**广州航海学院**

课程设计（实训）报告



课 程  软件开发技术课程设计(Java)

题 目 图书管理系统

专 业 软件工厂

指导教师 秦宗蓉

班级 软工191 学号 17 姓名 陈彦宏

2021 年 7 月 1 日《软件开发技术课程设计(Java)》任务书

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 院 系 | | 信通学院 | 专 业 | 软件工程191、192 |
| 题 目 | | 图书管理系统 | | |
| 指导教师 | | 秦宗蓉、青宏燕 | | |
| 设计时间 | | 2021年6 月 21 日 至 2021年 7 月 2 日 共2周 | | |
| 课  程  设  计  要  求 | **一、完成一个图书管理系统**  **要求：**   1. **登录窗口，该登录窗口需要具备三个功能：登录、注册和重置。（下图仅供参考，可自行设计）**      * 登录：用户名和密码正确，将关闭该登录窗口，打开系统主窗体。用户名和密码都不允许为空。而且可以实现按下回车键从用户名文本框跳转到密码框，以及最后执行登录功能操作。 * 注册：如果没有账号，可以直接单击注册按钮，进入注册窗口，进行注册。 * 重置：将用户名和密码清空，并将焦点定位到用户名文本框。   **2、主窗体，需要至少有3个菜单：用户管理、图书管理、图书类型管理（下图仅供参考，可自行设计）**     * 用户管理：主要是针对用户实体进行管理，实现增删改查 * 图书管理：主要是针对图书实体进行管理，实现增删改查 * 图书类型管理：主要是针对图书类型实体进行管理，实现增删改查   每个菜单的菜单项可以自行设计，并且请大家自行扩展功能，比如图书借阅功能等等，体现独创性和设计性。  **3、数据库设计**  需要包含用户、图书和图书类型几个实体，可以自行扩展哦，比如图书借阅等等。  **4、建议采用分层设计**  下图仅供参考。    **二、撰写课程设计报告**  要求大家在完成课程设计的同时，以书面形式交一份课程设计报告（用A4纸打印成册）。   * 课程设计报告的格式按照提供的格式，尤其是封面一定要严格按照标准格式设计； * 程序代码尽量实现重用，代码需要加上注释。 * 课程设计报告最后还需要写心得体会，总结设计中遇到的问题和解决方法。 * 自动生成简要的目录。   **三、其它要求**  准时参加课程设计安排的课程，不迟到、早退，生病或有事应在课前请假，考勤情况将计入课程设计总评成绩。 | | | |

目录

[一、 需求分析 5](#_Toc76078829)

[图书管理系统功能结构图 5](#_Toc76078830)

[二、 数据库设计 5](#_Toc76078831)

[1.对应数据需求描述 5](#_Toc76078832)

[2.数据库概念模型设计 6](#_Toc76078833)

[3.数据库逻辑模型设计 8](#_Toc76078834)

[三、 项目环境搭建 9](#_Toc76078835)

[四、 实体类设计 9](#_Toc76078836)

[1.用户实体类： 9](#_Toc76078837)

[2.图书实体类： 11](#_Toc76078838)

[3.图书类型实体： 14](#_Toc76078839)

[五、 工具类设计 16](#_Toc76078840)

[1.获取数据库连接类： 16](#_Toc76078841)

[2.判断字符串空串类： 17](#_Toc76078842)

[六、 各功能模块设计 18](#_Toc76078843)

[1.登录模块设计： 18](#_Toc76078844)

[2.注册模块设计 22](#_Toc76078845)

[3.主体模块设计 28](#_Toc76078846)

[4.添加功能模块 29](#_Toc76078847)

[5.查询功能模块 33](#_Toc76078848)

[6.修改模块设计 36](#_Toc76078849)

[7.删除功能模块 41](#_Toc76078850)

[七、 总结 44](#_Toc76078851)

[1.设计中遇到的问题以及解决方案 44](#_Toc76078852)

[2.心得体会 44](#_Toc76078853)

# 需求分析

本实训要求设计一个图书管理系统，利用Java语言与MySql数据库连接实现存储数据和数据的动态查询以及更改。系统要求有登录界面，注册界面，并在主界面中能对用户信息，图书信息以及图书类型信息进行管理，实现增删改查操作。并且能够将结果写入数据库。

功能需求描述：

1. 用户管理：可以实现对不同用户进行添加，查询，删除和修改。
2. 图书管理：可以实现对图书的添加，查询，删除和修改。
3. 图书类型管理：可以实现对图书的添加，查询，删除和修改。
4. 登录：用户名和密码正确，将关闭该登录窗口，打开系统主窗体。用户名和密码都不允许为空。而且可以实现按下回车键从用户名文本框跳转到密码框，以及最后执行登录功能操作。
5. 注册：如果没有账号，可以直接单击注册按钮，进入注册窗口，进行注册，用户名和密码不允许为空，且有二次确认密码。注册成功后点击登录即可打开登录界面。

