浙江工业大学

JavaEE 技术实验报告

**实验名称：Servlet 与 JSP 技术第一个用户登录模块**

**学 院： 计算机科学与技术学院**

**班 级： 软工1801**

**姓 名： 沈丹**

**学 号： 201806062416**

**组 号： 无**

**时 间： 2020/10/10**

一、实验主要步骤

**一、基础实验——Servlet 与 JSP 基础开发**

（一）实验目的

1、掌握 HttpServlet 的概念，相关 API 以及开发步骤；

2、掌握 JSP 技术的基本语法；

3、掌握 JSP 各隐含变量的使用方法；

4、掌握使用 Servlet 和 JSP 集成开发简单用户登录功能。

（二）基本知识与原理

1、Servlet 是用于实现 Web 应用程序设计的 Java 技术解决方案，旨在扩展 Web服务器的能力，它是由 Servlet 容器（例如：Tomcat）创建并管理。

2、JSP（Java Server Pages）页面是包含 Java 代码和 HTML 标签的 Web 页面。 它由 JSP 标签和 HTML 标签混合而成的 Web 页面，主要用于进行用户交互。

（三）实验内容及步骤

1、下载并解压安装 Eclipse 的 Java EE 集成开发环境，如图 1-1 所示：



图 1-1 下载并解压安装 Eclipse 的 Java EE 集成开发环境

1. 在 Eclipse 中新建动态 Web 工程(Dynamic Web Project ) javaweb-prj1，如图 1-2 所示：

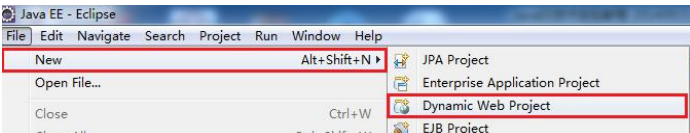


图 1-2 创建动态 Web 工程

3、为工程 javaweb-prj1 添加 Servlet 开发的外部库文件 servlet-api.jar，操作步骤如下：

（1）右 键 单 击 工 程 javaweb-prj1 图 标 ， 在 弹 出 的 菜 单 中 选 择 **[Build Path]**->**[Configure Build Path]**，如图 1-3 所示；

（2）在弹出的窗口中选择**[Java Build Path]**的**[Libraries]**选项卡，点击按钮**[Add**

**External JARS]**，接着在弹出的窗口中切换到 **Tomcat 安装目录（例如：**

**C:\Program Files (x86)\Apache Software Foundation\Tomcat 6.0）**下的 **lib** 目

录，然后选中并打开加载 servlet-api.jar 库文件，最后点击按钮**[OK]**，如图

1-4 所示；

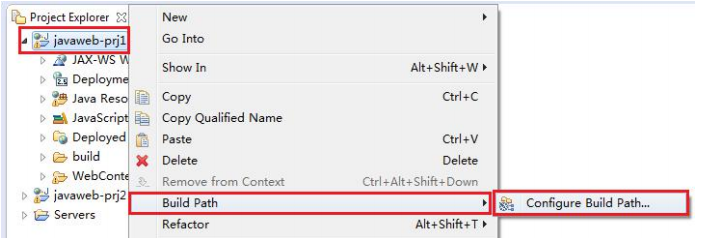


图 1-3 为工程添加外部库文件的菜单操作

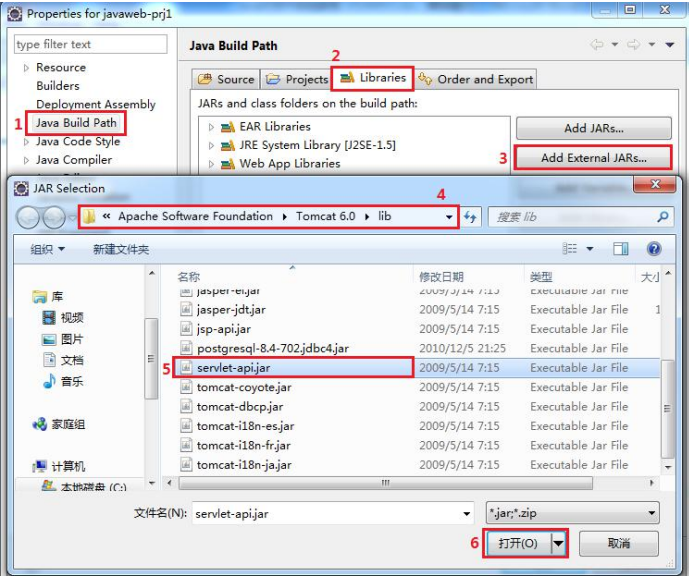


图 1-4 选择并添加外部库文件 Servlet-api.jar

1. 在 javaweb-prj1 中，右键单击目录**[WebContent]**，新建用户登录页面 login.jsp， 如图 1-5 所示，具体代码如下所示（注意表单 form 属性 **action** 的值，用户名与密码输入框中属性 **name** 的值）；

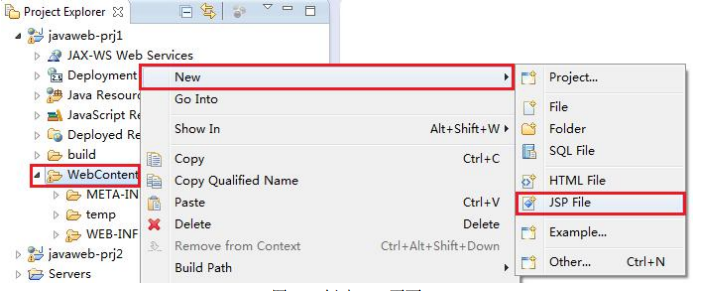


图 1-5 创建 JSP 页面

<%@ page language="java" contentType="text/html; charset=GBK"

pageEncoding="GBK"%>

<!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=GBK">

<title>用户登录页面</title>

</head>

<body>

<form **action="login"** method="post">

请输入用户名：<input **name="username"** type="text"><br>

请输入密码：<input **name="password"** type="password">

<input type="submit" value="登录">

</form>

</body>

</html>

5、在 javaweb-prj1 中，右键单击**[Java Resources]**->**[src]**目录，新建一个名称为

cn.edu.zjut 的包，然后在该包下创建一个 Servlet：LoginController.java（如图

1-6 所示），实现如下功能：用于接收 login.jsp 页面提交的用户名和密码，如

果用户名和密码均为 zjut，则输出“登录成功，欢迎您！”，否则输出“用

户名或密码错误！”，具体代码如下所示：

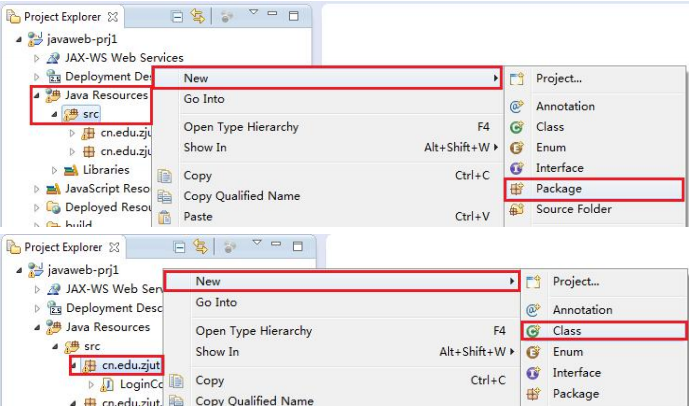


图 1-6 创建 Package 包及 Java 类

package cn.edu.zjut;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;public class LoginController **extends HttpServlet** {

protected void doPost(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

PrintWriter out=response.getWriter();

