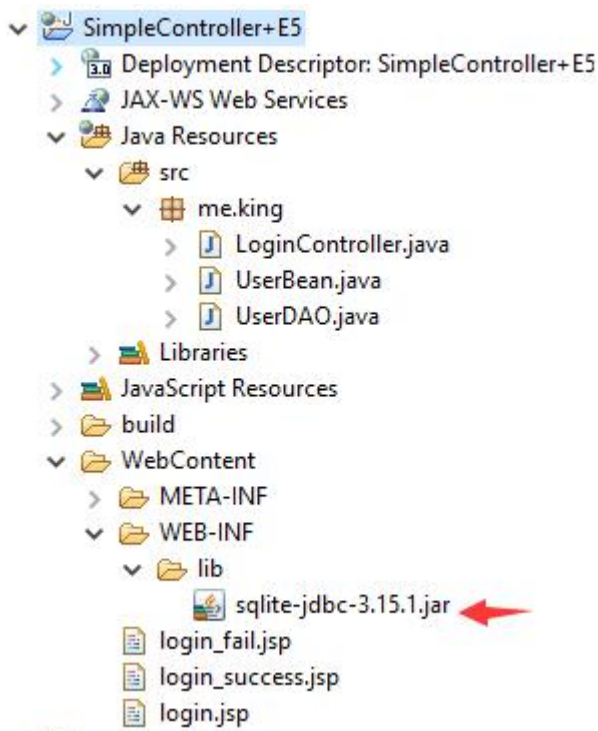


图 1：项目目录结构

图 1：项目目录结构，可以看出项目名称为 SimpleController，包名为 me.king，分别有两个类，一个是 LoginController 作为控制层，一个是 UserBean 作为模型层，UserDAO 作为数据库连接类，由若干与用户有关的数据库操作方法，

在 WEB\_INF/lib 下添加 sqlite-jdbc 的库，接下来将采用 sqlite 替换 mysql 测试，最后还有 3 个 jsp 页面作为视图层（同 E1），分别是 login\_fail.jsp，login\_success.jsp 和 login.jsp

```
LoginController.java login.jsp UserBean.java UserDAO.java
1 package me.king;
2
3 import java.io.IOException;
12
14 * Servlet implementation class LoginController
16 @WebServlet("/LoginController")
17 public class LoginController extends HttpServlet {
18     private static final long serialVersionUID = 1L;
19
21 * @see HttpServlet#HttpServlet()
23 public LoginController() {
24     super();
25     // TODO Auto-generated constructor stub
26 }
27
29 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
32 protected void doGet(HttpServletRequest request,
33     HttpServletResponse response) throws ServletException, IOException {
34     // TODO Auto-generated method stub
35     ServletContext sc = getServletContext(); //获取servlet上下文
36     RequestDispatcher rd = null; //新建一个requestDispatcher对象用于内部重定向
37     //传入request中的用户名和密码新建一个UserBean对象
38     UserBean ub = new UserBean(request.getParameter("name"), request.getParameter("pwd"));
39     try { //若登录成功则设置重定向到login_success.jsp, 否则为login_fail.jsp
40         if (ub.login()) rd = sc.getRequestDispatcher("/login_success.jsp");
41         else rd = sc.getRequestDispatcher("/login_fail.jsp");
42
43         rd.forward(request, response);
44     } catch (Exception e) {
45         // TODO Auto-generated catch block
46         e.printStackTrace();
47     }
48 }
49
52 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
55 protected void doPost(HttpServletRequest request,
56     HttpServletResponse response) throws ServletException, IOException {
57     // TODO Auto-generated method stub
58     doGet(request, response);
59 }
60 }
```