图书管理系统功能结构图

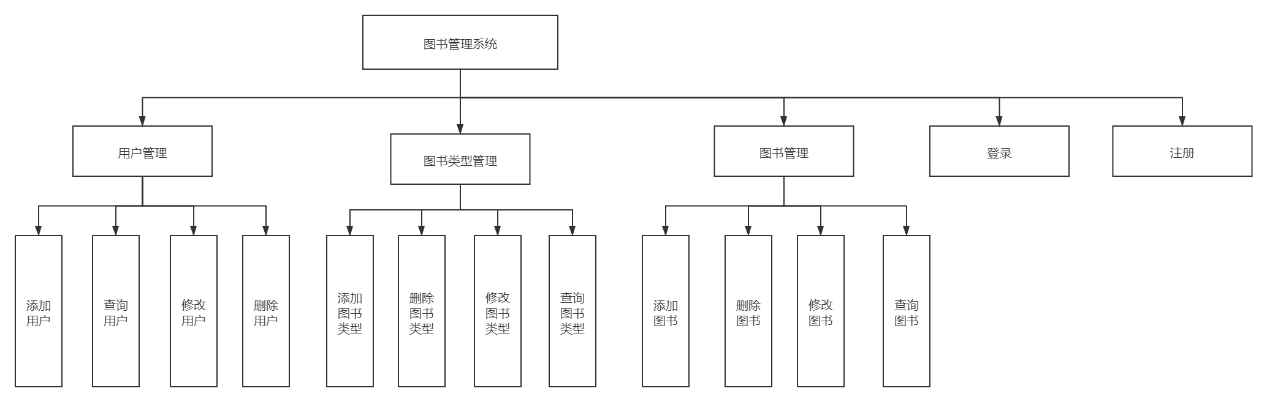


图 1图书管理系统功能结构图

# 数据库设计

1.对应数据需求描述

1. **用户信息数据需求**

|  |  |
| --- | --- |
| 数据项 | 数据需求 |
| 用户编号 | 用于在数据库中用户实体的唯一标识。自动增加，不可修改 |
| 用户名 | 20个字符以内 |
| 密码 | 20个字符以内 |

表格 1用户信息数据需求

**2.图书信息数据需求**

|  |  |
| --- | --- |
| 数据项 | 数据需求 |
| 图书编号 | 用于在数据库中图书实体的唯一标识。自动增加，不可修改 |
| 图书名称 | 20个字符以内 |
| 作者名称 | 20个字符以内 |
| 价格 | Floa类型 |
| 图书类型编号 | 对应图书类型信息的唯一编号，随着图书类型表改变而改变 |
| 图书描述 | 1000个字符以内 |

表格 2图书信息数据需求

**3.图书类型信息数据需求**

|  |  |
| --- | --- |
| 数据项 | 数据需求 |
| 图书类型编号 | 用于在数据库中图书类型实体的唯一标识。自动增加，不可修改 |
| 图书类型名称 | 20个字符以内 |
| 图书类型信息 | 20个字符以内 |

表格 3图书类型信息数据需求

2.数据库概念模型设计

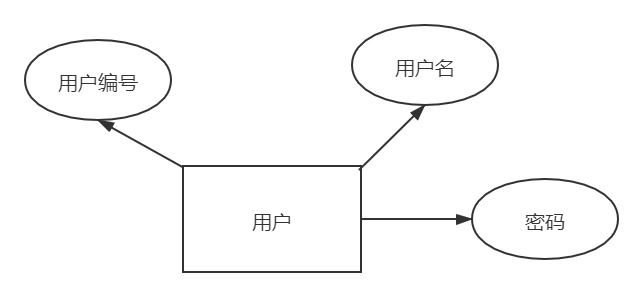


图 2用户实体

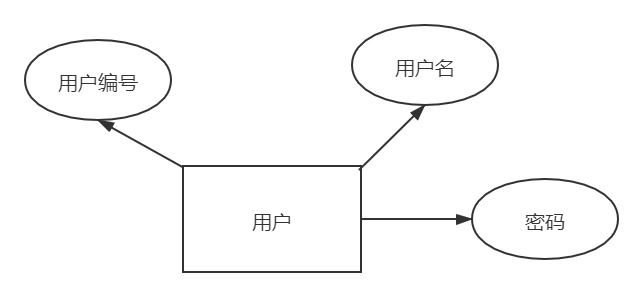


图 3图书实体

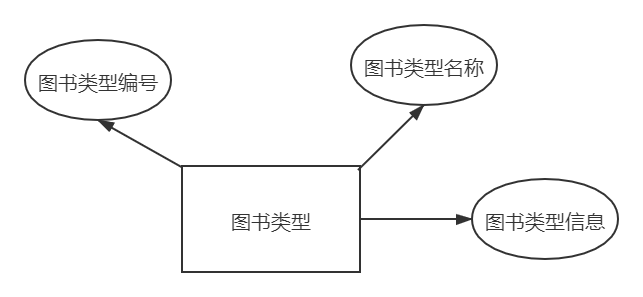


图 4图书类型实体

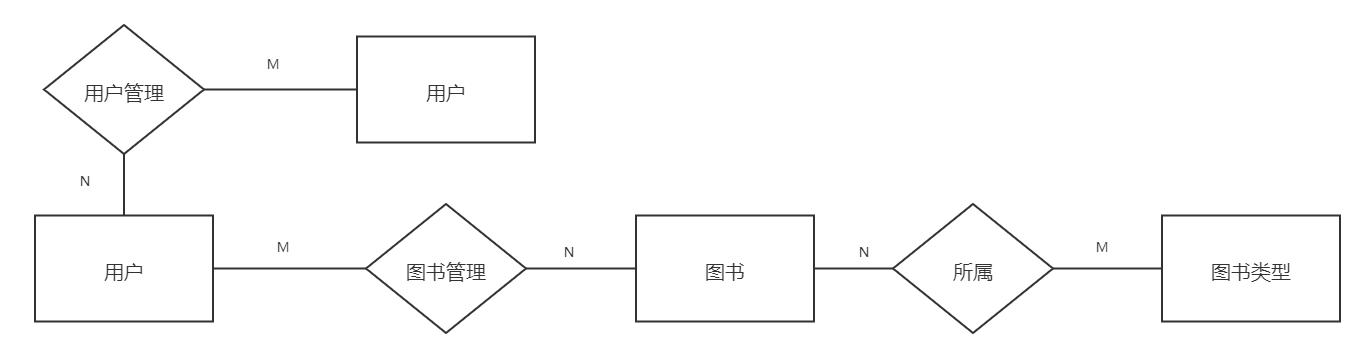


图 5各实体关系图

3.数据库逻辑模型设计

根据E-R图，将其关系模型转化为三个关系模式：

用户（用户编号、用户名、密码）

图书（图书编号、图书名、作者、价格、图书类型编号、图书描述）

图书类型（图书类型编号、图书类型名称、图书类型信息）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 是否为空 | 默认值 | 是否主键 |
| 用户编号（id） | Int（11） | Not null | 自动增加 | 主键 |
| 用户名（username） | Varchar（20） | Not null |  |  |
| 密码（password） |  | Not null |  |  |

