****

**G06小组事务管理系统**

**程序清单**

****

小组成员：江泓 曾雨晴 于欣汝

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [ ] 正式发布  [ ] 正在修改 | 文件标识： | G06-project |
| 当前版本： | 1.1.0 |
| 作者： | 于欣汝 |
| 完成日期： | 2017-06-13 |

[1、功能模块 3](#_Toc9088)

[2、PDL描述 3](#_Toc16715)

[3、模块开发表 5](#_Toc26601)

[4、源代码 7](#_Toc7618)

[1、用户模块 7](#_Toc16908)

[2、增删改模块 11](#_Toc31456)

[3、排序 14](#_Toc14778)

[4、提示 15](#_Toc29151)

[5、 界面模块 23](#_Toc6470)

1、功能模块

用户注册

用户登录

事务增加

事务修改

事务删除（进回收站）

事务彻底删除

事务统计

事务状态更改

提示

界面（以登录为例）

# 2、PDL描述

（1）提醒和用户登录注册模块（负责人：江泓）

登录模块：

Procedure 登录 is

Begin

输入用户名和密码

通过SQL语句查找数据库中对应的用户名和密码

If（找到了）

then登录

Else

抛出异常（用户名不存在、密码错误）

End 登录

注册模块：

Procedure 注册 is

Begin

输入用户名和密码

检查用户名和密码是否符合格式

If（不符合）

Then 抛出异常（用户名应为XXX类型）

检查数据库中是否用相同用户名

If（有）

then抛出异常，已存在相同用户名

Else

对数据库进行增添

返回注册成功的提示

End 注册

（2）事务统计模块（负责人：于欣汝）

Begin 状态更改

对数据库查询的sql语句

通过preparedstatement传递参数

If<查询不到>

抛出异常

未能查询到

End if

对数据库的修改的sql语句

通过preparedstatement和execute传递参数

End

Begin 统计

对数据库查询的sql语句

通过preparedstatement传递参数

Loop while<查询结果不为空>

将查询到结果存入事务对象

将每个事物对象加入到list中

End loop

返回list类型的结果

End

（3）事务增删改模块（负责人：曾雨晴）

Begin 事务增加

输入时间，内容

If（时间不合理）

报错退出模块

If（事务内容过长）

报错退出模块

查询数据库

If（时间冲突）

提示用户

加入数据库

End

Begin 事务删除

选择事务

提示用户

If（true）

加入回收站中

End

Begin 回收站事务删除

选择事务

提示用户

If（true）

永久删除

End

Begin 事务修改

选择事务

输入时间，内容

If（时间不合理）

报错退出模块

If（事务内容过长）

报错退出模块

查询数据库

If（时间冲突）

提示用户

修改数据库内容

End

# 

# 3、模块开发表

|  |  |
| --- | --- |
| 模块名称 | 用户注册 |
| 方法名 | createUser |
| 编写人员 | 曾雨晴 |
| 完成日期 | 2017-05-30 |

|  |  |
| --- | --- |
| 模块名称 | 用户登录 |
| 方法名 | login |
| 编写人员 | 曾雨晴 |
| 完成日期 | 2017-05-30 |

|  |  |
| --- | --- |
| 模块名称 | 事务增加 |
| 方法名 | addEvent |
| 编写人员 | 于欣汝 |
| 完成日期 | 2017-05-30 |

|  |  |
| --- | --- |
| 模块名称 | 事务删除（进回收站） |
| 方法名 | intoRecycle |
| 编写人员 | 于欣汝 |
| 完成日期 | 2017-05-30 |

|  |  |
| --- | --- |
| 模块名称 | 事务彻底删除 |
| 方法名 | deleteEvent |
| 编写人员 | 于欣汝 |
| 完成日期 | 2017-05-30 |

|  |  |
| --- | --- |
| 模块名称 | 事务修改 |
| 方法名 | modifyEvent |
| 编写人员 | 于欣汝 |
| 完成日期 | 2017-05-30 |

|  |  |
| --- | --- |
| 模块名称 | 事务状态更改 |
| 方法名 | changeState |
| 编写人员 | 于欣汝 |
| 完成日期 | 2017-05-30 |

