第 20 章　腾宇超市管理系统

进入 21 世纪，随着经济的高速发展，各行各业的竞争进入了前所未有的激烈状态，竞争已不再是规模的竞争，还包括技术的竞争、管理的竞争、人才的竞争。超市的竞争也随之进入了一个全新的阶段。仓储店、便利店、特许加盟店、专卖店等都对超市产生了很大的冲击，为了提高物资管理的水平和工作效率，尽可能避免商品流通中各环节出现的问题，为超市开发一套管理系统是十分必要的。本章介绍的超市管理系统主要包括基本档案管理、采购订货管理、仓库入货管理、仓库出货管理、人员管理和部门管理等功能。

通过本章的学习，可以掌握以下要点：

- ☑ 超市管理系统的软件结构和业务流程
- ☑ 超市管理系统的数据库设计
- ☑ Java 程序连接数据库的方法
- ☑ 设计项目的基本流程

20.1　项目设计思路

20.1.1　功能阐述

超市管理系统是一款辅助超市管理员管理超市的实用性项目，根据超市的日常管理需要，超市管理系统应包括基本档案管理、采购订货管理、仓库入库管理、仓库出库管理、人员管理、部门管理 6 大功能。其中基本档案管理又分为供货商管理、销售商管理、货品档案管理、仓库管理，为管理员提供日常基本信息的功能；采购订货管理模块，用来对日常的采购订货信息进行管理；仓库入库管理，用来管理各种商品入库的信息；仓库出库管理，用来管理商品出库记录；人员管理，用来实现对超市内员工的管理；部门管理，用来实现对超市的各个独立部门进行管理。

20.1.2　系统预览

超市管理系统由多个窗体组成，其中包括系统不可缺少的登录窗体、系统的主窗体、功能模块的子窗体等。下面列出几个典型窗体，其他窗体请参见光盘中的源程序。

超市管理系统的登录窗体如图 20.1 所示。

当用户输入合法的用户名和密码后，单击“登录”按钮，即可进入系统的主窗体，运行结果如图 20.2 所示。

图 20.1　超市管理系统的登录窗体

图 20.2　超市管理系统的主窗体

本程序的主窗体中提供了进入各功能模块的按钮，通过单击这些按钮，可进入各子模块中。各个子功能模块还提供了查询、修改和添加相关信息的操作，例如，修改仓库入库窗体运行结果如图 20.3 所示。