图 2.1: LoginController 代码截图

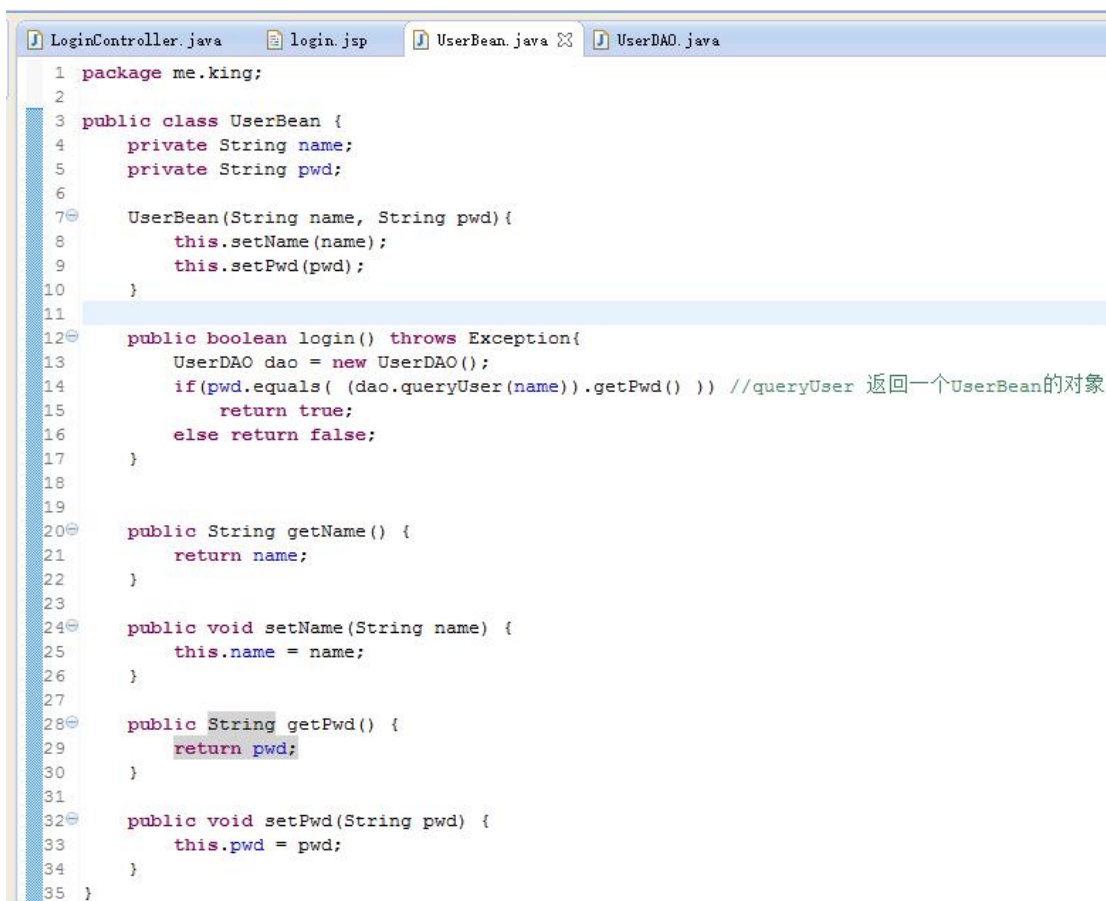
图 2.1: LoginController 代码截图,

使用注解的方式告知容器, LoginController servlet 映射的 url 为“/LoginController”。

代码为: `@WebServlet("/LoginController")`

在 doGet 方法中, 新建一个 UserBean 对象 ub, 传入 request 对象中的 name 和 pwd。调用 ub 的 login 方法, 若 login 验证用户名和密码成功返回 true 则设置跳转至 login\_success.jsp 页面, 否则跳转至 login\_fail.jsp 页面。最后采用 RequestDispatcher.forward 将 request 和 response 两个对象返回至指定的页面。