|  |  |
| --- | --- |
| 模块名称 | 事务统计 |
| 方法名 | listState |
| 编写人员 | 江泓 |
| 完成日期 | 2017-05-30 |

|  |  |
| --- | --- |
| 模块名称 | 提示 |
| 方法名 | remind |
| 编写人员 | 曾雨晴 |
| 完成日期 | 2017-05-30 |

|  |  |
| --- | --- |
| 模块名称 | 界面 |
| 方法名 | Frm\*\*\* |
| 编写人员 | 曾雨晴 |
| 完成日期 | 2017-06-15 |

# 4、源代码

## 1、用户模块

**public** **static** BeanUser *currentLoginUser*=**null**;

**public** **void** createUser(String userId,String pwd,String pwd1) **throws** Exception{

Connection conn=**null**;

**try**

{

conn=DBUtil.*getConnection*();

**if**(!pwd.equals(pwd1))

{

**throw** **new** BusinessException("两次输入密码不一致");

}

String sql="select userid from beanuser where userid=?";

java.sql.PreparedStatement pst=conn.prepareStatement(sql);

pst.setString(1, userId);

java.sql.ResultSet rs=pst.executeQuery();

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

{

**throw** **new** BusinessException("该用户已存在");

}

sql="insert into beanuser(userid,pwd) values(?,?)";

pst=conn.prepareStatement(sql);

pst.setString(1, userId);

pst.setString(2, pwd);

pst.execute();

pst.close();

rs.close();

}**catch**(SQLException e)

{

e.printStackTrace();

}**finally**

{

**if**(conn!=**null**)

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

//DriverManager.getConnection("jdbc:derby:H:\\db\\event;shutdown=true");

}

}

**public** **void** login(String userId,String pwd) **throws** Exception{

Connection conn=**null**;

**try**

{

conn=DBUtil.*getConnection*();

String sql="select \* from beanuser where userid=? and pwd=?";

java.sql.PreparedStatement pst=conn.prepareStatement(sql);

pst.setString(1, userId);

pst.setString(2, pwd);

java.sql.ResultSet rs=pst.executeQuery();

**if**(!rs.next())

{

**throw** **new** BusinessException("密码输入错误");

}

}**catch**(SQLException e)

{

e.printStackTrace();

}**finally**

{

**if**(conn!=**null**)

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

//DriverManager.getConnection("jdbc:derby:H:\\db\\event;shutdown=true");

}

}

**public** **void** modifyPwd(String userId,String pwd,String newpwd,String newpwd1) **throws** Exception{

Connection conn=**null**;

**if**(!newpwd.equals(newpwd1))

{

**throw** **new** BusinessException("两次输入密码不一致");

}

**try**

{

conn=DBUtil.*getConnection*();

String sql="select userid,pwd from beanuser where userid=? and pwd=?";

java.sql.PreparedStatement pst=conn.prepareStatement(sql);

pst.setString(1, userId);

pst.setString(2, pwd);

java.sql.ResultSet rs=pst.executeQuery();

**if**(!rs.next())

{

**throw** **new** BusinessException("密码输入错误");

}

sql="update beanuser set pwd=? where userid=?";

pst=conn.prepareStatement(sql);

pst.setString(1, userId);

pst.setString(2, newpwd);

pst.execute();

pst.close();

rs.close();

}**catch**(SQLException e)

{

e.printStackTrace();

}**finally**

{

**if**(conn!=**null**)

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

//DriverManager.getConnection("jdbc:derby:H:\\db\\event;shutdown=true");

}

}

**public** BeanUser loaduser (String userid) **throws** BaseException, SQLException

{

Connection conn=**null**;

**try**

{

conn=DBUtil.*getConnection*();

String sql="select userid,pwd from beanuser where userid=?";

java.sql.PreparedStatement pst=conn.prepareStatement(sql);

pst.setString(1,userid);

java.sql.ResultSet rs=pst.executeQuery();