表格 4用户表（user）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 是否为空 | 默认值 | 是否主键 |
| 图书编号（id） | Int（11） | Not null | 自动增加 | 主键 |
| 图书名称（bookName） | Varchar（20） | Not null |  |  |
| 作者（author） | Varchar（20） | Not null |  |  |
| 价格（price） | Float | Not null |  |  |
| 图书类型编号（bookTypeId） | Int（11） | Not null |  | 外键 |
| 图书描述（bookDesc） | Varchar（1000） |  |  |  |

表格 5图书表（book）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 是否为空 | 默认值 | 是否为主键 |
| 图书类型编号（id） | Int（11） | Not null | 自动增加 | 主键 |
| 图书类型名称（bookTypeName） | Varchar（20） | Not null |  |  |
| 图书类型信息(bookTypeDesc) | Varchar(1000) | Not null |  |  |

表格 6图书类型表（booktype）

# 项目环境搭建

1. **开发环境：**
2. windows系统环境：使用的是WIN7系统
3. 数据库为MySQL5.7.25.0
4. 使用SQLyog-12.4.3管理数据库
5. JDK环境：JDK15
6. Eclipse编写java程序
7. **Jar包：**



1. **WindowBuilder插件**

# 实体类设计

1.用户实体类：

**package** com.bk.model;

//用户实体

**public** **class** User {

**private** **int** id;//编号

**private** String username;//用户名

**private** String password;//密码

**public** User() {

**super**();

// **TODO** 自动生成的构造函数存根

}

**public** User(String username, String password) {

**super**();

**this**.username = username;

**this**.password = password;

}

**public** User(**int** id, String username) {

**super**();

**this**.id = id;

**this**.username = username;

}

**public** User(**int** id, String username, String password) {

**super**();

**this**.id = id;

**this**.username = username;

**this**.password = password;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getUsername() {

**return** username;

}

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

**this**.username = username;

}

**public** String getPassword() {

**return** password;

}

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

**this**.password = password;

}

}

2.图书实体类：

package com.bk.model;

//图书实体

public class Book {

private int id;//编号

private String bookName;//图书名

private String author;//作者

private Float price;//价格

private Integer bookTypeId;//图书类型编号

private String bookTypeName;//图书类型名称

private String bookDesc;//图书简介

public Book() {

super();

// TODO 自动生成的构造函数存根

}

public Book(String bookName, String author, Float price, Integer bookTypeId, String bookDesc) {

super();

this.bookName = bookName;

this.author = author;

this.price = price;

this.bookTypeId = bookTypeId;

this.bookDesc = bookDesc;

}

public Book(String bookName, String author, Integer bookTypeId) {

super();

this.bookName = bookName;

this.author = author;

this.bookTypeId = bookTypeId;

}

public Book(int id, String bookName, String author, Float price, Integer bookTypeId, String bookDesc) {

super();

this.id = id;

this.bookName = bookName;

this.author = author;

this.price = price;

this.bookTypeId = bookTypeId;

this.bookDesc = bookDesc;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getBookName() {

return bookName;

}

public void setBookName(String bookName) {

this.bookName = bookName;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public Float getPrice() {

return price;

}

public void setPrice(Float price) {

this.price = price;

}

public Integer getBookTypeId() {

return bookTypeId;

}

public void setBookTypeId(Integer bookTypeId) {

this.bookTypeId = bookTypeId;

}

public String getBookTypeName() {

return bookTypeName;

}

public void setBookTypeName(String bookTypeName) {

this.bookTypeName = bookTypeName;

}

public String getBookDesc() {

return bookDesc;

}

public void setBookDesc(String bookDesc) {

this.bookDesc = bookDesc;

}

}

3.图书类型实体：

package com.bk.model;

//图书类型实体

public class BookType {

private int id;//编号

private String bookTypeName;//图书类型名

private String bookTypeDesc;//图书类型描述

public BookType() {

super();

// TODO 自动生成的构造函数存根

}

public BookType(String bookTypeName, String bookTypeDesc) {

super();

this.bookTypeName = bookTypeName;

this.bookTypeDesc = bookTypeDesc;

}

public BookType(int id, String bookTypeName, String bookTypeDesc) {

super();

this.id = id;

this.bookTypeName = bookTypeName;

this.bookTypeDesc = bookTypeDesc;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getBookTypeName() {

return bookTypeName;

}

public void setBookTypeName(String bookTypeName) {

this.bookTypeName = bookTypeName;

}

public String getBookTypeDesc() {

return bookTypeDesc;

}

public void setBookTypeDesc(String bookTypeDesc) {

this.bookTypeDesc = bookTypeDesc;

}

@Override

public String toString() {

return bookTypeName;

}

}

# 工具类设计

1.获取数据库连接类：

package com.bk.util;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import com.mysql.jdbc.Driver;

public class DbUtil {

private String dbDriver="com.mysql.jdbc.Driver";//驱动名称

private String dbUrl="jdbc:mysql://localhost:3306/bookmanager?characterEncoding=utf8&useSSL=true";//url

private String dbusername="root";//数据库用户名

private String dbpassword="1234";//数据库密码

//连接数据库

public Connection getconn()throws Exception{

Class.forName(dbDriver);

Connection conn=DriverManager.getConnection(dbUrl,dbusername,dbpassword);

return conn;

}

//关闭连接

public void clossconn(Connection conn)throws SQLException{

if(conn!=null) {

conn.close();