**String username = request.getParameter("username");**

**String password = request.getParameter("password");**

if("zjut".equals(username) && "zjut".equals(password)){

out.println("登录成功，欢迎您！");

}else{

out.println("用户名或密码错误！");

}

}

}

6、在 javaweb-prj1 的**[WebContent]->[WEB-INF]**目录下新建 web.xml 文件，为

LoginController 配置 url-pattern 映射，具体代码如下所示：

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"

id="WebApp\_ID" version="2.5">

<display-name>javaweb-prj1</display-name>

<servlet>

<servlet-name>**LoginController**</servlet-name>

<display-name>LoginController</display-name>

<description></description>

<servlet-class>**cn.edu.zjut.LoginController**</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>**LoginController**</servlet-name>

<url-pattern>**/login**</url-pattern>

</servlet-mapping>

</web-app>

7、将 javaweb-prj1 部署在 Tomcat 服务器上，具体操作步骤如下：

（1）下载安装好 Tomcat（本例为版本 6.0，实际应用中可选择 6.0 以上版本）；

（2）打开 Eclipse，选择菜单**[Window]->[Preferences]，**如图 1-7 所示**；**

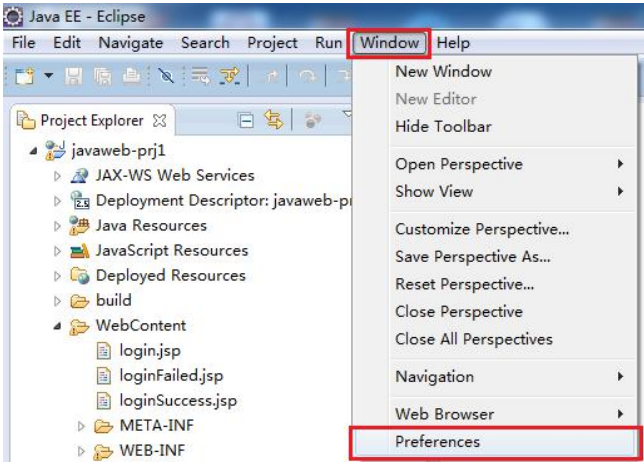


图 1-7 打开 Eclipse 的首选项

（3）在弹出的窗口左侧栏中选择[**Server**]->[**Runtime Environments**]，然后点击右

侧栏的按钮[**add**]，如图 1-8 所示。

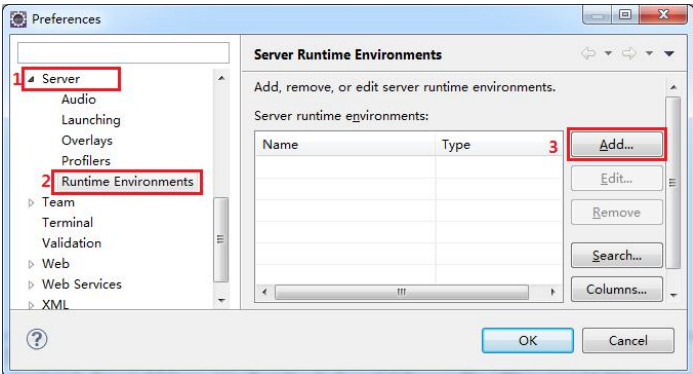


图 1-8 添加新的运行时环境

（4）在弹出的服务器窗口中选择[**Apache Tomcat v6.0**]，并点击按钮[**next**]，如图

1-9 所示；此时进入 Tomcat 的服务器配置窗口，点击[**Browse**]按钮，接着

在弹出的目录选择窗口中选择 **Tomcat 安装目录（例如：C:\Program Files**

**(x86)\Apache Software Foundation\Tomcat 6.0）**，点击按钮[**确定**]完成

Tomcat 服务器的选择，然后依次点击按钮[**Finish**]和按钮**[OK**]来保存服务器配置方案，如图 1-10 所示。

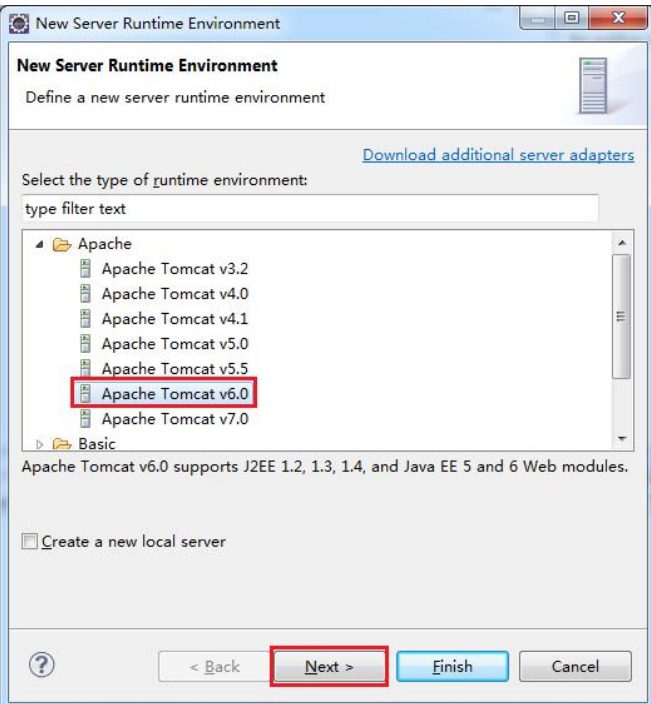


图 1-9 选定 Tomcat 服务器版本

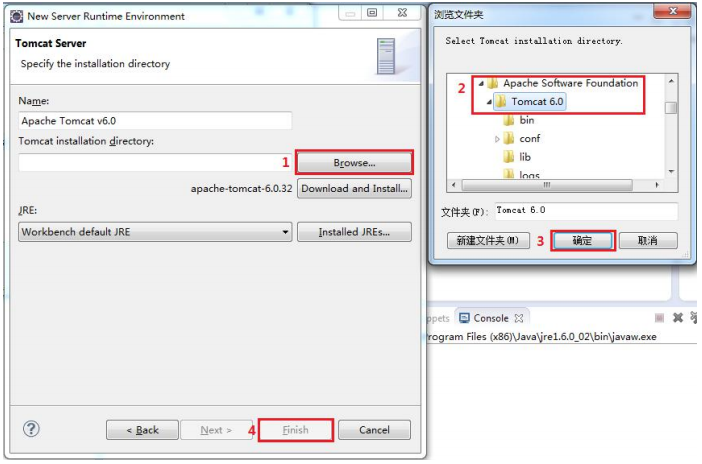


图1-10 选择实际已安装的 Tomcat 服务器

（5）右键单击 javaweb-prj1 工程，在弹出的菜单中依次选择[**Run As**]->[**Run on**

**Server**]，如所示；在弹出的窗口中选择刚才配置的[**Tomcat v6.0 Server**]，并

点击按钮[**Finish**]，此时服务器会自动启动并加载运行 javaweb-prj1 工程，

同时服务器启动的相关信息会出现在 Console 控制栏，如图 1-11 所示。

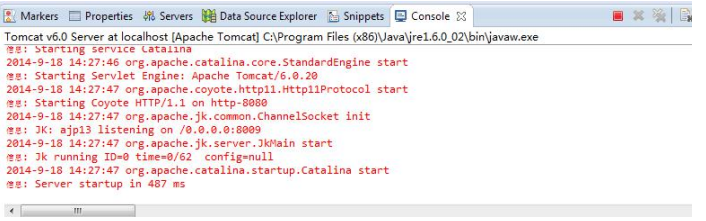


图 1-11 Console 控制栏中显示的 Tomcat 服务器启动信息

（6）打开任一浏览器，输入网址：**http://localhost:8080/javaweb-prj1/login.jsp**，

即可开始访问 **login.jsp** 页面，进行相关运行调试。

8、运行 login.jsp 页面，输入用户名和密码并记录运行结果；

9、修改 login.jsp 页面，使用表格对表单域进行对齐排列，运行并观察结果；

10、修改 login.jsp 页面，使用 JavaScript 对用户名表单域 username 和密码表单域password 进行校验(校验规则：不能为空且不能超过 6 位)，运行并观察结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）修改后的关键代码，及相应的运行结果或报错信息；

（3）实验收获及总结。

2、上交程序源代码，主要代码中应加注详细的注释。

**二、提高实验——Servlet 与 JSP 集成的 MVC 方案**

（一）实验目的

1、掌握 JavaBeans 的编写要点；

2、掌握 MVC 设计模式的三个组成要素：模型 Model、视图 View 和控制器

Controller；

3、能在 Web 应用程序设计中熟练使用 MVC 模式。

（二）基本知识与原理

1、MVC 模式将交互式应用分成模型（Model）、视图（View）和控制器（Controller）