**if**(!rs.next())

{

**throw** **new** BusinessException("登陆账号不存在");

}

BeanUser u=**new** BeanUser();

u.setUserid(rs.getString(1));

u.setPwd(rs.getString(2));

pst.close();

rs.close();

**return** u;

}**catch** (SQLException e) {

e.printStackTrace();

**throw** **new** DbException(e);

}**finally**

{

**if**(conn!=**null**)

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

//DriverManager.getConnection("jdbc:derby:H:\\db\\event;shutdown=true");

}

}

## 2、增删改模块

**public** **void** addEvent(String contect,String reminderTime,String state,String reminderMode,String classification )**throws** Exception{

Time t = **new** Time();

**if**(t.*timeCompare*(t.*getCurrentTime*(), reminderTime)>=0){

**throw** **new** Exception("时间设置不合理，该时间已过");

}

**if**(contect.length()>200 ){

**throw** **new** Exception("事务内容长度不能超过200个字");

}

Connection conn=**null**;

**try** {

conn=DBUtil.*getConnection*();

String sql="select count(\*) from beanevent where reminderTime = ? ";

java.sql.PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1,reminderTime );

java.sql.ResultSet rs = pst.executeQuery();

**if**(rs.getInt(1)>0){

**throw** **new** Exception("该时间已有"+rs.getInt(1)+"个事务");

}

}**catch**(Exception e){

e.printStackTrace();

conn=DBUtil.*getConnection*();

String sql="insert into Beanevent(contect,reminderTime,state,reminderMode,classification) values(?,?,?,?,?)";

java.sql.PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, contect);

pst.setString(2,reminderTime );

pst.setString(3, state);

pst.setString(4,reminderMode );

pst.setString(5, classification);

pst.execute();

pst.close();

}

}

**public** **void** deleteEvent(BeanEvent beanevent){

Connection conn=**null**;

**try**{

conn=DBUtil.*getConnection*();

String sql="DELETE FROM beanevent WHERE eventid = ?";

java.sql.PreparedStatement pst = conn.prepareStatement(sql);

pst.setInt(1, beanevent.getEventId());

pst.execute();

pst.close();

}**catch**(Exception e){

e.printStackTrace();

}

}

**public** **void** modifyEvent(BeanEvent beanevent,String contect,String reminderTime,String state,String reminderMode,String classification )**throws** Exception{

Time t = **new** Time();

**if**(t.*timeCompare*(t.*getCurrentTime*(), reminderTime)>=0){

**throw** **new** Exception("时间设置不合理，该时间已过");

}

**if**(contect.length()>200 ){

**throw** **new** Exception("事务内容长度不能超过200个字");

}

Connection conn=**null**;

**try** {

conn=DBUtil.*getConnection*();

String sql="select count(\*) from beanevent where reminderTime = ? ";

java.sql.PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1,reminderTime );

java.sql.ResultSet rs = pst.executeQuery();

**if**(rs.getInt(1)>0){

**throw** **new** Exception("该时间已有"+rs.getInt(1)+"个事务");

}

}**catch**(Exception e){

e.printStackTrace();

conn=DBUtil.*getConnection*();

String sql="update Beanevent set contect=?,reminderTime=?,state=?,reminderMode=?,classification = ? where eventid = ?";

java.sql.PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, contect);

pst.setString(2,reminderTime );

pst.setString(3, state);

pst.setString(4,reminderMode );

pst.setString(5, classification);

pst.setInt(6, beanevent.getEventId());

pst.execute();

pst.close();

}

}

**public** **void** intoRecycle(BeanEvent beanevent){

Connection conn=**null**;

**try**{

conn=DBUtil.*getConnection*();

String sql="UPDATE beanevent SET state = '回收站' WHERE eventid = ？";

java.sql.PreparedStatement pst = conn.prepareStatement(sql);

pst.setInt(1, beanevent.getEventId());

pst.execute();

pst.close();

}**catch**(Exception e){

e.printStackTrace();

}

}

**public** **void** changeState(String eventid,String state) {

Connection conn=**null**;

**try** {

conn=DBUtil.*getConnection*();