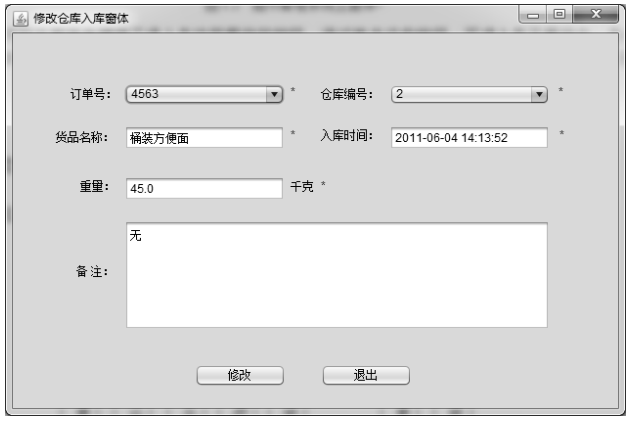

图 20.3　修改仓库入库信息

20.1.3　功能结构

超市管理系统是辅助超市管理员实现对超市的日常管理而设计的，本系统的功能结构如图 20.4 所示。

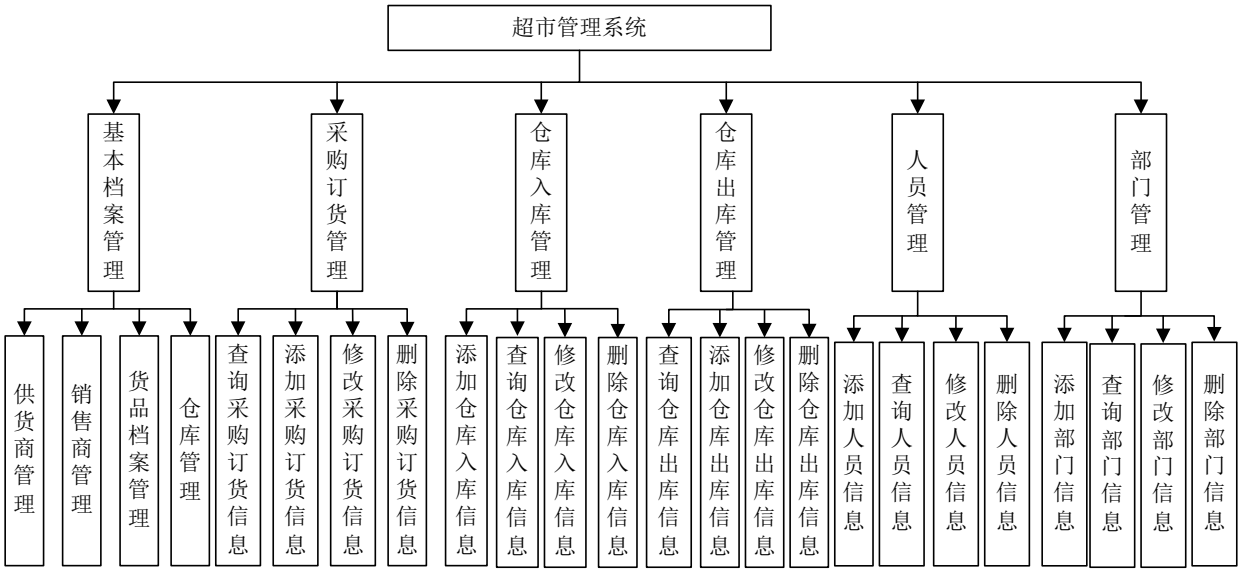

图 20.4　系统功能结构图

20.1.4　文件组织结构

超市管理系统中使用的根目录文件夹是 16，其中包括的文件架构如图 20.5 所示。

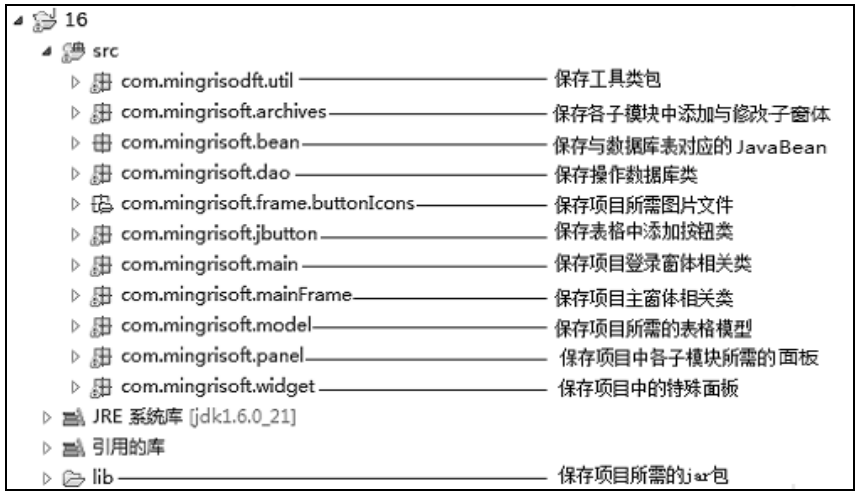

图 20.5　超市管理系统的文件架构图

20.2 数据库设计

20.2.1 数据库设计

超市管理系统采用的是 SQL Server 2014 数据库，数据库命名为 db_supermarket，包括的数据表有 tb_basicMessage、tb_contact、tb_depot 等，各数据表描述如图 20.6 所示。

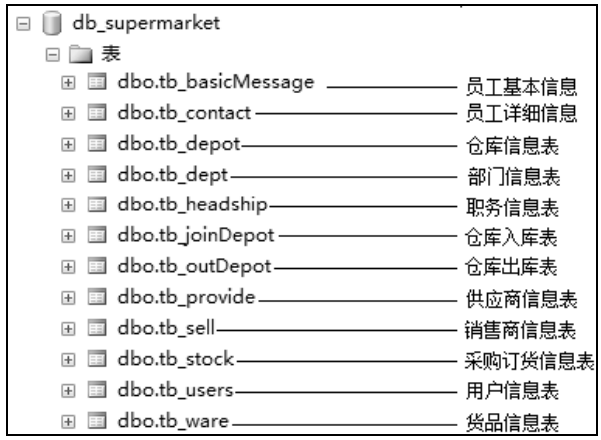

图 20.6 数据库结构

20.2.2 数据表设计

数据表设计是一个非常关键的环节，下面对系统中的数据表结构进行分析。由于篇幅有限，本章只给出了主要的数据表结构。其他数据表结构可参考资源包中的源程序。

1．员工基本信息表（tb_basicMessage）

员工基本信息表包括了员工姓名、年龄、性别、员工所在部门等信息，数据表字段设计如表 20.1 所示。

表 20.1 员工基本信息表设计（tb_basicMessage）

字　段	类　型	额　外	说　明
id	int	自动编号	主键
name	varchar(10)		员工姓名
age	int		员工年龄
sex	varchar(50)		员工性别
dept	int		员工部门，与部门表主键对应
headship	int		员工职务，与职务表主键对应