其他采用默认, 若前端页面采用 post 方式提交, 则在 doPost 方法中也需要进行转发处理, 也可以直接调用 doGet()方法。



```
1 package me.king;
2
3 public class UserBean {
4     private String name;
5     private String pwd;
6
7     UserBean(String name, String pwd){
8         this.setName(name);
9         this.setPwd(pwd);
10    }
11
12    public boolean login() throws Exception{
13        UserDAO dao = new UserDAO();
14        if(pwd.equals( (dao.queryUser(name)).getPwd() )) //queryUser 返回一个UserBean的对象
15            return true;
16        else return false;
17    }
18
19
20    public String getName() {
21        return name;
22    }
23
24    public void setName(String name) {
25        this.name = name;
26    }
27
28    public String getPwd() {
29        return pwd;
30    }
31
32    public void setPwd(String pwd) {
33        this.pwd = pwd;
34    }
35 }
```

图 2.2: UserBean 代码截图

图 2.2: UserBean 代码截图,

定义一个简单的无参 login 方法, 函数体内需要先 new 一个 UserDAO 对象 dao, 调用 dao 对象的 queryUser 方法, 传入 name 获取对应的 UserBean 对象, 然后对比数据库返回的 UserBean 对象的密码与当前 UserBean 对象的密码, 若一致则表示正确, 返回 true, 否则返回 false。

```

1  UserDAO.java
2  import java.sql.DriverManager;
3  import java.sql.ResultSet;
4  import java.sql.SQLException;
5  import java.sql.Statement;
6  import java.sql.Connection;
7  import com.mysql.jdbc.PreparedStatement;
8  //import com.mysql.jdbc.Connection;
9
10
11 public class UserDAO {
12     private Connection con = null;
13     private Statement stat = null;
14     private ResultSet rs = null;
15     private boolean rt = new Boolean(false); //用于保存临时的返回结果
16     private static boolean useMySQL = new Boolean(false); //true使用MySQL数据库, false使用SQLite数据库
17     static // 确定数据库 使用静态方法, 只执行一次
18     {
19         try {
20             if(useMySQL) Class.forName("com.mysql.jdbc.Driver"); //MySQL数据库类
21             else Class.forName("org.sqlite.JDBC"); //SQLite数据库类
22         } catch (ClassNotFoundException e) {
23             // TODO Auto-generated catch block
24             e.printStackTrace();
25         }
26         // 负责打开MySQL 数据库连接
27     public Connection openDBConnection() throws Exception { // 获取数据库连接
28         if(useMySQL) //使用MySQL数据库
29             return (Connection)DriverManager.getConnection("jdbc:mysql://localhost:3306/monitor", "root", "hello");
30         else //使用SQLite数据库
31             return (Connection)DriverManager.getConnection("jdbc:sqlite:E:test.db");
32     }
33     // 负责关闭MySQL 数据库连接
34     public boolean closeDBConnection() {
35         try {
36             if (stat != null) stat.close();
37             if (rs != null) rs.close();
38             if (con != null) con.close();
39             return true;
40         } catch (SQLException e) {
41             // TODO Auto-generated catch block
42             e.printStackTrace();
43             return false;
44         }
45     }
46
47     // 负责根据参数查询对象记录
48     public UserBean queryUser(String userName) throws Exception {}
49     // 负责根据参数增加对象记录
50     public boolean insertUser(UserBean u) throws Exception {}
51     // 负责根据参数修改对象记录
52     public boolean updateUser(UserBean u) throws Exception {}
53     // 负责根据参数删除对象记录
54     public boolean deleteUser(UserBean u) throws Exception {}

```

图 3.1: UserDAO 类概览截图

图 3.1: UserDAO 类概览截图

先在类中定义了一些静态变量，因为所有的方法都需要打开数据库连接，执行 sql 语句，同时返回结果，rs 和 rt 为不同的执行 sql 语句返回的结果类型。

同时在 17-25 行在确认数据库的时候也采用静态方法，MySQL 和 SQLite 分别进行测试，开关为 useMySQL 变量，按照作业要求将 UserDAO 类中的方法都实现了，但是最后仅仅只给出了登录的测试的截图。