String sql="update BeanEvent set state=? where eventid=?";

java.sql.PreparedStatement pst=conn.prepareStatement(sql);

pst=conn.prepareStatement(sql);

pst.setString(1, eventid);

pst.setString(2, state);

pst.execute();

pst.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

## 3、排序

public List<BeanEvent> listState(String state,String classification){

List<BeanEvent> result=new ArrayList<BeanEvent>();

Connection conn=null;

try {

conn=DBUtil.getConnection();

String sql ="select eventId,contect,reminderTime,state,classification,userid from BeanEvent where userId=? and state = ? and classification = ? order by reminderTime ";

java.sql.PreparedStatement pst=conn.prepareStatement(sql);

pst.setString(1, UserManager.currentLoginUser.getUserid());

pst.setString(2, state);

pst.setString(3, classification);

java.sql.ResultSet rs=pst.executeQuery();

while(rs.next()){

BeanEvent r=new BeanEvent();

r.setEventId(rs.getInt(1));

r.setContect(rs.getString(2));

r.setReminderTime(rs.getString(3));

r.setState(rs.getString(4));

r.setClassification(rs.getString(5));

result.add(r);

}

} catch (SQLException e) {

e.printStackTrace();

}

return result;

}

4、提示

package event.ui;

import java.awt.Color;

import java.awt.Font;

import java.awt.Image;

import java.awt.MediaTracker;

import java.awt.Point;

import java.awt.Rectangle;

import java.awt.Shape;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.MouseMotionAdapter;

import java.awt.geom.Area;

import java.awt.image.BufferedImage;

import java.awt.image.PixelGrabber;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPasswordField;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import org.apache.derby.impl.tools.sysinfo.Main;

import com.sun.awt.AWTUtilities;

import event.control.EventManager;

import event.control.UserManager;

import event.model.BeanEvent;

import event.model.BeanUser;

import event.util.BusinessException;

import javafx.scene.shape.Circle;

/\*\*

\* 用户登录窗体

\* @author Admin

\*

\*/

public class FrmRemind extends JFrame {

private static final long serialVersionUID = 1L;

private Point origin; // 用于移动窗体

private BufferedImage img; // 用来设定窗体不规则样式的图片

private ImageIcon background;

//private JTextField numText = new JTextField(30);

//private JPasswordField passwordText = new JPasswordField(30);

private JLabel numLabel = new JLabel();

ImageIcon iconOk = new ImageIcon("bin/img/提醒-是.png");

ImageIcon iconNo = new ImageIcon("bin/img/提醒-否.png");

//ImageIcon iconEx = new ImageIcon("bin/img/×.png");

private JButton okbtn = new JButton(iconOk);

//private JButton register = new JButton(iconRe);

private JButton exit = new JButton(iconNo);

private String contect=null;

private String reminderTime=null;

private String state=null;

private String reminderMode=null;

private String classification=null;

private int numb=0;

private String rem=null;

/\*\*

\* 初始化窗体

\*/

public FrmRemind() {

super();

background = new ImageIcon("bin/img/remind.png");

JLabel back = new JLabel(background);

back.setBounds(0, 0, background.getIconWidth(),

background.getIconHeight());

/\*

\* 首先初始化一张图片，我们可以选择一张有透明部分的不规则图片

\* (要想图片能够显示透明，必须使用PNG格式的图片)

\*/

MediaTracker mt = new MediaTracker(this);

setTitle("事务提醒系统");

try {

img = ImageIO.read(new File("bin/img/remind.png"));

} catch (IOException e1) {

e1.printStackTrace();

}

mt.addImage(img, 0);

try {

mt.waitForAll(); // 开始加载由此媒体跟踪器跟踪的所有图像

} catch (InterruptedException e) {

e.printStackTrace();

}

try {

initialize(); // 窗体形状初始化

} catch (IOException e) {

e.printStackTrace();

}

addMenu();

this.add(back);

this.setVisible(true);

}

public FrmRemind(String contect,String reminderTime,String state,String reminderMode,String classification,int numb) {

super();

background = new ImageIcon("bin/img/remind.png");