2．员工详细信息表（tb_contact）

员工详细信息表中保存了员工联系电话、办公电话、传真、邮箱地址、家庭地址等详细信息，数据表字段设计如表 20.2 所示。

表 20.2　员工详细信息表设计（tb_contact）

字　段	类　型	额　外	说　明
id	int	自动编号	主键
hid	int	外键	与员工基本信息表主键对应
contact	varchar(20)		联系电话
officePhone	varchar(30)		办公电话
fax	varchar(20)		传真
email	varchar(50)		邮箱地址
faddress	varchar(50)		家庭地址

3．仓库入库表（tb_ joinDepot）

仓库入库表保存仓库入库信息，其中包括订单编号、仓库编号、货品名称等，数据表字段设计如表 20.3 所示。

表 20.3　仓库入库表设计（tb_ joinDepot）

字　段	类　型	额　外	说　明
id	int	自动编号	主键
oid	vrchar(50)		订货编号
dId	int		仓库编号
wareName	varchar(40)		货品名称
joinTime	varchar(50)		入库时间
weight	float		货品重量
remark	varchar(200)		备注信息

4．用户信息表（tb_users）

用户信息表主要用于存储登录系统用户的用户名与密码信息，数据表字段设计如表 20.4 所示。

表 20.4　用户信息表设计（tb_users）

字　段	类　型	额　外	说　明
id	int	自动编号	主键
userName	varchar(20)		登录系统用户名
passWord	varchar(20)		登录系统密码

5．供应商信息表（tb_provide）

供应商信息表用于保存供应商相关信息，数据表字段设计如表 20.5 所示。

表 20.5　供应商信息表设计（tb_provide）

字　段	类　型	额　外	说　明
id	int	自动编号	主键

续表

字　段	类　型	额　外	说　明
cName	varchar(20)		供应商姓名
address	varchar(40)		供应商地址
linkman	varchar(50)		联系人
linkPhone	varchar(20)		联系电话
faxes	varchar(20)		传真
postNum	varchar(10)		邮箱地址
bankNum	varchar(30)		银行账号
netAddress	varchar(30)		主页
emaillAddress	varchar(50)		邮箱地址
remark	varchar(200)		备注信息

20.3　公共类设计

20.3.1　连接数据库

任何系统的设计都离不开数据库，每一步数据库操作都需要与数据库建立连接，为了增加代码的重用性，可以将连接数据库的相关代码保存在一个类中，以便随时调用。创建类 GetConnection，在该类的构造方法中加载数据库驱动，具体代码如下：

```
private Connection con;                                                    //定义数据库连接类对象
private PreparedStatement pstm;
private String user="sa";                                                  //连接数据库用户名
private String password="";                                                //连接数据库密码
private String className="com.microsoft.sqlserver.jdbc.SQLServerDriver";
//数据库驱动
private String url="jdbc:sqlserver://localhost:1433;DatabaseName=db_supermarket";
//连接数据库的 URL
public GetConnection(){
    try{
        Class.forName(className);
    }catch(ClassNotFoundException e){
        System.out.println("加载数据库驱动失败！");
        e.printStackTrace();
    }
}
```

在该类中定义获取数据库连接方法 getCon()，该方法返回值为 Connection 对象，具体代码如下：

```
public Connection getCon(){
    try {
```

```
            con=DriverManager.getConnection(url,user,password);          //获取数据库连接
        } catch (SQLException e) {
            System.out.println("创建数据库连接失败！");
            con=null;
            e.printStackTrace();
        }
        return con;                                                      //返回数据库连接对象
}
```

20.3.2　获取当前系统时间类

本系统中多处使用到了应用系统时间的模块，因此可以将获取当前系统时间类作为公共类设计。创建类 GetDate，在该类中定义获取时间方法 getDateTime()，具体代码如下：

```
public static String getDateTime(){                                     //该方法返回值为 String 类型
    SimpleDateFormat format;
    //SimpleDateFormat 类可以选择任何用户定义的日期-时间格式的模式
    Date date = null;
    Calendar myDate = Calendar.getInstance();
    //Calendar 的方法 getInstance()，以获得此类型的一个通用的对象
    myDate.setTime(new java.util.Date());
    //使用给定的 Date 设置此 Calendar 的时间
    date = myDate.getTime();
    //返回一个表示此 Calendar 时间值（从历元至现在的毫秒偏移量）的 Date 对象
    format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    //编写格式化时间为“年-月-日 时:分:秒”
    String strRtn = format.format(date);
    //将给定的 Date 格式化为日期/时间字符串，并将结果赋值给给定的 String
    return strRtn;                                                       //返回保存返回值变量
}
```