注意 1，SQLite 的连接必须使用绝对位置，本次作业中 test.db 的位置为 E 盘根目录下。若写错位置则报图 3.1.1 的错误，找不到数据库或表。

```

org.sqlite.SQLiteException: [SQLITE_ERROR] SQL error or missing database (no such table: monitor_user)
    at org.sqlite.core.DB.newSQLException(DB.java:909)
    at org.sqlite.core.DB.newSQLException(DB.java:921)

```

图 3.1.1: SQLite 数据库位置错误报错截图

注意 2，Connection 的类型，若为了兼容 MySQL 和 SQLite 的类型，需要导入 `import java.sql.Connection;` 而不是 `com.mysql.jdbc.Connection`，若导入错误，则会在 SQLite 测试的报如图 3.1.2 的错误，表示 Connection 类型不兼容，无法将 MySQL 的 Connection 转为 SQLite 的 Connection。

```
Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (Dec 24, 2016, 10:53:41 AM)
java.lang.ClassCastException: org.sqlite.SQLiteConnection cannot be cast to com.mysql.jdbc.Connection
    at me.king.UserDAO.openDBConnection(UserDAO.java:28)
    at me.king.UserDAO.queryUser(UserDAO.java:51)
    at me.king.UserBean.login(UserBean.java:14)
```

图 3.1.2: 导入 Connection 的类型报错截图

注意 3, closeDBConnection 方法中关闭顺序, 在 MySQL 中 connection 可以正在 statement 和 resultSet 变量之前关闭, 但是在 SQLite 中不行。否则会报如图 3.1.3 的错误, 找不到数据库, 提示连接已经关闭。

```
37 public boolean closeDBConnection() {
38     try {
39         if (con != null) con.close();
40         if (stat != null) stat.close();
41         if (rs != null) rs.close();
42         return true;
43     } catch (SQLException e) {
44         // TODO Auto-generated catch block
45         e.printStackTrace();
46         return false;
47     }
48 }
```

```
org.sqlite.SQLiteException: [SQLITE_ERROR] SQL error or missing database (Connection is closed)
    at org.sqlite.core.DB.newSQLException(DB.java:909)
    at org.sqlite.core.CoreStatement.internalClose(CoreStatement.java:115)
    at org.sqlite.jdbc3.JDBC3Statement.close(JDBC3Statement.java:35)
    at org.sqlite.jdbc4.JDBC4Statement.close(JDBC4Statement.java:27)
    at me.king.UserDAO.closeDBConnection(UserDAO.java:40)
    at me.king.UserDAO.queryUser(UserDAO.java:60)
    at me.king.UserBean.login(UserBean.java:14)
    at me.king.LoginController.doGet(LoginController.java:40)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:621)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:728)
```

图 3.1.3: closeDBConnection 方法中关闭顺序错误报错截图



```

47 // 负责根据参数查询对象表记录
48 public UserBean queryUser(String userName) throws Exception {
49     con = openDBConnection();
50     String sql = "select * from monitor_user where user_Name='" + userName + "'";
51     UserBean ub = null;
52     stat = con.createStatement();
53     rs = stat.executeQuery(sql);
54     if (rs.next())
55         ub = new UserBean(rs.getString(1), rs.getString(2));
56
57     closeDBConnection();
58     return ub;
59 }
60 // 负责根据参数增加对象表记录
61 public boolean insertUser(UserBean u) throws Exception {
62     con = openDBConnection();
63     String sql = "insert into monitor_user(user_Name,user_Password) values(?,?)";
64     PreparedStatement prepStmt = (PreparedStatement) con.prepareStatement(sql);
65     prepStmt.setString(1, u.getName()); // 用name参数代替第一个?
66     prepStmt.setString(2, u.getPwd()); // 用pwd参数代替第二个?
67
68     if (prepStmt.executeUpdate() > 0) rt = true;
69     else rt = false;
70     closeDBConnection();
71
72     return rt;
73 }
74 // 负责根据参数修改对象表记录
75 public boolean updateUser(UserBean u) throws Exception {
76     con = openDBConnection();
77     String sql = "UPDATE monitor_user set user_Name=" + u.getName()
78         + ", user_Password=" + u.getPwd() + " where user_Name="
79         + u.getName();
80     stat = con.createStatement();
81
82     if (stat.executeUpdate(sql) > 0)
83         rt = true;
84     else rt = false;
85     closeDBConnection();
86     return rt;
87 }
88 // 负责根据参数删除对象表记录
89 public boolean deleteUser(UserBean u) throws Exception {
90     con = openDBConnection();
91     String sql = "delete from user where user_Name=" + u.getName();
92     stat = con.createStatement();
93     if (stat.executeUpdate(sql) > 0)
94         rt = true;
95     else rt = false;
96     return rt;
97 }