JLabel back = new JLabel(background);

back.setBounds(0, 0, background.getIconWidth(),

background.getIconHeight());

/\*

\* 首先初始化一张图片，我们可以选择一张有透明部分的不规则图片

\* (要想图片能够显示透明，必须使用PNG格式的图片)

\*/

MediaTracker mt = new MediaTracker(this);

setTitle("事务提醒系统");

try {

img = ImageIO.read(new File("bin/img/remind.png"));

} catch (IOException e1) {

e1.printStackTrace();

}

mt.addImage(img, 0);

try {

mt.waitForAll(); // 开始加载由此媒体跟踪器跟踪的所有图像

} catch (InterruptedException e) {

e.printStackTrace();

}

try {

initialize(); // 窗体形状初始化

} catch (IOException e) {

e.printStackTrace();

}

addMenu();

this.add(back);

this.setVisible(true);

this.contect=contect;

this.reminderTime=reminderTime;

this.state=state;

this.reminderMode=reminderMode;

this.classification=classification;

this.numb=numb;

this.rem="该时段已经有"+numb+"个事务存在了，是否继续添加事务？";

//System.out.println(rem);

this.numLabel.setText(rem);

//System.out.println(numLabel);

}

/\*\*

\* 组件初始化

\*/

public void addMenu() {

this.setLayout(null);

//设置字体

Font font = new Font("", 0, 18);

/\*userLabel.setForeground(Color.black);

passwordLabel.setForeground(Color.BLACK);

userLabel.setFont(font);

passwordLabel.setFont(font);

passwordText.setEchoChar('\*');

\*/

numLabel.setForeground(Color.BLACK);

numLabel.setFont(font);

numLabel.setBounds(20,30,500,50);

okbtn.setBorderPainted(false);

okbtn.setBounds(100, 100, 67, 26);

exit.setBorderPainted(false);

exit.setBounds(294, 100, 67, 26);

this.add(okbtn);

this.add(exit);

this.add(numLabel);

okbtn.addMouseListener(new OwnListener());

exit.addMouseListener(new OwnListener());

//userText.setText("20160601");

//passwordText.setText("84878323");

}

/\*\*

\* 创建和图片形状一样的窗体

\* @throws IOException

\*/

private void initialize() throws IOException { // 窗体初始化

// 设定窗体大小和图片一样大

this.setSize(img.getWidth(null), img.getHeight(null));

// 设定禁用窗体装饰，这样就取消了默认的窗体结构

this.setUndecorated(true);

// 初始化用于移动窗体的原点

this.origin = new Point();

// 调用AWTUtilities的setWindowShape方法设定本窗体为制定的Shape形状

AWTUtilities.setWindowShape(this, getImageShape(img));

// 设定窗体可见度

AWTUtilities.setWindowOpacity(this, 0.8f);

this.setLocationRelativeTo(null);

this.setAlwaysOnTop(false);

// 由于取消了默认的窗体结构，所以我们要手动设置一下移动窗体的方法

this.addMouseListener(new OwnListener());

//监听鼠标移动事件

addMouseMotionListener(new MouseMotionAdapter() {

public void mouseDragged(MouseEvent e) {

Point p = getLocation();

setLocation(p.x + e.getX() - origin.x, p.y + e.getY()

- origin.y);

}

});

}

/\*\*

\* 将Image图像转换为Shape图形

\* @param img

\* @return

\*/

public Shape getImageShape(Image img) {

ArrayList<Integer> x = new ArrayList<Integer>();

ArrayList<Integer> y = new ArrayList<Integer>();

int width = img.getWidth(null);// 图像宽度

int height = img.getHeight(null);// 图像高度

// 筛选像素

// 首先获取图像所有的像素信息

PixelGrabber pgr = new PixelGrabber(img, 0, 0, -1, -1, true);

try {

pgr.grabPixels();

} catch (InterruptedException ex) {

ex.getStackTrace();

}

int pixels[] = (int[]) pgr.getPixels();