三个部分。

2、模型（Model）是指从现实世界中抽象出来的对象模型，是应用逻辑的反映。

模型封装了数据和数据的操作，是实际进行数据处理计算的地方。

3、视图（View）是应用和用户之间的接口，它负责将应用显现给用户和显示模

型的状态。

4、控制器（Controller）负责视图和模型之间的交互，控制对用户输入的响应、

响应方式和流程。它主要负责两方面的动作：把用户的请求分发到相应的模

型；将模型的改变及时反映到视图上。

5、MVC 设计模式将业务逻辑和显示逻辑进行了良好的分离，使代码更加清晰，

可维护性更好。

（三）实验内容及步骤

1、在工程 javaweb-prj1 中新建用户模型 UserBean.java，具体代码如下所示：

package cn.edu.zjut.model;

public class UserBean {

// 属性声明

private String username;private String password;

//构造方法

public UserBean(){}

// get 方法

public String **getUsername()**{ return username; }

public String **getPassword()**{ return password; }

// set 方法

public void **setUsername(String username)**{

this.username = username;

}

public void **setPassword(String password)**{

this.password = password;

}

}

2、在 javaweb-prj1 中新建视图 loginSuccess.jsp 和 loginFailed.jsp 页面，分别用于

显示登录成功和登录失败后的页面，具体代码如下所示：

<!-- loginSuccess.jsp 页面源代码 -->

<%@ page language="java" contentType="text/html; charset=GBK"

pageEncoding="GBK" import="cn.edu.zjut.model.UserBean"%>

<!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=GBK">

<title>登录成功</title>

</head>

<body>

**<% UserBean user=(UserBean)(request.getAttribute("USER")); %>**

登录成功,欢迎您,**<%=user.getUsername() %>**!

</body>

</html>

<!-- loginFailed.jsp 页面源代码 -->

<%@ page language="java" contentType="text/html; charset=GBK"

pageEncoding="GBK"%>

<!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=GBK">

<title>登录失败</title>

</head>

<body>登录失败，用户名或密码错误！

</body>

</html>

3、修改 javaweb-prj1 中的控制器 LoginController.java，实现如下功能：

（1） 调用 UserBean 用户模型记录用户信息；

（2） 如果用户名和密码均为 zjut，则跳转到视图 loginSuccess.jsp，否则跳转到

视图 loginFailed.jsp，具体代码如下所示：

package cn.edu.zjut;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import cn.edu.zjut.model.\*;

public class LoginController extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse

response) throws ServletException, IOException {

String username = request.getParameter("username");

String password = request.getParameter("password");

**UserBean user=new UserBean();**

**user.setUsername(username);**

**user.setPassword(password);**

if(checkUser(user)){

**request.setAttribute("USER", user);**

**RequestDispatcher dispatcher = request**

**.getRequestDispatcher("/loginSuccess.jsp");**

**dispatcher.forward(request, response);**

}else{

**response.sendRedirect("/javaweb-prj1/loginFailed.jsp");**

}

}

boolean checkUser(UserBean user){

if("zjut".equals(user.getUsername()) &&

"zjut".equals(user.getPassword())) {

return true;

}else{

return false;