```

图 3.2: UserDao 类 query、insert、update 和 delete 等操作 User 方法截图

图 3.2: UserDao 类 query、insert、update 和 delete 等操作 User 方法截图  
其中 queryUserBean 方法查询该用户是否存在，如果存在，从数据库中取出 pwd 属性，构造一个新的 UserBean 对象，否则返回 NULL。

```

X .project  LoginController.java  UserBean.java  login_success.jsp  login_fail.jsp  login.jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login</title>
</head>
<body>

<form action="LoginController">
  <label>NAME:</label><input type="text" name="name"><br>
  <label>PASSWORD:</label><input type="password" name="pwd"><br>
  <input type="submit" value="LOGIN">
</form>

</body>
</html>
    
```

图 4.1: login.jsp 代码截图

图 4.1: login.jsp 代码截图，编写一个 form 表单，action 设置为 LoginController（就是在 LoginController 类中使用 @webservert 注解映射的 url），设置用户名和密码的 name 属性为 name 和 pwd。

```

X .project  LoginController.java  UserBean.java  login_success.jsp  login_fail.jsp  login.jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login Success</title>
</head>
<body>
Login Success
</body>
</html>
    
```

图 4.2: login\_success.jsp 代码截图

图 4.2: login\_success.jsp 代码截图，简单的在 body 中写入 Login Success。

```

X .project  LoginController.java  UserBean.java  login_success.jsp  login_fail.jsp  login.jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login Fail</title>
</head>
<body>
Login Fail
</body>
</html>
    
```