}

}

}

2.判断字符串空串类：

**package** com.bk.util;

//工具类，进行字符串空串等的工具

**public** **class** ToolUtil {

//判断字符串是否为空

**public** **static** Boolean isEmpty(String str) {

**if**(str==**null** || "".equals(str.trim())) {

**return** **true**;

}

**else** {

**return** **false**;

}

}

//判断字符串不为空

**public** **static** Boolean isnotEmpty(String str) {

**if**(str!=**null** && !"".equals(str.trim())) {

**return** **true**;

}

**else** {

**return** **false**;

}

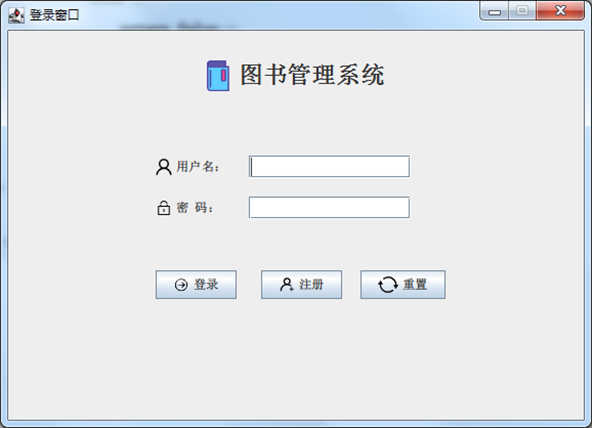
}

}

# 各功能模块设计

1.登录模块设计：

1. 窗体设计：使用WindowsBuilder插件设计。输入用户名文本框名为userNameTxt，输入密码的文本框名为passwordTxt



1. 数据库操作类（UserDao）中实现的方法

//设置登录验证，登录成功，会返回登录用户的所有信息

**public** User login(Connection conn,User user)**throws** SQLException{

User resultuser=**null**;//声明用户

String sql ="select \* from user where username=? and password=?";//预处理

PreparedStatement pstmt=conn.prepareStatement(sql);//创建预处理对象

pstmt.setString(1, user.getUsername());//设置第一个?的值

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

ResultSet rs=pstmt.executeQuery();//如果查询成功，返回结果集对象

**if**(rs.next()) {

resultuser=**new** User();//如果有数据，返回用户实例化

resultuser.setId(rs.getInt("id"));

resultuser.setUsername(rs.getString("username"));//设置用户实例的用户名为结果集中的用户名

resultuser.setPassword(rs.getString("password"));

}

**return** resultuser;//返回用户的实体信息

}

1. 按钮事件
2. 登录按钮事件：点击登录按钮后，会判断文本框是否为空，若为空则会提示（用户名或密码为空），输入完成后会判断数据库中是否有该用户，以及密码是否正确。若找不到用户名或者密码错误则提示（用户名或密码错误）若正确则会提示登录成功，然后关闭登录窗口，打开图书管理系统主界面。下面为登录按钮事件的代码：

//登录事件处理

**protected** **void** loginActionPerfomed(ActionEvent e) {

String userName=**this**.userNameTxt.getText();

String password=**new** String(**this**.passwordTxt.getText());

**if**(ToolUtil.*isEmpty*(userName) || ToolUtil.*isEmpty*(password)) {

JOptionPane.*showMessageDialog*(**null**, "用户名或密码不能为空");

**return**;

}

User user=**new** User(userName,password);//封装一个用户进行查询

Connection conn=**null**;

**try** {

conn=dbUtil.getconn();

User currentuser=userDao.login(conn, user);

**if**(currentuser!=**null**) {

//登录成功一定会返回一条用户的所有信息

JOptionPane.*showMessageDialog*(**null**,"登陆成功！");

**this**.dispose();

**new** Mainframe().setVisible(**true**);

}

**else** {

JOptionPane.*showMessageDialog*(**null**, "用户名或密码错误");

}

} **catch** (Exception e2) {

// **TODO**: handle exception

e2.printStackTrace();

}**finally** {

**try** {

dbUtil.clossconn(conn);

} **catch** (SQLException e1) {

// **TODO** 自动生成的 catch 块

e1.printStackTrace();

}

}

}

1. 注册按钮事件：点击注册按钮将会关闭登录窗口，打开注册窗口。下面为注册按钮事件的代码：

//注册事件处理

**private** **void** registerActionPerformed(ActionEvent e) {

// **TODO** 自动生成的方法存根

**this**.dispose();

**new** Register().setVisible(**true**);

}

③重置按钮事件：点击重置按钮，会将文本框的内容清空。下面为重置按钮事件的代码：

//重置事件处理

**private** **void** resetValueActionPerformed(ActionEvent evt) {

**this**.userNameTxt.setText("");

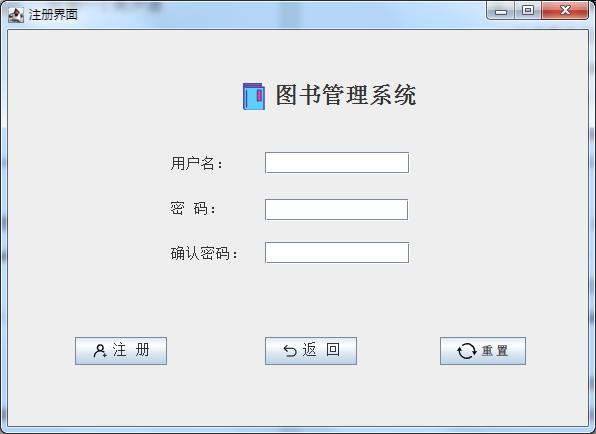
**this**.passwordTxt.setText("");

}

2.注册模块设计

1. 窗体设计

使用WindowsBuilder插件设计。输入用户名文本框名为userNameTxt，输入密码的文本框名为passwordTxt，确认密码输入的文本框名为c\_passwordTxt。