}

}

}

4、使用 javaweb-prj1 工程中的 web.xml，为 LoginController 配置 url-pattern 映射；

5、将 javaweb-prj1 工程重新部署在 Tomcat 服务器上；

6、运行 login.jsp 页面，输入用户名和密码并运行，观察运行结果，如果出现运行错误，请尝试修正；

7、对 javaweb-prj1 工程做如下修改：

（1）修改视图 login.jsp，新增一个下拉框的表单域，用于选择用户类型（值为

“普通用户”和“管理员”）；

（2）修改用户模型 UserBean.java，新增一个用户类型 type 属性；

（3）修改控制器 LoginController.java，要求用户名和密码均为“zjut”，并且用

户类型为“管理员”才转到 loginSuccess.jsp 页面，否则转到 loginFailed 页面。

8、运行并观察修改后的结果，领会采用视图、模型和控制器分离表示逻辑和业

务逻辑的优点。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）根据实验过程，查找相关资料，总结 MVC 设计模式的优缺点；

（3）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

**三、扩展实验——JDBC 与 DAO 设计模式**

（一）实验目的

1、掌握数据库操作的基本 SQL 语句；

2、掌握连接数据库的基本步骤；

3、掌握利用 JDBC 操纵数据库的基本 API 和方法；

4、掌握 DAO 设计模式；

5、掌握在一个项目中集成 MVC 设计模式和 DAO 设计模式；

6、进一步理解 MVC 设计模型中模型和现实中实体的对应关系；

7、理解 DAO 设计模式中业务逻辑和数据访问逻辑相分离的优点。

（二）基本知识与原理

1、JDBC（Java Data Base Connectivity）是用于执行 SQL 语句的 Java API，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提供了一种基准用于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。

2、DAO（Data Access Object）设计模式将所有对数据源的访问操作抽象封装在一

个公共 API 接口中，其中定义了应用程序中将会用到的所有事务方法。当程

序开发人员需要和数据源进行交互的时候可直接使用这个接口，无需操纵数

据库。

（三）实验内容及步骤

1、下载和安装 PostgreSQL 数据库(版本 8.3)，下载链接地址：

http://www.enterprisedb.com/products-services-training/pgdownload#windows

；

2、创建一个用户名为 dbuser，密码为 dbpassword 的数据库登录角色，然后为该角色创建一个名称为 myDB 的数据库，并在该数据库中创建一个名称为

usertable 的数据表，具体表结构如表 1-1 所示：

表 1-1 usertable 数据表

**字段名称 类型 中文含义**

username Character varying(10) 登录用户名

password Character varying(10) 登录密码

type int 用户类型（1表示管理员，2表示普通用户）

3、在表 usertable 添加 3 条记录，具体如表 1-2 所示：

表 1-2 usertable 中的记录

**登录用户名 密码 用户类型**

zjut zjut 1

admin admin 1

temp temp 2

4、修改工程 javaweb-prj1：添加 PostgreSQL 驱动程序库文件 postgresql-8.4-702

.jdbc4.jar 到工程中；

5、修改工程 javaweb-prj1：创建数据库操作类 UserDAO.java，实现按用户名和密码校验用户是否合法的功能：

package cn.edu.zjut.dao;

import java.sql.\*;import javax.sql.\*;import javax.naming.\*;

import cn.edu.zjut.model.UserBean;public class UserDAO{

private static final String GET\_ONE\_SQL =

"SELECT \* FROM usertable WHERE username=? and password=?";

public UserDAO( ){ }

public Connection getConnection(){

Connection conn = null;

String driver = "org.postgresql.Driver";

String dburl = "jdbc:postgresql://127.0.0.1:5432/myDB";

String username = "dbuser"; //数据库登录用户名

String password = "dbpassword"; //数据库登录密码

try{

**Class.forName(driver);** //加载数据库驱动程序

**conn = DriverManager.getConnection(dburl,username,password);**

}catch( Exception e ){ e.printStackTrace(); }

return conn;

}

public boolean searchUser(UserBean user){

// 按用户名和密码校验用户是否合法

Connection conn = null;

PreparedStatement pstmt = null;

ResultSet rst=null;

try{

**conn = getConnection();**

**pstmt = conn.prepareStatement(GET\_ONE\_SQL);**

**pstmt.setString(1, user.getUsername());**

**pstmt.setString(2, user.getPassword());**

**rst = pstmt.executeQuery();**

if(rst.next()){

return true;

}

}catch(SQLException se){

se.printStackTrace();

return false;

}finally{

try{

pstmt.close();

conn.close();

}catch(SQLException se){ se.printStackTrace(); }

}

return false;

}

}

6、修改工程 javaweb-prj1 中的控制器 LoginController.java：

（1）引入 UserDAO 类，具体代码如下：……

import cn.edu.zjut.dao.UserDAO;

……

（2）重写 checkUser(UserBean user)方法：实现通过数据库表 usertable 的记录来

校验用户名和密码的合法性，具体修改代码如下所示：

……

boolean checkUser(UserBean user){

UserDAO ud=new UserDAO();

if( ud.searchUser(user) ) {

return true;

}

return false;

}

……

7、将 javaweb-prj1 工程重新部署在 Tomcat 服务器上；

8、运行 login.jsp 页面，输入用户名和密码并运行，观察运行结果，如果出现运

行错误，请尝试修正；

9、对 javaweb-prj1 工程进行如下修改：

（1）修改登录成功的条件为用户名、密码和用户类型三者匹配；

（2）新增一个用户注册视图 register.jsp，用于普通用户的注册；

（3）修改 UserDAO 类，新增一个 insert(UserBean user)方法，用于插入一条记录到 usertable 表中；

（4）运行并观察结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）查找相关资料，总结 DAO 设计模型的优点；

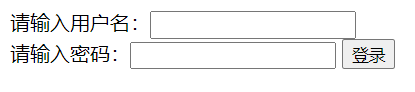
（3）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