20.4　登录模块设计

20.4.1　登录模块概述

运行程序，首先进入系统的登录窗体。为了使窗体中的各个组件摆放得更加美观，笔者采用了绝对布局方式，并在窗体中添加了时钟面板来显示时间。运行结果请读者参照 20.1.2 小节中的图 20.1。

20.4.2 实现带背景的窗体

在创建窗体时，需要向窗体中添加面板，之后在面板中添加各种组件。Swing 中代表面板组件的类为 JPanel，该类是以灰色为背景，并且没有任何图片，这样就不能达到很好的美观效果。要实现在窗体中添加背景，就要通过重写 JPanel 面板来实现。

本项目中通过自定义 JPanel 组件来实现，并重写了面板绘制方法，面板绘制方法的声明如下：

```
protected void paintComponent(Graphics graphics)
```

其中，参数 graphics 是指控件中的绘图对象。

例如，本系统中创建的自定义面板 BackgroundPanel，该类继承 JPanel 类，在该类中定义表示背景图片的 Image 对象，重写 paintComponent 方法，实现绘制背景，具体代码如下：

```
public class BackgroundPanel extends JPanel {
    private Image image;                                    //背景图片
    public BackgroundPanel() {
        setOpaque(false);
        setLayout(null);                                    //使用绝对定位布局控件
    }
    /**
     * 设置背景图片对象的方法
     *
     * @param image
     */
    public void setImage(Image image) {
        this.image = image;
    }
    /**
     * 画出背景
     */
    protected void paintComponent(Graphics g) {
        if (image != null) {                                //如果图片已经初始化
            g.drawImage(image, 0, 0, this);                 //画出图片
        }
        super.paintComponent(g);
    }
}
```

20.4.3 登录模块实现过程

登录窗体设计十分简单，由一个“用户名”文本框和一个“密码”文本框组成，为了窗体的美观，笔者还添加了一个显示时钟的面板，该窗体设计如图 20.7 所示。

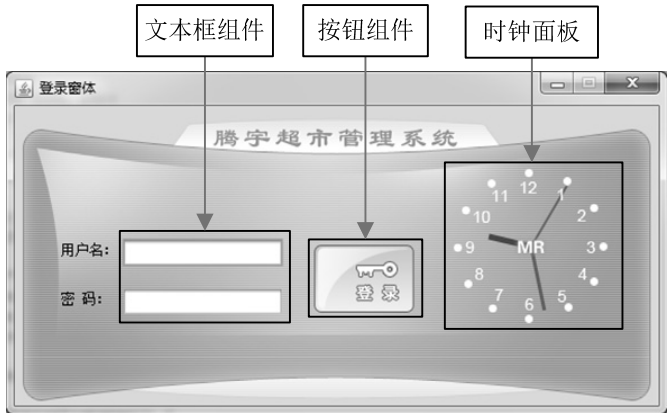

图 20.7　登录窗体设计效果

下面为大家详细地介绍登录模块的实现过程。

（1）实现用户登录操作的数据表是 tb_users，首先创建与数据表对应的 JavaBean 类 User，该类中属性与数据表中字段一一对应，并包含了属性的 setXXX()与 getXXX()方法，具体代码如下：

```
public class User {
    private int id;                          //定义映射主键的属性
    private String userName;                 //定义映射用户名的属性
    private String passWord;                 //定义映射密码的属性
    public int getId() {                     //id 属性的 getXXX()方法
        return id;
    }
    public void setId(int id) {              //id 属性的 setXXX()方法
        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）由于本系统的主窗体中显示了当前登录系统的用户名，而当前登录的用户对象是在登录窗体中查询出来的，为了实现两个窗体间的通信，可以创建保存用户会话的 Session 类，该类中包含有 User 对象的属性，并含有该属性的 setXXX()与 getXXX()方法，代码如下：

```
public class Session {
    private static User user;                          //User 对象属性
```

```
    public static User getUser() {                          //属性的 getXXX()方法
        return user;
    }
    public static void setUser(User user) {                 //属性的 setXXX()方法
        Session.user = user;
    }
}
```

（3）定义类 UserDao，在该类中实现按用户名与密码查询用户方法 getUser()，该方法的返回值为 User 对象，具体代码如下：