图 4.3: login\_fail.jsp 代码截图

图 4.3: login\_fail.jsp 代码截图，简单的在 body 中写入 Login Fail。

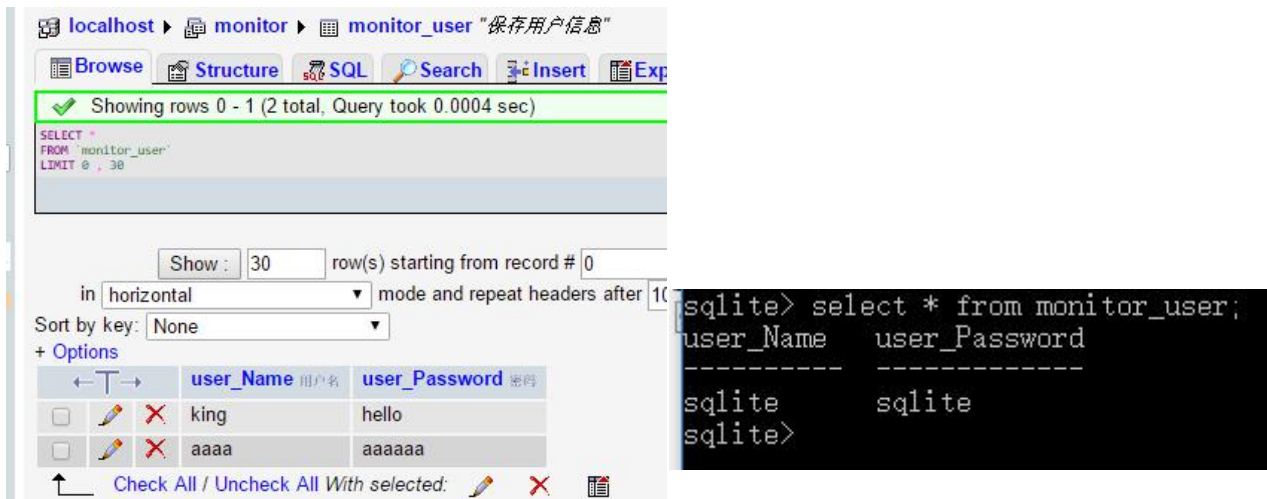


图 5. MySQL 和 SQLite 数据库的 monitor\_user 表内数据展示

图 5. 用户表 monitor\_user， user\_Name 表示用户名， user\_Password 表示密码，MySQL 数据库暂时只有 king 和 aaaa 两位用户，SQLite 数据库只有 sqlite 一个用户，密码和用户名一样，为了测试方便，密码采用明文存储。