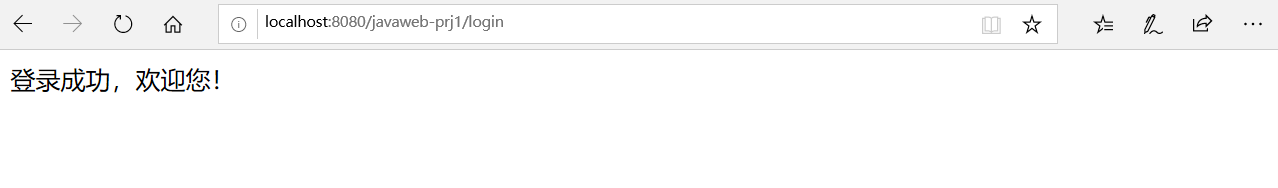
二、实验结果及问题分析

**（1）实验一运行结果：**

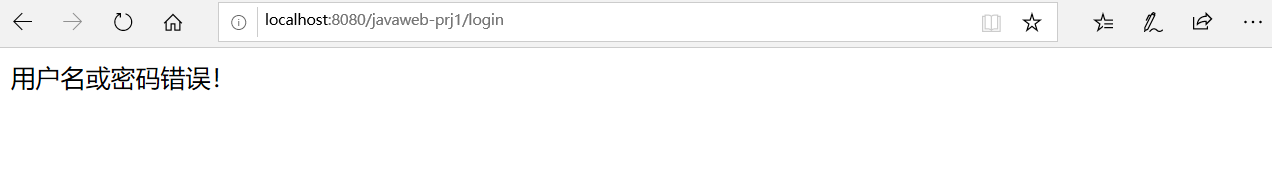
**login修改前：**



输入正确的用户名和密码，**登陆成功**。



输入不正确的用户名或密码，**登陆失败**。



**修改 login.jsp 页面，使用表格对表单域进行对齐排列：**

主要代码：

<body>

<form action=*"login"* method=*"post"*>

<table>

<tr><td>请输入用户名：</td><td><input name=*"username"* type=*"text"*></td></tr>

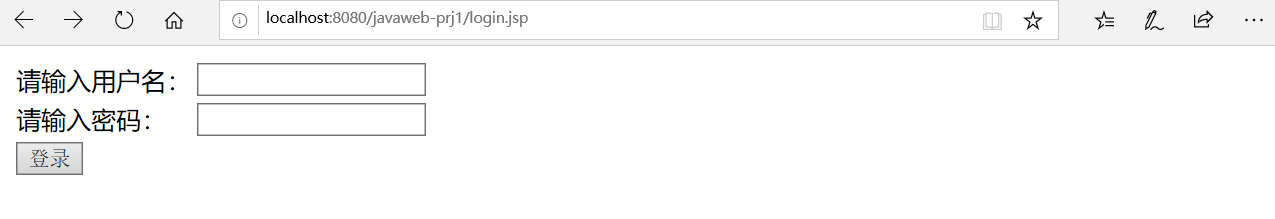
<tr><td>请输入密码：</td><td><input name=*"password"* type=*"password"*></td></tr>

<tr><td><input type=*"submit"* value=*"登录"*></td></tr>

</table>

</form>

</body>



**修改login.jsp 页面，使用 JavaScript 对表单域进行校验**

主要代码：

<script>

**function** checkForm()

{

**var** flag1=**true**,flag2=**true**;

**var** usernameValue=document.getElementById("username").value; //获取输入的用户名

**var** passwordValue=document.getElementById("password").value; //获取输入的密码

//alert(usernameValue+"----"+passwordValue); //此句是用来测试用

**if**(usernameValue==**null** || usernameValue=="")

{

**var** useMes=document.getElementById("s\_username");

useMes.innerHTML="<font color='red'> \*用户名不能为空</font>";

flag1=**false**;

}

**else**{

//var username = document.getElementById("username").value;

**var** reg\_username = /^\w{1,6}$/; //定义正则表达式

**var** flag1 = reg\_username.test(usernameValue); //判断值是否符合正则的规则

**var** s\_username = document.getElementById("s\_username"); //提示信息

**if**(!flag1){ //不符合的情况

s\_username.innerHTML = "<font color='red'> \*用户名不能超过 6 位</font>";

}

**else**{s\_username.innerHTML = "√";}

}

**if**(passwordValue==**null** || passwordValue=="")

{

**var** passMes=document.getElementById("s\_password");

passMes.innerHTML="<font color='red'> \*密码不能为空</font>";

flag2=**false**;

}

**else**{

//var password = document.getElementById("password").value;

**var** reg\_password = /^\w{1,6}$/;

**var** flag2 = reg\_password.test(passwordValue);

**var** s\_password = document.getElementById("s\_password");

**if**(!flag2){

s\_password.innerHTML = "<font color='red'> \*密码不能超过 6 位</font>";

}

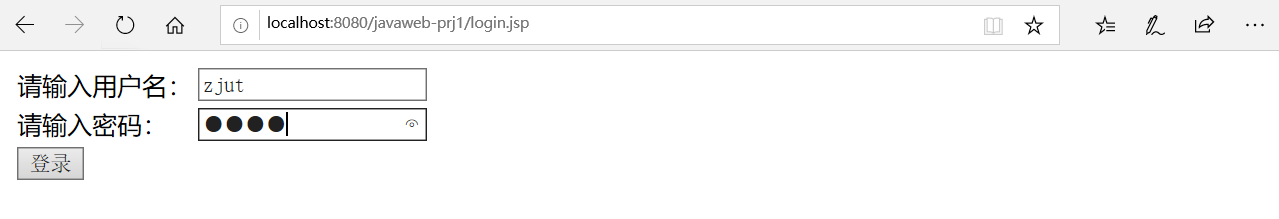
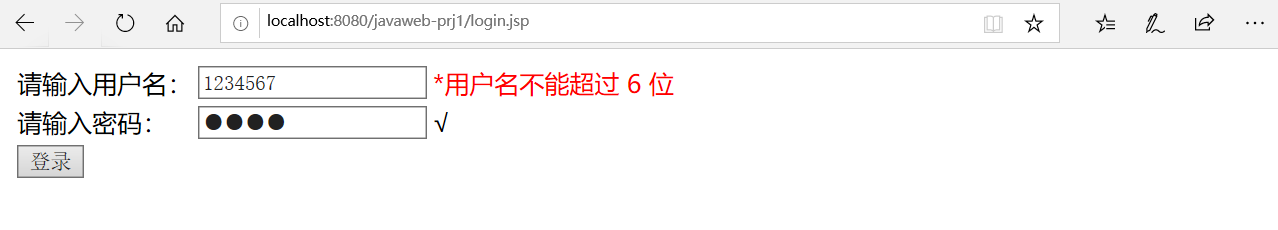
**else**{s\_password.innerHTML = "√";}