```
GetConnection connection = new GetConnection();
Connection conn = null;
//编写按用户名和密码查询用户的方法
public User getUser(String userName,String passWord){
    User user = new User();                                 //创建 JavaBean 对象
    conn = connection.getCon();                             //获取数据库连接
    try {
        String sql = "select * from tb_users where userName = ? and passWord = ?";
        //定义查询预处理语句
        PreparedStatement statement = conn.prepareStatement(sql);
        //实例化 PreparedStatement 对象
        statement.setString(1, userName);                   //设置预处理语句参数
        statement.setString(2, passWord);
        ResultSet rest = statement.executeQuery();          //执行预处理语句
        while(rest.next()){
            user.setId(rest.getInt(1));                     //应用查询结果设置对象属性
            user.setUserName(rest.getString(2));
            user.setPassWord(rest.getString(3));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return user;                                            //返回查询结果
}
```

（4）在“登录”按钮的单击事件中，调用判断用户是否合法方法 getUser()，实现如果用户输入的用户名与密码合法将转发至系统主窗体，如果用户输入了错误的用户名与密码，则给出相应的提示，具体代码如下：

```
enterButton.addActionListener(new ActionListener() {            //按钮的单击事件
    public void actionPerformed(ActionEvent e) {
        UserDao userDao = new UserDao();                        //创建保存有操作数据库类对象
        //以用户添加的用户名与密码为参数调用查询用户方法
        User user
        = userDao.getUser(userNameTextField.getText(),passwordField.getText());
        if(user.getId()>0){                                     //判断用户编号是否大于 0
            Session.setUser(user);                              //设置 Session 对象的 User 属性值
            RemoveButtomFrame frame = new RemoveButtomFrame();                  //创建主窗体对象
            frame.setVisible(true);                             //显示主窗体
```

```
                Enter.this.dispose();                                    //销毁登录窗体
            }
            else{                                                        //如果用户输入的用户名与密码错误
                JOptionPane.showMessageDialog(getContentPane(), "用户名或密码错误");    //给出提示信息
                userNameTextField.setText("");                           //“用户名”文本框设置为空
                passwordField.setText("");                               //“密码”文本框设置为空
            }
        }
});
```

20.5　主窗体设计

20.5.1　主窗体概述

成功登录系统后，即可进入系统的主窗体。系统的主窗体中以移动面板的形式显示了各功能按钮，并在初始化状态中显示了基本档案管理模块的相关功能，并为用户提供了时钟和日历面板。主窗体运行结果如图 20.8 所示。

图 20.8　主窗体运行结果

20.5.2　平移面板控件

在主窗体中笔者添加了移动面板控件，移动面板在水平方向添加了多个控件，通过左右平移两个按钮可以调整显示内容。在窗体中添加平移面板不仅可以增加窗体的灵活性，还能够提升窗体的美观效果。实现平移面板关键在于控制滚动面板中滚动条的当前值，就需要获取滚动面板的滚动条与设置滚动条当前值的相关知识，下面分别进行介绍。

☑ 获取滚动面板的水平滚动条

滚动面板包含水平和垂直两个方向的滚动条，通过适当的方法可以获取它们，下面的方法可以获取控制视口的水平视图位置的水平滚动条。方法声明如下：

```
public JScrollBar getHorizontalScrollBar()
```

☑ 获取滚动条当前值

滚动条的控制对象就是当前值，这个值控制着滚动条滑块的位置和滚动面板视图的位置。可以通过 getValue()方法来获取这个值，方法声明如下：

```
public int getValue()
```

☑ 设置滚动条当前值

```
public void setValue(int value)
```

其中，参数 value 指滚动条新的当前值。

创建成功滚动面板后，将按钮添加到滚动面板即可，本系统实现滚动面板的类为 SmallScrollPanel，该类是一个面板类，在该类的构造方法中初始化面板滚动事件处理器，代码如下：

```
public SmallScrollPanel() {
    scrollMouseAdapter = new ScrollMouseAdapter();                    //初始化处理器
    //初始化程序用图
    icon1 = new ImageIcon(getClass().getResource("top01.png"));
    icon2 = new ImageIcon(getClass().getResource("top02.png"));
    setIcon(icon1);                                                   //设置用图
    setIconFill(BOTH_FILL);                                           //将图标拉伸适应界面大小
    initialize();                                                     //调用初始化方法
}
```

在 SmallScrollPanel 类的初始化方法中设置面板布局，并在窗体中添加左侧和右侧的微调按钮，具体代码如下：