// 循环像素

for (int i = 0; i < pixels.length; i++) {

// 筛选，将不透明的像素的坐标加入到坐标ArrayList x和y中

int alpha = getAlpha(pixels[i]);

if (alpha == 0) {

continue;

} else {

x.add(i % width > 0 ? i % width - 1 : 0);

y.add(i % width == 0 ? (i == 0 ? 0 : i / width - 1) : i / width);

}

}

// 建立图像矩阵并初始化(0为透明,1为不透明)

int[][] matrix = new int[height][width];

for (int i = 0; i < height; i++) {

for (int j = 0; j < width; j++) {

matrix[i][j] = 0;

}

}

// 导入坐标ArrayList中的不透明坐标信息

for (int c = 0; c < x.size(); c++) {

matrix[y.get(c)][x.get(c)] = 1;

}

/\*

\* 由于Area类所表示区域可以进行合并，我们逐一水平"扫描"图像矩阵的每一行，

\* 将不透明的像素生成为Rectangle，再将每一行的Rectangle通过Area类的rec

\* 对象进行合并，最后形成一个完整的Shape图形

\*/

Area rec = new Area();

int temp = 0;

for (int i = 0; i < height; i++) {

for (int j = 0; j < width; j++) {

if (matrix[i][j] == 1) {

if (temp == 0)

temp = j;

else if (j == width) {

if (temp == 0) {

Rectangle rectemp = new Rectangle(j, i, 1, 1);

rec.add(new Area(rectemp));

} else {

Rectangle rectemp = new Rectangle(temp, i,

j - temp, 1);

rec.add(new Area(rectemp));

temp = 0;

}

}

} else {

if (temp != 0) {

Rectangle rectemp = new Rectangle(temp, i, j - temp, 1);

rec.add(new Area(rectemp));

temp = 0;

}

}

}

temp = 0;

}

return rec;

}

/\*\*

\* 取得透明度

\* @param pixel

\* @return

\*/

private int getAlpha(int pixel) {

return (pixel >> 24) & 0xff;

}

/\*\*

\* 事件监听

\* @author Admin

\*

\*/

private class OwnListener extends MouseAdapter {

public void mousePressed(MouseEvent e ) {

origin.x = e.getX();

origin.y = e.getY();

}

public void mouseClicked(MouseEvent e) {

//如果点击的区域位于右上角红色按钮，则关闭程序

if (e.getSource() == exit) {

setVisible(false);

} else if (e.getSource() == okbtn) {

setVisible(false);

EventManager em = new EventManager();

try {

em.addEvent(contect,reminderTime,state,reminderMode, classification);

} catch (Exception e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

}

}

public void mouseReleased(MouseEvent e) {

super.mouseReleased(e);

}

}

private class KeyOwnListener extends KeyAdapter {

public void keyPressed(KeyEvent e) {

if (10 == e.getKeyCode()) {

//验证用户是否合法，合法打开主程序

}

}

}

public static void main(String[] args) {

new FrmRemind();

}

}

## 界面模块

（由于代码过长，只摘取登录界面的源代码。）

package event.ui;

import java.awt.Color;

import java.awt.Font;

import java.awt.Image;

import java.awt.MediaTracker;

import java.awt.Point;

import java.awt.Rectangle;

import java.awt.Shape;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.MouseMotionAdapter;

import java.awt.geom.Area;

import java.awt.image.BufferedImage;

import java.awt.image.PixelGrabber;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPasswordField;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import com.sun.awt.AWTUtilities;

import event.control.UserManager;

import event.model.BeanEvent;

import event.model.BeanUser;

import event.util.BusinessException;

import javafx.scene.shape.Circle;

/\*\*

\* 用户登录窗体

\* @author Admin

\*

\*/

public class FrmLogin extends JFrame {

private static final long serialVersionUID = 1L;

private Point origin; // 用于移动窗体

private BufferedImage img; // 用来设定窗体不规则样式的图片

private ImageIcon background;

private JTextField userText = new JTextField(30);

private JPasswordField passwordText = new JPasswordField(30);

private JLabel userLabel = new JLabel("用户名");

private JLabel passwordLabel = new JLabel("密 码 ");