1. 数据库操作类（UserDao）实现的方法

//用户添加

**public** **int** add(Connection conn,User user)**throws** SQLException{

String sql="insert into user values(null,?,?)";

PreparedStatement pstmt=conn.prepareStatement(sql);

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

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

**return** pstmt.executeUpdate();

}

//是否有重复用户

**public** **boolean** existUserName(Connection conn,String userName)**throws** SQLException{

String sql="select \* from user where username=?";

PreparedStatement pstmt=conn.prepareStatement(sql);

pstmt.setString(1, userName);

ResultSet rs=pstmt.executeQuery();

**return** rs.next();

}

1. 按钮事件
2. 注册按钮事件：点击注册按钮，会判断文本框是否为空，输入是否符合要求，会判断两次输入是否正确，以及查询数据库中是否已经有相同的用户名。对于以上判断，发现错误时都会有提示窗口。用户名限制为20个字符以内且不能有空格，密码则是通过正则表达式限制在6-20个字符，且只能包含数字字母以及下划线。注册成功后也会有弹窗提示并且可以清空文本框。以下为注册按钮的事件代码：

//注册事件处理

private void userAddActionPerformed(ActionEvent evt) {

// TODO 自动生成的方法存根

String userName=this.userNameTxt.getText();

String password=new String(this.passwordTxt.getPassword());

String c\_password=new String(this.c\_passwordTxt.getPassword());

boolean flag=password.equals(c\_password);

String matches="^\\w{6,20}$";

boolean flag2=password.matches(matches);

if(ToolUtil.isEmpty(userName)) {

JOptionPane.showMessageDialog(null, "用户名不能为空！");

return;

}

if(userName.indexOf(" ")!=-1) {

JOptionPane.showMessageDialog(null, "用户名不能包含空格！");

return;

}

if(ToolUtil.isEmpty(password)) {

JOptionPane.showMessageDialog(null, "密码不能为空！");

return;

}

if(flag2==false) {

JOptionPane.showMessageDialog(null, "密码为6-20位且只能由字母，数字，下划线组成！");

return;

}

if(ToolUtil.isEmpty(c\_password)) {

JOptionPane.showMessageDialog(null, "请确认密码！");

return;

}

if(userName.length()>20) {

JOptionPane.showMessageDialog(null, "用户名必须小于20位");

return;

}

if(flag==false) {

JOptionPane.showMessageDialog(null, "两次输入的密码不相同!");

return;

}

User user=new User(userName,password);

Connection conn=null;

try {

conn=dbUtil.getconn();

boolean flag3=userDao.existUserName(conn, userName);

if(flag3) {

JOptionPane.showMessageDialog(null, "用户名已存在!");

return;

}

int n=userDao.add(conn, user);

if(n==1) {

JOptionPane.showMessageDialog(null, "注册成功！");

resetValue();

}

else {

resetValue();

JOptionPane.showMessageDialog(null, "注册失败！");

}

} catch (Exception e) {

// TODO: handle exception

e.printStackTrace();

JOptionPane.showMessageDialog(null, "注册失败！");

}finally {

try {

dbUtil.clossconn(conn);

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

}

}

1. 返回按钮事件：点击返回按钮后，将关闭注册窗口，打开登录窗口。以下为返回按钮事件的代码：

//返回事件处理

private void returnActionPerformed(ActionEvent e) {

// TODO 自动生成的方法存根

this.dispose();

new Loginframe().setVisible(true);

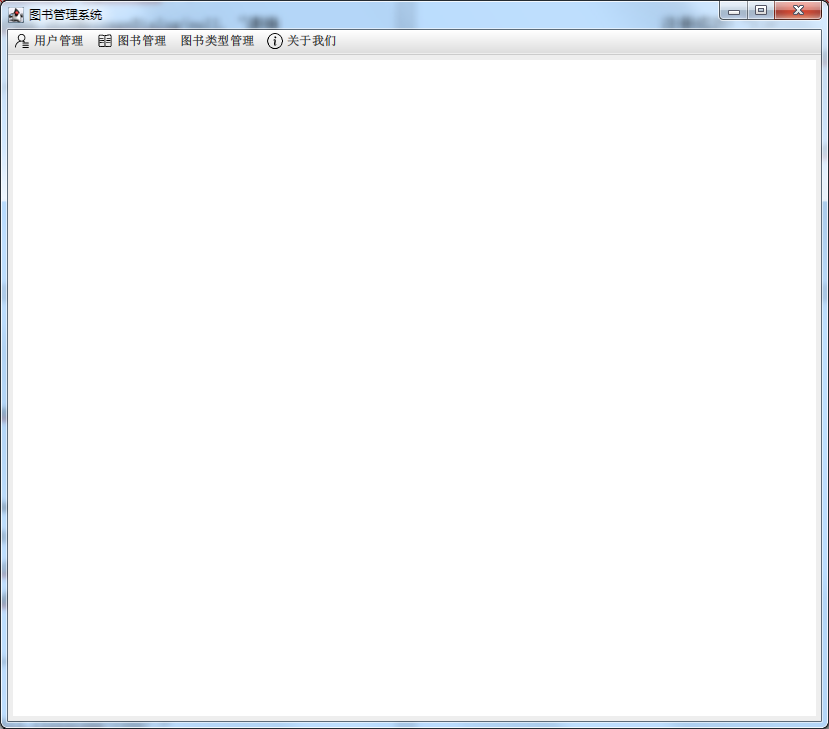
}

1. 重置按钮：点击重置按钮，将会清空文本框。事件代码为通用操作与上面给出的登录界面的代码类似。

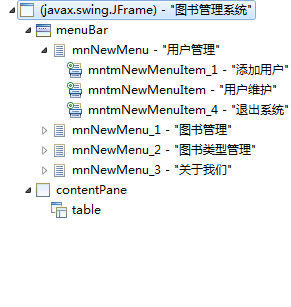
3.主体模块设计

1. 窗体设计

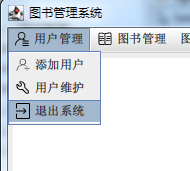
使用WindowsBuilder插件设计。



菜单栏弹出的子窗体为JInternalFrame可以在主窗体内显示，窗体布局选择用GroupLayout，窗体的具体设计如下



1. 菜单栏点击事件
2. 退出事件：



点击退出系统后，会出现弹出，让你选择是否退出以及取消，点击是则退出系统，否和取消都会返回主窗口。具体代码如下：

public void actionPerformed(ActionEvent e) {

int result=JOptionPane.showConfirmDialog(null, "是否退出系统");

if(result==0) {

dispose();