}

**return** flag1&&flag2;

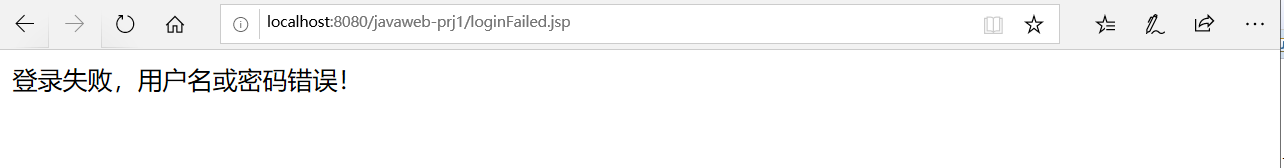
}

</script>

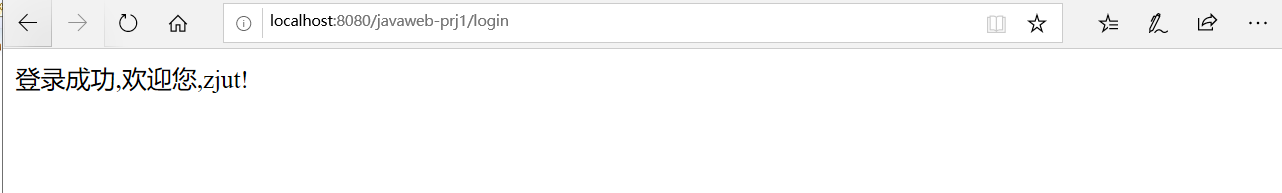


**（2）提高实验运行结果：**

输入错误，转到loginFailed.jsp

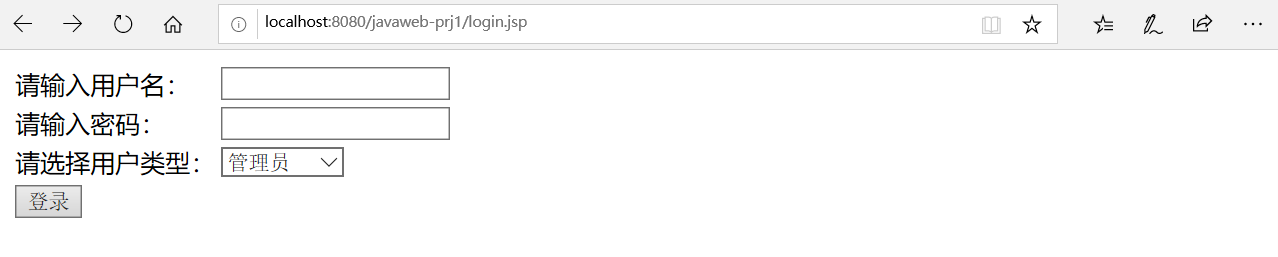


输入正确



**新增一个下拉框的表单域**

主要代码：

<tr><td>请选择用户类型：</td><td><select name=*"type"*><option value =*"administrator"*>管理员</option><option value =*"user"*>普通用户</option></select></td></tr>

**新增一个用户类型 type属性；用户名和密码均为“zjut”，并且用户类型是“管理员”才转到 loginSuccess.jsp 页面，否则转到 loginFailed页面。**

主要代码：

在User Bean.java中加入这几行代码

**private** int type;//用户类型

**public** int getType() {**return** type;}

**public** **void** setType(int type) {**this**.type=type;}

LoginController.java中修改代码为：

**boolean** checkUser(UserBean user){

**if**("zjut".equals(user.getUsername()) && "zjut".equals(user.getPassword())&& (user.getType()==1)){

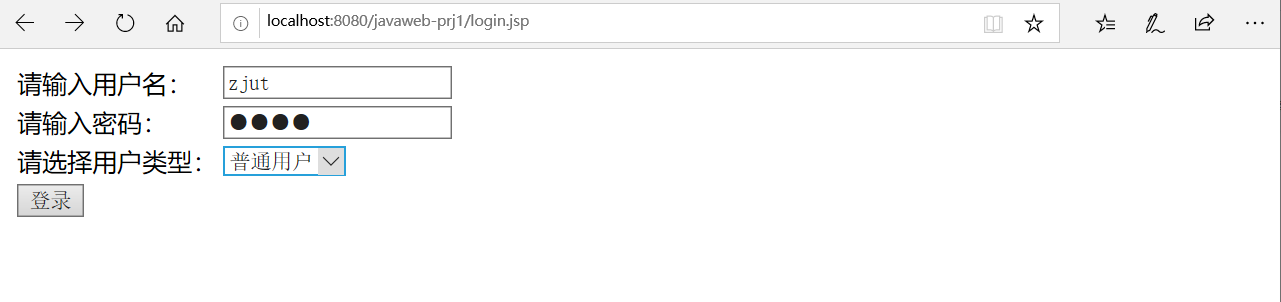
**return** **true**; }

**else**{

**return** **false**; }

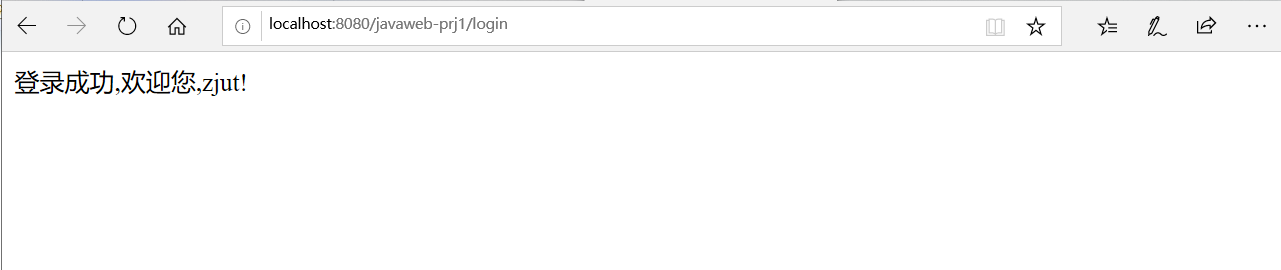
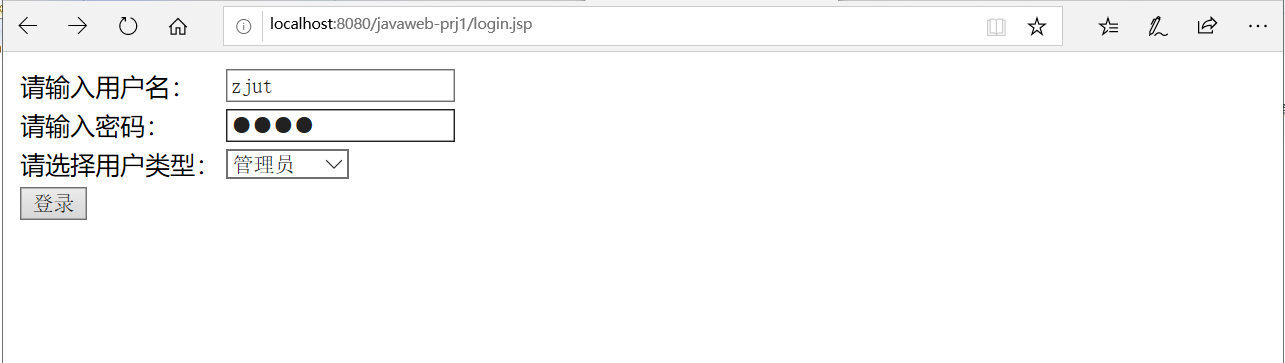
}