```
private void initialize() {
    BorderLayout borderLayout = new BorderLayout();
    borderLayout.setHgap(0);
    this.setLayout(borderLayout);                                     //设置布局管理器
    this.setSize(new Dimension(300, 84));
    this.setOpaque(false);                                            //使控件透明
    //添加滚动面板到界面居中位置
    this.add(getAlphaScrollPanel(), BorderLayout.CENTER);
    //添加左侧微调按钮
    this.add(getLeftScrollButton(), BorderLayout.WEST);
    //添加右侧微调按钮
    this.add(getRightScrollButton(), BorderLayout.EAST);
}
```

在平移面板中左右侧的两个箭头形状平移按钮，为两个添加背景的按钮，将该按钮的边框去掉，

就可显示大家看到的效果。下面以左侧微调按钮为例，介绍微调按钮的实现代码：

```
private JButton getLeftScrollButton() {
    if (leftScrollButton == null) {
        leftScrollButton = new JButton();
        //创建按钮图标
        ImageIcon icon1 = new ImageIcon(getClass().getResource(
                "/com/mingrisoft/frame/buttonIcons/zuoyidongoff.png"));
        //创建按钮图标 2
        ImageIcon icon2 = new ImageIcon(getClass().getResource(
                "/com/mingrisoft/frame/buttonIcons/zuoyidongon.png"));
        leftScrollButton.setOpaque(false);                          //按钮透明
        //设置边框
        leftScrollButton.setBorder(createEmptyBorder(0, 10, 0, 0));
        //设置按钮图标
        leftScrollButton.setIcon(icon1);
        leftScrollButton.setPressedIcon(icon2);
        leftScrollButton.setRolloverIcon(icon2);
        //取消按钮内容填充
        leftScrollButton.setContentAreaFilled(false);
        //设置初始大小
        leftScrollButton.setPreferredSize(new Dimension(38, 0));
        //取消按钮焦点功能
        leftScrollButton.setFocusable(false);
        //添加滚动事件监听器
        leftScrollButton.addMouseListener(scrollMouseAdapter);
    }
    return leftScrollButton;
}
```

创建左右微调按钮的事件监听器，实现当用户单击左右微调按钮时，移动面板，具体代码如下：

```
private final class ScrollMouseAdapter extends MouseAdapter implements
        Serializable {
    private static final long serialVersionUID = 5589204752770150732L;
    JScrollBar scrollBar = getAlphaScrollPanel().getHorizontalScrollBar();
    //获取滚动面板的水平滚动条
    private boolean isPressed = true;                               //定义线程控制变量
    public void mousePressed(MouseEvent e) {
        Object source = e.getSource();                              //获取事件源
        isPressed = true;
        if (source == getLeftScrollButton()) {     //判断事件源是左侧按钮还是右侧按钮，并执行相应操作
            scrollMoved(-1);
        } else {
            scrollMoved(1);
        }
    }
    /**
     * 移动滚动条的方法
     * @param orientation
```

```
     *       移动方向 -1 是左或上移动，1 是右或下移动
     */
    private void scrollMoved(final int orientation) {
        new Thread() {                                          //开辟新的线程
            private int oldValue = scrollBar.getValue();        //保存原有滚动条的值

            public void run() {
                while (isPressed) {                             //循环移动面板
                    try {
                        Thread.sleep(10);
                    } catch (InterruptedException e1) {
                        e1.printStackTrace();
                    }
                    oldValue = scrollBar.getValue();            //获取滚动条当前值
                    EventQueue.invokeLater(new Runnable() {
                        public void run() {
                            scrollBar.setValue(oldValue + 3 * orientation);
                            //设置滚动条移动 3 个像素
                        }
                    });
                }
            }
        }.start();
    }
    public void mouseExited(java.awt.event.MouseEvent e) {
        isPressed = false;
    }