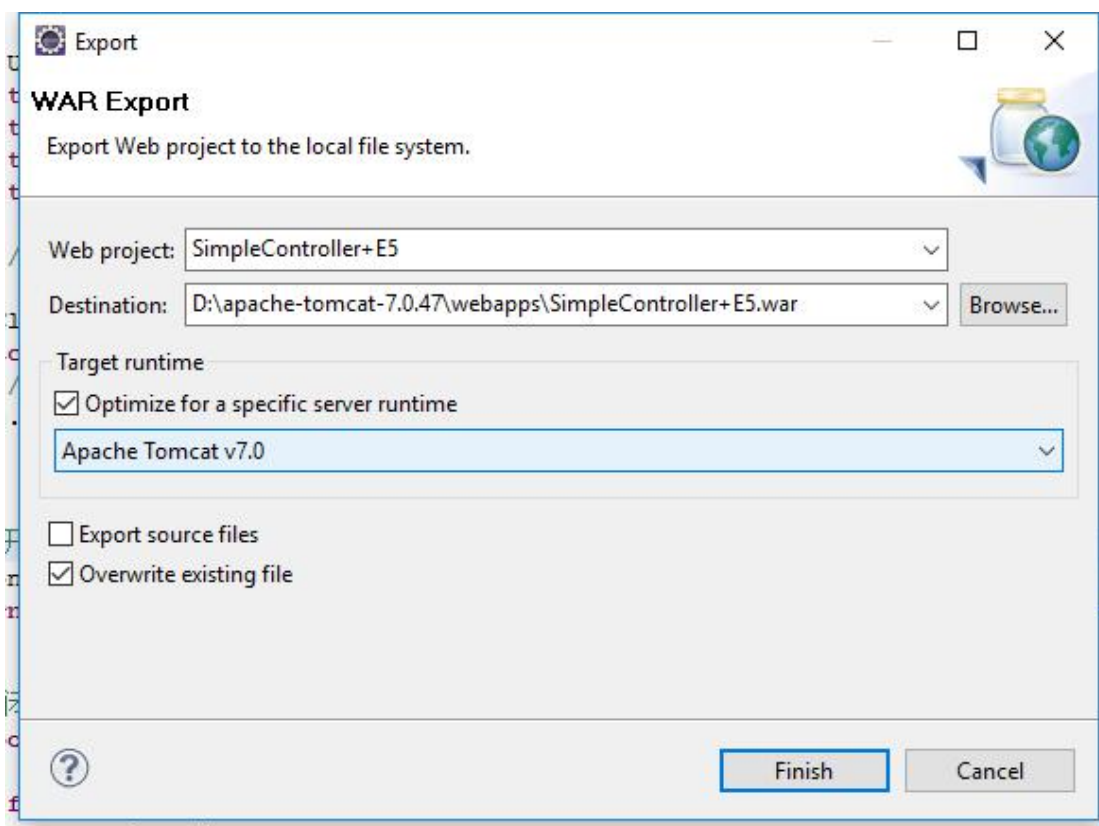


图 6.1: 导出为 War 包操作截图

图 6.1: 导出为 War 包操作截图,

在项目目录中右键选择 Export --> WAR file, 然后选择导出到 Tomcat 目录下的 webapps 中即可。

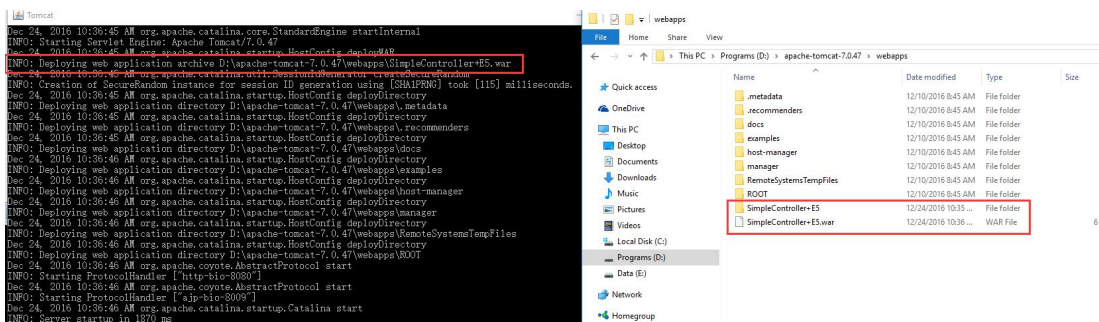


图 6.2: 部署在 Tomcat 中后使用 Chrome 浏览器测试截图

图 6.2: 部署在 Tomcat 中后使用 Chrome 浏览器测试截图,

在 Tomcat 目录下 bin 中选择 startup.bat 启动 Tomcat 服务, Tomcat 会自动将 war 包解压并将应用部署到同名文件夹下, 然后在浏览器地址栏输入 SimpleController 应用的登录页面地址:

<http://localhost:8080/SimpleController+E5/login.jsp>

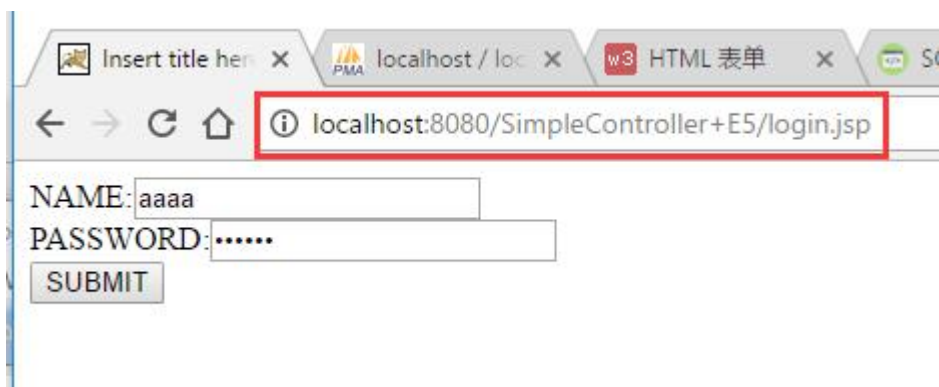


图 7.1: 使用 Chrome 浏览器测试截图

图 7.1: 使用 Chrome 浏览器测试截图，可以看到 url 为：<http://localhost:8080/SimpleController+E5/login.jsp>，可以在表单中分别填入 NAME 和 PASSWORD，点击 LOGIN 按钮提交至后台服务器。