用户名和密码均为“zjut”，用户类型是“普通用户”转到loginFailed页面。



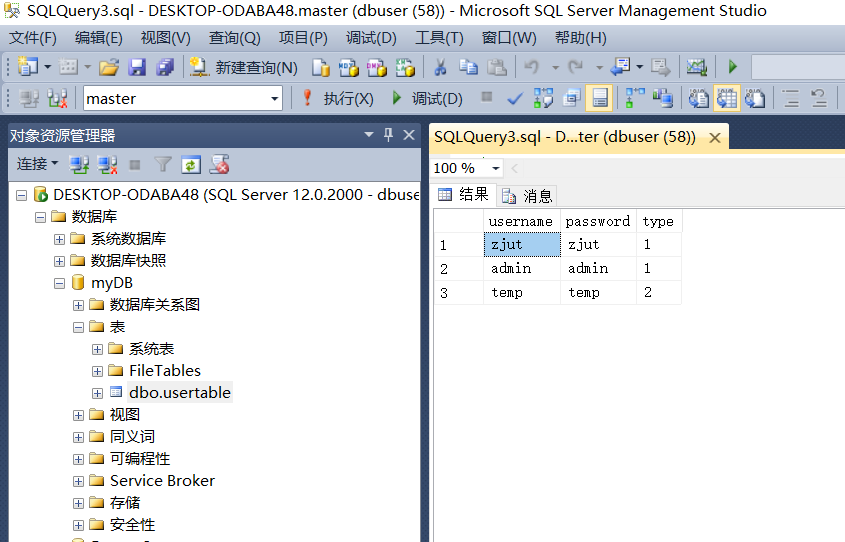


用户名和密码均为“zjut”，用户类型是“管理员”转到loginSuccess.jsp页面。

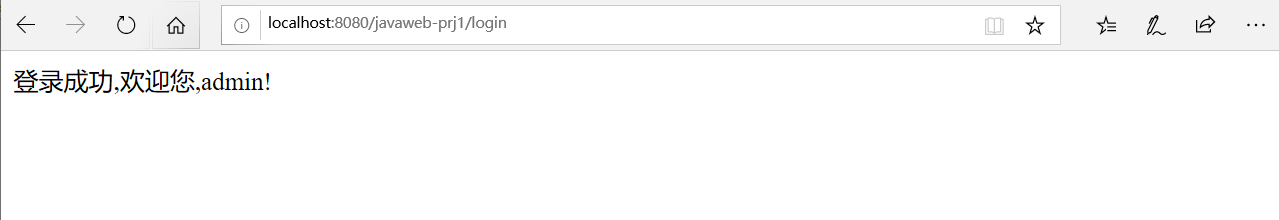
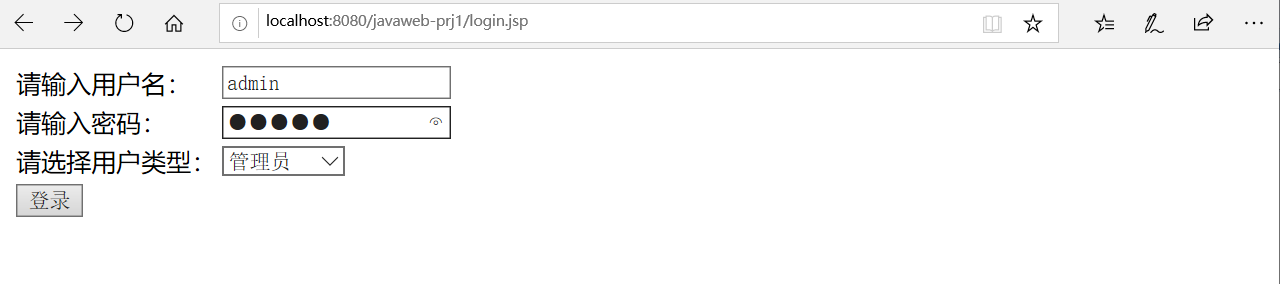


**（3）扩展实验运行结果**

**创建数据库以及用户表格：**



**登陆成功：**



修改登录成功的条件为用户名、密码和用户类型三者匹配：

主要修改代码：

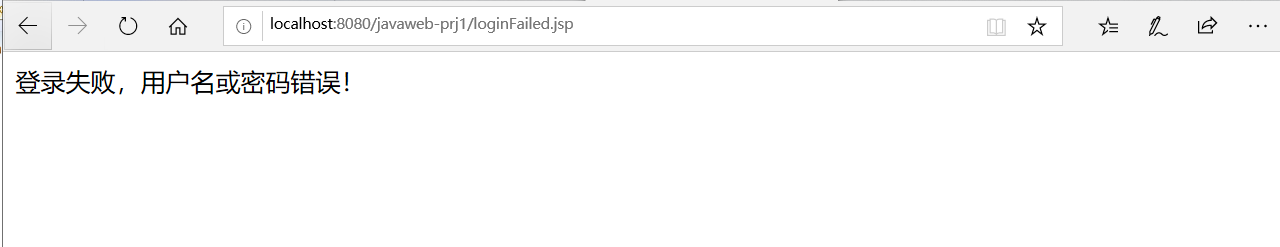
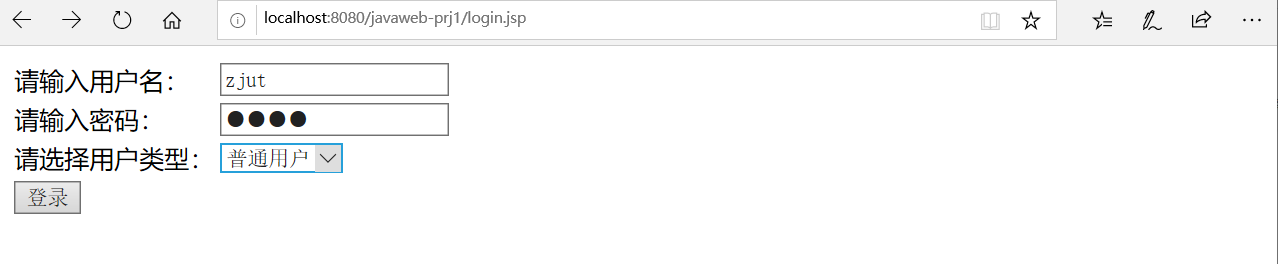
**if**(rst.next()){