    @Override
    public void mouseReleased(MouseEvent e) {
        isPressed = false;
    }
}
```

平移面板 SmallScrollPanel 类的设计效果如图 20.9 所示。

图 20.9　平移面板设计效果

20.5.3　主窗体实现过程

主窗体由多个面板组成，除了前面介绍过的功能按钮面板、时钟面板、日历面板外，还包括功能区面板，与主窗体中的其他面板不同，功能区面板是随时更换的，当用户单击不同的功能按钮，系统通过显示不同的面板来实现窗体内容的随时更换，设计效果如图 20.10 所示。

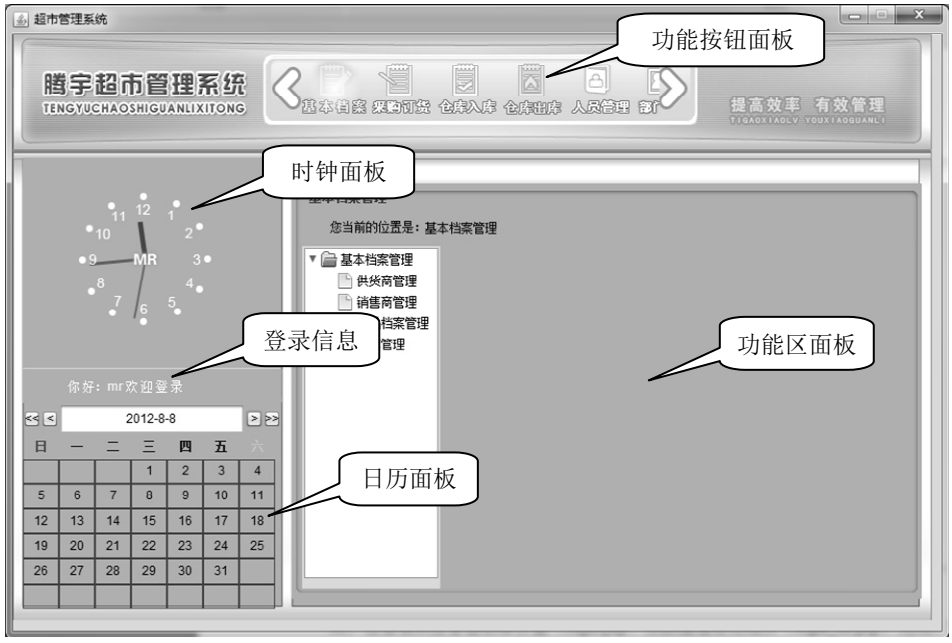

图 20.10　主窗体设计效果

下面介绍在主窗体的实现过程中几个重要的实现过程。

（1）通过如图 20.10 所示的主窗体设计效果可以看到，在主窗体中显示了当前登录的用户名，实现显示当前登录用户名代码如下：

```
User user = Session.getUser();                                    //获取登录用户对象
String info = "<html><body>" + "<font color=#FFFFFF>你 好：</font>"
            + "<font color=yellow><b>" + user.getUserName() + "</b></font>"
            + "<font color=#FFFFFF> 欢 迎 登 录</font>" + "</body></html>";
//定义窗体显示内容
clockpanel.add(getPanel());
JLabel label = new JLabel(info);                                  //定义显示指定内容的标签对象
```

（2）创建完成如图 20.9 所示的平移面板后，需要创建按钮组面板，再将按钮组面板添加到平移面板，才实现了主窗体中显示的效果，按钮组面板采用网格布局，设计效果如图 20.11 所示。

图 20.11　按钮组面板设计效果

按钮组面板实现代码如下：

```
public BGPanel getJPanel() {
    if (jPanel == null) {
        GridLayout gridLayout = new GridLayout();                 //定义网格布局管理器
        gridLayout.setRows(1);                                    //设置网格布局管理器的行数
```

```
            gridLayout.setHgap(0);                                    //设置组件间水平间距
            gridLayout.setVgap(0);                                    //设置组件间垂直间距
            jPanel = new BGPanel();
            jPanel.setLayout(gridLayout);                             //设置布局管理器
            jPanel.setPreferredSize(new Dimension(400, 50));          //设置初始大小
            jPanel.setOpaque(false);
            jPanel.add(getWorkSpaceButton(), null);                   //添加按钮
            jPanel.add(getProgressButton(), null);
            jPanel.add(getrukuButton(), null);
            jPanel.add(getchukuButton(), null);
            jPanel.add(getPersonnelManagerButton(), null);
            jPanel.add(getDeptManagerButton(), null);
            if (buttonGroup == null) {
                  buttonGroup = new ButtonGroup();
            }
            // 把所有按钮添加到一个组控件中
            buttonGroup.add(getProgressButton());
            buttonGroup.add(getWorkSpaceButton());
            buttonGroup.add(getrukuButton());
            buttonGroup.add(getchukuButton());
            buttonGroup.add(getPersonnelManagerButton());
            buttonGroup.add(getDeptManagerButton());
      }
      return jPanel;
}
```

（3）本系统中将平移面板中的各个按钮都封装在单独的方法中，下面以“基本档案”按钮为例，介绍平移面板中的各按钮的实现代码：