ImageIcon iconLo = new ImageIcon("bin/img/登录.png");

ImageIcon iconRe = new ImageIcon("bin/img/注册.png");

ImageIcon iconEx = new ImageIcon("bin/img/×.png");

private JButton okbtn = new JButton(iconLo);

private JButton register = new JButton(iconRe);

private JButton exit = new JButton(iconEx);

/\*\*

\* 初始化窗体

\*/

public FrmLogin() {

super();

background = new ImageIcon("bin/img/login.png");

JLabel back = new JLabel(background);

back.setBounds(0, 0, background.getIconWidth(),

background.getIconHeight());

/\*

\* 首先初始化一张图片，我们可以选择一张有透明部分的不规则图片

\* (要想图片能够显示透明，必须使用PNG格式的图片)

\*/

MediaTracker mt = new MediaTracker(this);

setTitle("事务提醒系统");

try {

img = ImageIO.read(new File("bin/img/login.png"));

} catch (IOException e1) {

e1.printStackTrace();

}

mt.addImage(img, 0);

try {

mt.waitForAll(); // 开始加载由此媒体跟踪器跟踪的所有图像

} catch (InterruptedException e) {

e.printStackTrace();

}

try {

initialize(); // 窗体形状初始化

} catch (IOException e) {

e.printStackTrace();

}

addMenu();

this.add(back);

this.setVisible(true);

}

/\*\*

\* 组件初始化

\*/

public void addMenu() {

this.setLayout(null);

//设置字体

Font font = new Font("", 0, 18);

userLabel.setForeground(Color.black);

passwordLabel.setForeground(Color.BLACK);

userLabel.setFont(font);

passwordLabel.setFont(font);

passwordText.setEchoChar('\*');

userLabel.setBounds(375, 140, 100, 50);

passwordLabel.setBounds(375, 220, 100, 50);

userText.setBounds(375, 180, 170, 30);

passwordText.setBounds(375, 260, 170, 30);

okbtn.setBorderPainted(false);

okbtn.setBounds(375, 320, 60, 25);

register.setBorderPainted(false);

register.setBounds(475, 320, 60, 25);

exit.setBorderPainted(false);

exit.setBounds(614, 0, 34, 33);

this.add(userLabel);

this.add(userText);

this.add(passwordLabel);

this.add(passwordText);

this.add(okbtn);

this.add(register);

this.add(exit);

okbtn.addMouseListener(new OwnListener());

register.addMouseListener(new OwnListener());

userText.addKeyListener(new KeyOwnListener());

passwordText.addKeyListener(new KeyOwnListener());

exit.addMouseListener(new OwnListener());

//userText.setText("20160601");

//passwordText.setText("84878323");

}

/\*\*

\* 创建和图片形状一样的窗体

\* @throws IOException

\*/

private void initialize() throws IOException { // 窗体初始化

// 设定窗体大小和图片一样大

this.setSize(img.getWidth(null), img.getHeight(null));

// 设定禁用窗体装饰，这样就取消了默认的窗体结构

this.setUndecorated(true);

// 初始化用于移动窗体的原点

this.origin = new Point();

// 调用AWTUtilities的setWindowShape方法设定本窗体为制定的Shape形状

AWTUtilities.setWindowShape(this, getImageShape(img));

// 设定窗体可见度

AWTUtilities.setWindowOpacity(this, 0.8f);

this.setLocationRelativeTo(null);

this.setAlwaysOnTop(false);

// 由于取消了默认的窗体结构，所以我们要手动设置一下移动窗体的方法

this.addMouseListener(new OwnListener());

//监听鼠标移动事件

addMouseMotionListener(new MouseMotionAdapter() {

public void mouseDragged(MouseEvent e) {

Point p = getLocation();

setLocation(p.x + e.getX() - origin.x, p.y + e.getY()

- origin.y);

}

});

}

/\*\*

\* 将Image图像转换为Shape图形

\* @param img

\* @return

\*/

public Shape getImageShape(Image img) {

ArrayList<Integer> x = new ArrayList<Integer>();