**if**(rst.getInt("type")==user.getType()) **return** **true**;

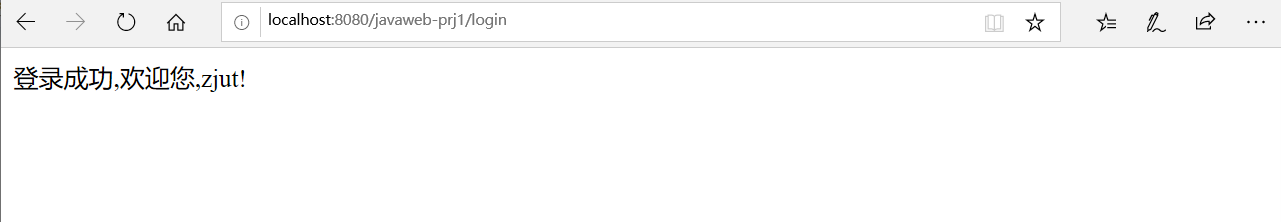
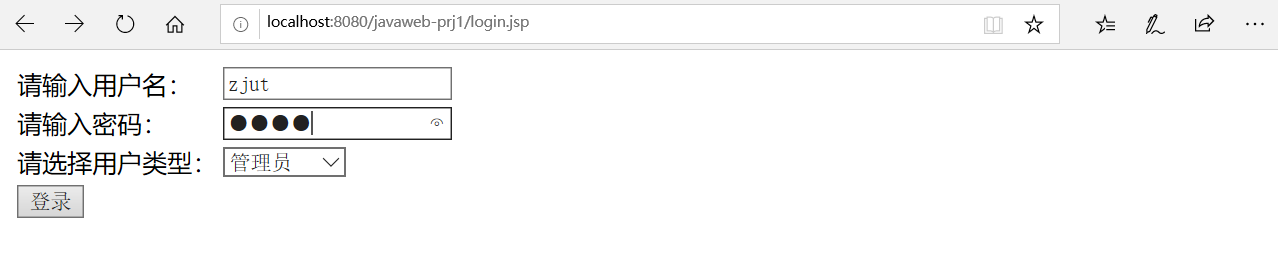
**else** **return** **false**;}

}

**用户名，密码，用户类型不匹配，登陆失败**

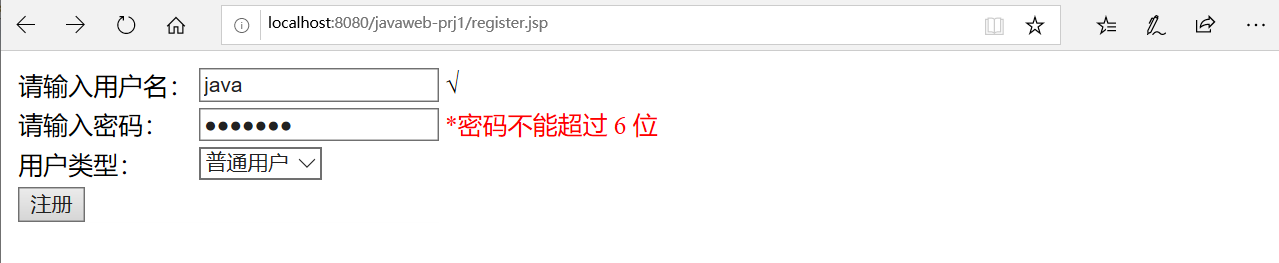


**用户名，密码，用户类型匹配，登陆成功**



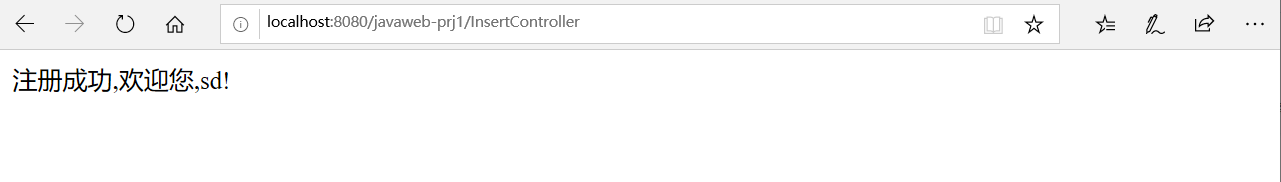
**注册用户（因格式问题注册失败）**



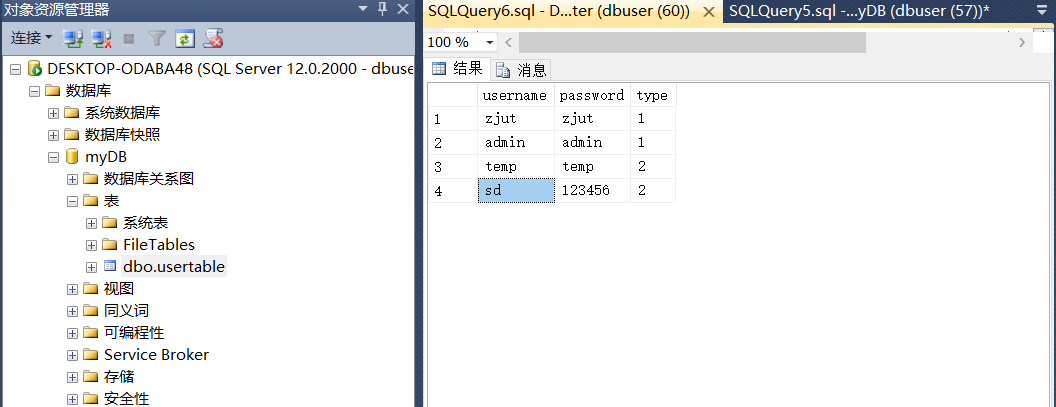


**注册成功**





**新注册员工插入到 usertable 表中**



1. 实验总结及收获

1.MVC模式的优缺点

1)优点：

各司其职、互不干涉；将业务逻辑和数据访问从表示层分离出来。

有利于开发中的分工；有利于组建的重用；

2)缺点：

系统结构和实现复杂；

视图与控制器过于紧密；

不适用于小型甚至中型应用程序；

1. DAO模式的优点

（1）数据存储逻辑的分离：一方面避免业务代码中混杂的JDBC代码，另一方面，数据访问接口与数据访问实现相分离。

（2）数据访问底层实现的分离：DAO模式将数据访问分为抽象层和实现层，分离了数据使用和数据访问的底层实现细节。这样可以在保持上层结构不变的情况下，通过更改底层实现来修改数据访问的机制。

（3）资源管理和调度的分离：数据访问逻辑从业务逻辑中脱离开来，使数据访问层实现统一的资源调度，通过数据库连接池和各种缓存机制的使用，可以保持上层系统不变的情况下来提高系统性能。

（4）数据抽象：通过对底层数据的封装，开发人员可以使用面向对象思想对数据进行操作。比如通过调用方法获取数据比通过SQL语句访问数据库获取数据，在代码上更易于理解，清晰，对日后维护带来便利。

3.本次实验再次复习了上学期学习的javaweb知识，三个实验循序渐进，熟悉了servlet，jsp技术，以及MVC和DAO模式，为之后JavaEE课程的学习打下了基础。