}

}

1. 各类子窗口打开事件

由于打开各类子窗口的代码为通用操作，下面只展示打开添加用户窗口的代码：

public void actionPerformed(ActionEvent e) {

Useradd Useradd =new Useradd();

Useradd.setVisible(true);

table.add(Useradd);

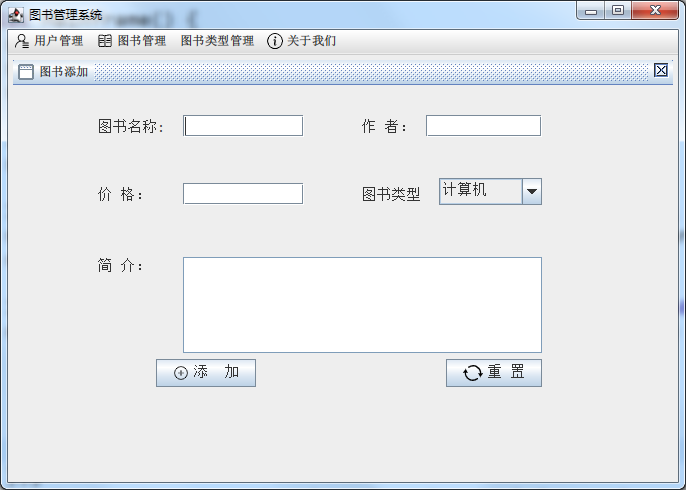
}

4.添加功能模块

对用户，图书以及图书类型三个实体的添加功能设计以及添加按钮事件的处理基本类似，其中用户添加功能的设计与注册相同，下面展示图书添加功能的设计。

1. 窗体设计

使用WindowsBuilder插件设计。输入图书名称的文本框名为bookNameTxt，输入作者的文本框名为authorTxt，输入价格的文本框名为priceTxt，图书类型下拉框名为bookTypeJcb，输入简介的文本框名为bookDesc。当用户输入完成后，点击添加按钮即可实现查找。



1. 数据库操作类（BookDao）实现的方法

在BookDao.java控制类中的图书查询功能实现

//图书添加

**public** **int** add(Connection conn,Book book)**throws** SQLException{

String sql="insert into book values(null,?,?,?,?,?)";

PreparedStatement pstmt=conn.prepareStatement(sql);

pstmt.setString(1, book.getBookName());

pstmt.setString(2, book.getAuthor());

pstmt.setFloat(3, book.getPrice());

pstmt.setInt(4, book.getBookTypeId());

pstmt.setString(5, book.getBookDesc());

**return** pstmt.executeUpdate();

}

1. 按钮事件设计
2. 添加按钮事件

当点击添加按钮时，会对文本框的内容以及下拉框的选择进行判断，能够判断文本框是否为空，简介允许为空，以及错误输入的判断，对于不输入以及错误输入会有弹窗提示。添加成功或失败后也会有弹窗提示，并且会清空文本框内容。下面为添加事件处理的代码：

//图书添加事件处理

**protected** **void** bookAddActionPerformed(ActionEvent evt) {

// **TODO** 自动生成的方法存根

String bookName=**this**.bookNameTxt.getText();

String author=**this**.authorTxt.getText();

String price=**this**.priceTxt.getText();

String bookDesc=**this**.bookDescTxt.getText();

**if**(ToolUtil.*isEmpty*(bookName)) {

JOptionPane.*showMessageDialog*(**null**, "图书名称不能为空！");

**return**;

}

**if**(ToolUtil.*isEmpty*(author)) {

JOptionPane.*showMessageDialog*(**null**, "作者不能为空！");

**return**;

}

**if**(ToolUtil.*isEmpty*(price)) {

JOptionPane.*showMessageDialog*(**null**, "价格不能为空！");

**return**;

}

**if**(price.indexOf(" ")!=-1) {

JOptionPane.*showMessageDialog*(**null**, "价格不能包含空格！");

**return**;

}

BookType bookType=(BookType)bookTypeJcb.getSelectedItem();

**int** bookTypeId=bookType.getId();

Book book=**new** Book(bookName, author, Float.*parseFloat*(price), bookTypeId, bookDesc);

Connection conn=**null**;

**try** {

conn=dbUtil.getconn();

**int** addNum=bookDao.add(conn, book);

**if**(addNum==1) {

JOptionPane.*showMessageDialog*(**null**, "图书添加成功！");

resetValue();

}

**else** {

JOptionPane.*showMessageDialog*(**null**, "图书添加失败！");

}

} **catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

JOptionPane.*showMessageDialog*(**null**, "图书添加失败！");

}**finally** {

**try** {

dbUtil.clossconn(conn);

} **catch** (Exception e2) {

// **TODO**: handle exception

e2.printStackTrace();