ArrayList<Integer> y = new ArrayList<Integer>();

int width = img.getWidth(null);// 图像宽度

int height = img.getHeight(null);// 图像高度

// 筛选像素

// 首先获取图像所有的像素信息

PixelGrabber pgr = new PixelGrabber(img, 0, 0, -1, -1, true);

try {

pgr.grabPixels();

} catch (InterruptedException ex) {

ex.getStackTrace();

}

int pixels[] = (int[]) pgr.getPixels();

// 循环像素

for (int i = 0; i < pixels.length; i++) {

// 筛选，将不透明的像素的坐标加入到坐标ArrayList x和y中

int alpha = getAlpha(pixels[i]);

if (alpha == 0) {

continue;

} else {

x.add(i % width > 0 ? i % width - 1 : 0);

y.add(i % width == 0 ? (i == 0 ? 0 : i / width - 1) : i / width);

}

}

// 建立图像矩阵并初始化(0为透明,1为不透明)

int[][] matrix = new int[height][width];

for (int i = 0; i < height; i++) {

for (int j = 0; j < width; j++) {

matrix[i][j] = 0;

}

}

// 导入坐标ArrayList中的不透明坐标信息

for (int c = 0; c < x.size(); c++) {

matrix[y.get(c)][x.get(c)] = 1;

}

/\*

\* 由于Area类所表示区域可以进行合并，我们逐一水平"扫描"图像矩阵的每一行，

\* 将不透明的像素生成为Rectangle，再将每一行的Rectangle通过Area类的rec

\* 对象进行合并，最后形成一个完整的Shape图形

\*/

Area rec = new Area();

int temp = 0;

for (int i = 0; i < height; i++) {

for (int j = 0; j < width; j++) {

if (matrix[i][j] == 1) {

if (temp == 0)

temp = j;

else if (j == width) {

if (temp == 0) {

Rectangle rectemp = new Rectangle(j, i, 1, 1);

rec.add(new Area(rectemp));

} else {

Rectangle rectemp = new Rectangle(temp, i,

j - temp, 1);

rec.add(new Area(rectemp));

temp = 0;

}

}

} else {

if (temp != 0) {

Rectangle rectemp = new Rectangle(temp, i, j - temp, 1);

rec.add(new Area(rectemp));

temp = 0;

}

}

}

temp = 0;

}

return rec;

}

/\*\*

\* 取得透明度

\* @param pixel

\* @return

\*/

private int getAlpha(int pixel) {

return (pixel >> 24) & 0xff;

}

/\*\*

\* 事件监听

\* @author Admin

\*

\*/

private class OwnListener extends MouseAdapter {

public void mousePressed(MouseEvent e) {

origin.x = e.getX();

origin.y = e.getY();

}

// 窗体上单击鼠标右键关闭程序

public void mouseClicked(MouseEvent e) {

//如果点击的区域位于右上角红色按钮，则关闭程序

if (e.getSource() == exit) {

System.exit(0);

} else if (e.getSource() == okbtn) {

//验证用户是否合法，合法打开主程序

String userid=userText.getText();

String pwd=new String (passwordText.getPassword());

try

{

UserManager um=new UserManager();

BeanUser u=new BeanUser();

um.login(userid, pwd);

u=um.loaduser(userid);

UserManager.currentLoginUser=u;

FrmMain dlg=new FrmMain();

dlg.setVisible(true);

setVisible(false);

}catch (BusinessException e1) {

JOptionPane.showMessageDialog(null, e1.getMessage(), "错误",JOptionPane.ERROR\_MESSAGE);

return;

} catch (Exception e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

} else if (e.getSource() == register) {

//打开注册页面

FrmRegister dlg=new FrmRegister();

dlg.setVisible(true);

dlg.setTitle("事务提醒系统");

setVisible(false);

}

}

public void mouseReleased(MouseEvent e) {

super.mouseReleased(e);

}

}

private class KeyOwnListener extends KeyAdapter {

public void keyPressed(KeyEvent e) {

if (10 == e.getKeyCode()) {

}

}

}

public static void main(String args[]) {

new FrmLogin();

}

}