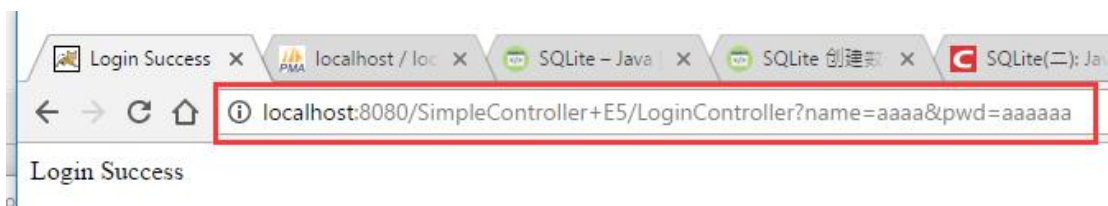


图 7.2: 使用 MySQL 数据库登录成功截图



图 7.3: 使用 MySQL 数据库登录失败截图

当数据库采用 MySQL 的时候，图 7.2: 登录成功截图，图 7.3: 登录失败截图可以通过 url 看到，当输入的用户名 name 为 aaaa，密码 pwd 为 aaaaaa 时验证通过时显示 Login Success  
当输入的用户名 name 为 aaaa，密码 pwd 为 xxxx 时，验证失败，显示 Login Fail（虽然跳转至 login\_fail.jsp 页面，但此时 url 没有改变为 login\_fail.jsp）

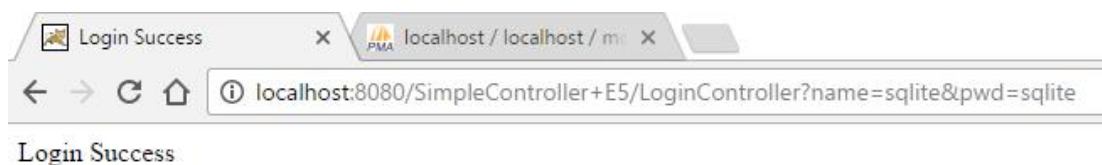


图 7.4: 使用 SQLite 数据库登录成功截图

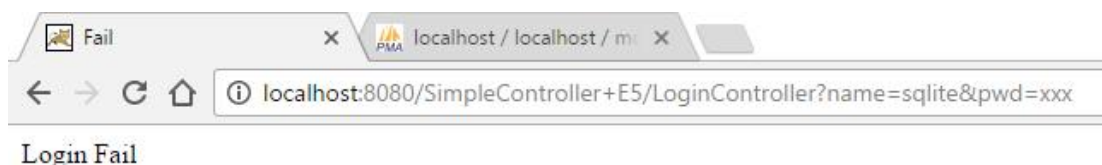


图 7.5: 使用 SQLite 数据库登录失败截图

当数据库采用 SQLite 的时候，图 7.4: 登录成功截图，图 7.5: 登录失败截图可以通过 url 看到，当输入的用户名 name 为 sqlite，密码 pwd 为 sqlite 时验证通过时显示 Login Success  
当输入的用户名 name 为 sqlite，密码 pwd 为 xxx 时，验证失败，显示 Login Fail

浏览器地址栏 URL 没有改变的原因是：

由于在 UserBean 中采用的是 `RequestDispatcher.forward()` 方式，这表示在服务器端运行。采用采用请求转发，request 对象始终存在，不会重新创建，前后页面共享同一个 request 重定向后浏览器地址栏 URL 不变。

若使用 `Response.sendRedirect()` 方式，则表示在用户的浏览器端工作。重新定向，前后页面不共享一个 request。重定向后在浏览器地址栏上会出现重定向页面的 URL。

## 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。

实验中遇到的一些错误已经在上文提及了。

经过本次作业，发现 SQLite 是一个非常小巧而方便的数据库，而且还跨平台，个人觉得以后可以多多使用它作为测试或实验用的数据库。同时，SQLite 也有可视化工具，但是工具使用的相关资料和教程比较少。

虽然说从理论上使用 DAO 的数据库连接模式后，更改数据库只需要加载相应的 jdbc 的库后，在 DAO 实现中加载不同的 Driver，然后通过 DriverManager 的 getConnection 方法获取连接即可，但是结果测试表面其实不同数据库在操作还是由细微差别的，比如 closeDBConnection 中关闭对应连接时的顺序。

## 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

[1] J2EE DAO: <http://www.oracle.com/technetwork/java/dataaccessobject-138824.html>

[2] MySQL: <http://www.mysql.com>

[2] SQLite: <http://www.runoob.com/sqlite/sqlite-commands.html>