}

}

}

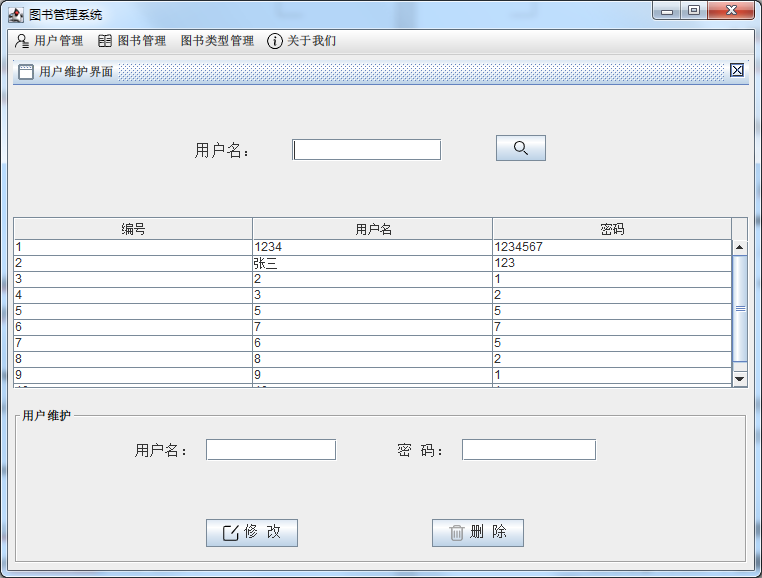
1. 重置按钮事件处理：点击重置按钮，将会清空文本框。事件代码为通用操作与上面给出的登录界面的代码类似。

5.查询功能模块

对用户，图书以及图书类型三个实体的查询功能设计以及查询按钮事件的处理基本类似，下面展示用户查询功能的设计。

1. 窗体设计

窗体使用WindowsBuilder插件设计，查询功能其中使用到的文本框为第一个输入用户名的文本框名为s\_userNameTxt，用于查询。窗体中有设置一个JTable名为userTable显示数据库中的user表的内容，且会根据操作动态显示。当用户输入完成后，点击查询按钮即可实现查找。



1. 数据库操作类（UserDao）实现的方法

使用的是模糊查询实现。

//用户查询

**public** ResultSet list(Connection conn,User user)**throws** SQLException{

StringBuffer sb=**new** StringBuffer("select \* from user");

//判断用户名文本框是否为空，不为空，则有条件

**if**(ToolUtil.*isnotEmpty*(user.getUsername())) {

sb.append(" and username like '%"+user.getUsername()+"%'");

}

//sb转化为字符串并且将第一个的单词and改成where形成条件

PreparedStatement pstmt=conn.prepareStatement(sb.toString().replaceFirst("and", "where"));

**return** pstmt.executeQuery();

}

1. 初始化表格方法

将数据库中的user表的内容添加到userTable中显示

//初始化表格

private void fillTable(User user) {

DefaultTableModel dtm=(DefaultTableModel) userTable.getModel();

dtm.setRowCount(0);//设置成0行,清空表格

Connection conn=null;

try {

conn=dbUtil.getconn();

ResultSet rs=userDao.list(conn, user);

while(rs.next()) {

Vector<String> v=new Vector<String>();

v.add(rs.getString("id"));

v.add(rs.getString("username"));

v.add(rs.getString("password"));

dtm.addRow(v);

}

} catch (Exception e) {

// TODO: handle exception

e.printStackTrace();

}finally {

try {

dbUtil.clossconn(conn);

} catch (SQLException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

}

}

1. 查询按钮事件处理

点击查询按钮，若文本框为空，则显示user表中的所有数据，查询功能可以实现模糊搜索。如输入的信息在数据库中搜索不到，会显示空表。以下为查询按钮事件处理的代码

**private** **void** userSearchActionPerformed(ActionEvent evt) {

// **TODO** 自动生成的方法存根

String s\_userName=**this**.s\_userNameTxt.getText();

User user=**new** User();

user.setUsername(s\_userName);

**this**.fillTable(user);

}

6.修改模块设计

对用户，图书以及图书类型三个实体的修改功能设计以及修改按钮事件的处理基本类似，下面展示用户修改功能的设计。

1. 窗体设计



窗体使用WindowsBuilder插件设计，修改功能其中使用到的文本框为第二个输入用户名的文本框名为userNameTxt，输入密码的文本框名为passwordTxt用于修改。窗体中有设置一个JTable名为userTable显示数据库中的user表的内容，且会根据操作动态显示。当用户点击表格行后，表格中的数据会添加到下面的文本框中显示，用户输入完成后，点击修改按钮即可查看修改结果。

1. 数据库操作类（UserDao）实现的方法

//用户修改

**public** **int** update(Connection conn,User user)**throws** SQLException{

String sql="update user set username=?,password=? where id=?";

PreparedStatement pstmt=conn.prepareStatement(sql);

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

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

pstmt.setInt(3, user.getId());

**return** pstmt.executeUpdate();

}

//(用于修改中除了此id)是否有重复用户

**public** **boolean** existUserName2(Connection conn,User user)**throws** SQLException{

String sql="select \* from user where id!=? and username=?";

PreparedStatement pstmt=conn.prepareStatement(sql);

pstmt.setInt(1, user.getId());

pstmt.setString(2, user.getUsername());

ResultSet rs=pstmt.executeQuery();

**return** rs.next();

}

1. 表格行点击事件处理

设置一个鼠标点击事件，当用户点击表格行时，可以获取表格的行数和列数，并将对应的用户名和密码添加到下面的文本框中以下为表格行点击事件的处理代码

//表格行点击事件处理

**private** **void** userTableMousePressed(MouseEvent e) {

// **TODO** 自动生成的方法存根

//获取选中行

**int** row=userTable.getSelectedRow();

//获取列

userNameTxt.setText((String)userTable.getValueAt(row, 1));

passwordTxt.setText((String)userTable.getValueAt(row, 2));

}

1. 修改事件处理

当点击修改按钮时，会判断是否有点击表格行，即是否已经选择了需要修改的数据，然后会对文本框的内容进行判断，是否为空，输入的用户名和密码是否符合要求，且会判断修改后的用户名是否已经存在。对于修改成功以及错误的输入都有弹窗提示。修改成功后会刷新表格中的数据，显示修改后的结果并将输入文本框清空（清空的代码与重置的代码类似）。以下为修改事件处理的代码：

//用户修改事件处理

**private** **void** userUpdateActionPerformed(ActionEvent evt) {

// **TODO** 自动生成的方法存根

String userName=userNameTxt.getText();

String password=passwordTxt.getText();

String matches="^\\w{6,20}$";

**boolean** flag2=password.matches(matches);

**if**(ToolUtil.*isEmpty*(userName)) {

JOptionPane.*showMessageDialog*(**null**, "请选择要修改的记录");

**return**;

}

**int** row=userTable.getSelectedRow();

String id=(String) userTable.getValueAt(row, 0);

**if**(ToolUtil.*isEmpty*(userName)) {

JOptionPane.*showMessageDialog*(**null**, "用户名不能为空");

**return**;

}

**if**(userName.indexOf(" ")!=-1) {

JOptionPane.*showMessageDialog*(**null**, "用户名不能包含空格！");

**return**;

}

**if**(ToolUtil.*isEmpty*(password)) {

JOptionPane.*showMessageDialog*(**null**, "密码不能为空！");

**return**;

}

**if**(flag2==**false**) {

JOptionPane.*showMessageDialog*(**null**, "密码为6-20位且只能由字母，数字，下划线组成！");

**return**;

}

**if**(userName.length()>20) {

JOptionPane.*showMessageDialog*(**null**, "用户名必须小于20位");

**return**;

}

User user=**new** User(Integer.*parseInt*(id),userName,password);

Connection conn=**null**;

**try** {

conn=dbUtil.getconn();

**boolean** flag=userDao.existUserName2(conn, user);

**if**(flag) {

JOptionPane.*showMessageDialog*(**null**, "用户名已存在!");

**return**;

}

**int** modifyNum=userDao.update(conn,user);

**if**(modifyNum==1) {

JOptionPane.*showMessageDialog*(**null**, "修改成功!");

**this**.resetValue();

**this**.fillTable(**new** User());

}

**else** {

JOptionPane.*showMessageDialog*(**null**, "修改失败!");

}

} **catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

JOptionPane.*showMessageDialog*(**null**, "修改失败!");

}**finally** {

**try** {

dbUtil.clossconn(conn);

} **catch** (SQLException e) {

// **TODO** 自动生成的 catch 块

e.printStackTrace();

}

}

}

7.删除功能模块

对用户，图书以及图书类型三个实体的删除功能设计以及修改按钮事件的处理基本类似，下面展示用户删除功能的设计。

1. 窗体设计：与修改模块相同
2. 数据库操作类（UserDao）实现的方法

//用户删除

**public** **int** delete(Connection conn,String id)**throws** SQLException{

String sql="delete from user where id=?";

PreparedStatement pstmt=conn.prepareStatement(sql);

pstmt.setString(1, id);

**return** pstmt.executeUpdate();

}

1. 表格行点击事件处理：与修改模块相同
2. 删除按钮事件处理

当点击删除按钮时，会判断是否有点击表格行，即是否已经选择了需要删除的数据，然后会进行删除操作对于删除成功以及删除失败都有弹窗提示。删除成功后会刷新表格中的数据，显示删除后的结果并将输入文本框清空（清空的代码与重置的代码类似）。以下为删除事件处理的代码：

//用户删除事件处理

**private** **void** userDeleteActionPerformed(ActionEvent evt) {

// **TODO** 自动生成的方法存根

String userName=userNameTxt.getText();

System.***out***.println(userName);

**if**(ToolUtil.*isEmpty*(userName)) {

JOptionPane.*showMessageDialog*(**null**, "请选择要删除的记录");

**return**;

}

**int** row=userTable.getSelectedRow();

String id=(String) userTable.getValueAt(row, 0);

**int** n=JOptionPane.*showConfirmDialog*(**null**, "确定删除该记录吗？");

**if**(n==0) {

Connection conn=**null**;

**try** {

conn=dbUtil.getconn();

**int** deleteNum=userDao.delete(conn, id);

**if**(deleteNum==1) {

JOptionPane.*showMessageDialog*(**null**, "删除成功！");

**this**.resetValue();

**this**.fillTable(**new** User());

}

**else** {

JOptionPane.*showMessageDialog*(**null**, "删除失败！");

}

} **catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

JOptionPane.*showMessageDialog*(**null**, "删除失败！");

}**finally** {

**try** {

dbUtil.clossconn(conn);

} **catch** (SQLException e) {

// **TODO** 自动生成的 catch 块

e.printStackTrace();

}

}

}

}

# 总结

1.设计中遇到的问题以及解决方案

在设计的过程中我遇到密码可以输入汉字的问题，我需要限制密码只能由数字字母以及下划线组成，后来通过网络搜索，了解到可以通过正则表达式来限制密码的输入。然后通过查阅资料，学会了如何使用正则表达式进行限制。还有一个就是在最初的设计中，修改功能遇到问题，在修改用户名的时候，允许相同的用户名出现的问题。后来我通过在控制类UserDao中设置一个方法existUserName2(Connection conn,User user)：当点击选中行时，通过获取选中行的id以及修改后的用户名对数据库进行搜索，即除此行外若存在相同的用户名即会提示“用户名已存在”。

2.心得体会

通过本次的实训，我了解到设计图书管理系统的流程，实际上代表了一种从抽象到具体化的过程。本次的实训可以说是对本学期所学知识的一次实践总结，此次实验中广泛用到了比如JDBC、Java基础语言、GUI编程、数据库SQL语句以及数据库关系模式分析等知识。本次设计让我受益匪浅，学会了做事情要分主次，锻炼了解决问题的能力。更重要的是提高了自己的实践能力，以及更熟练的掌握本学期所学的知识。