```
private GlassButton getWorkSpaceButton() {
      if (workSpaceButton == null) {
            workSpaceButton = new GlassButton();
            workSpaceButton.setActionCommand("基本档案管理");    //设置按钮的动作命令
            workSpaceButton.setIcon(new ImageIcon(getClass().getResource(
                        "/com/mingrisoft/frame/buttonIcons/myWorkSpace.png")));
            //定义按钮的初始化背景
            ImageIcon icon = new ImageIcon(getClass().getResource(
                        "/com/mingrisoft/frame/buttonIcons/myWorkSpace2.png"));
            //创建图片对象
            workSpaceButton.setRolloverIcon(icon);                    //设置按钮的翻转图片
            workSpaceButton.setSelectedIcon(icon);                    //设置按钮被选中时显示图片
            workSpaceButton.setSelected(true);
            workSpaceButton.addActionListener(new toolsButtonActionAdapter());
            //按钮的监听器
      }
      return workSpaceButton;
}
```

20.6　采购订货模块设计

20.6.1　采购订货模块概述

在超市的日常管理活动中，对于商品的采购和订货是不可缺少的。当用户单击平移面板中的“采购订货”按钮，即可进入采购订货模块，该模块中以表格的形式显示采购订货信息，在采购订货模块中还包括添加采购订货信息、修改采购订货信息、删除采购订货信息功能，运行效果如图 20.12 所示。

图 20.12　采购订货模块运行效果

20.6.2　在表格中添加按钮

表格用于显示复合数据，其中可以指定表格的表头和表文，默认的表格控件完全是以文本方式显示目标数据，要实现在表格中添加按钮或其他组件就要通过设置自定义的渲染器来实现，表格的渲染器通过 TableCellRenderer 接口实现，该接口中定义了 getTableCellRendererComponent()方法，这个方法将被表格控件回调来渲染指定的单元格控件。重写这个方法并在方法体中控制单元格的渲染，就可以把按钮作为表格的单元格控件。该方法的声明如下：

```
Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected,boolean hasFocus,
int row, int column)
```

方法中的参数说明如表 20.6 所示。

表 20.6　getTableCellRendererComponent()方法的参数说明

字　　段	类　　型
table	要求渲染器绘制的 JTable；可以为 NULL
value	要呈现的单元格的值。由具体的渲染器解释和绘制该值。例如，如果 value 是字符串 "TRUE"，则它可呈现为字符串，或者也可呈现为已选中的复选框。NULL 是有效值
isSelected	如果使用选中样式的高亮显示来呈现该单元格，则为 TRUE；否则为 FALSE
hasFocus	如果为 TRUE，则适当地呈现单元格。例如，在单元格上放入特殊的边框，如果可以编辑该单元格，则以彩色呈现它，用于表示正在进行编辑
row	要绘制的单元格的行索引。绘制表头时，row 值是-1
column	要绘制的单元格的列索引

例如，本模块中，设置“是否入库”列的渲染器，代码如下：

```
table.getColumn("是否入库").setCellRenderer(new ButtonRenderer());      //设置指定列的渲染器
```

20.6.3　添加采购订货信息实现过程

用户单击采购订货窗体中的“添加”按钮，即可弹出添加采购订货窗体，该窗体运行结果如图 20.13 所示。

图 20.13　添加采购订货窗体运行结果

下面详细地介绍添加采购订货窗体的实现过程。

（1）创建与采购订货信息表 tb_stock 对应的 JavaBean 对象 Stock，该类中的属性与 tb_stock 表中的字段一一对应，并包括了各属性的 setXXX()与 getXXX()方法，具体代码如下：

```
public class Stock {
    private int id;
    private String sName;
    private String orderId;
    private String consignmentDate;
    private String baleName;
    private String count;
    private float money;
    private String lairage;
```

```
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    …//省略了其他属性的 setXXX()与 getXXX()方法
}
```

（2）定义对采购订货信息表 tb_stock 中数据进行操作类 StockDao，其中添加采购订货信息方法 insertStock()，该方法以 Stock 为对象，具体代码如下：

```
public void insertStock(Stock stock) {
    conn = connection.getCon();                                    //获取数据库连接
    try {
        PreparedStatement statement = conn
                .prepareStatement("insert into tb_stock values(?,?,?,?,?,?)");
        //定义查询数据的 SQL 语句
        statement.setString(1,stock.getsName());                   //设置预处理语句参数
        statement.setString(2,stock.getOrderId());
        statement.setString(3,stock.getConsignmentDate());
        statement.setString(4,stock.getBaleName());
        statement.setString(5,stock.getCount());
        statement.setFloat(6,stock.getMoney());
        statement.executeUpdate();                                 //执行插入操作
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

（3）在添加采购订货窗体的“添加”按钮的单击事件中，实现判断用户填写的信息是否合法，再将这些信息保存到数据库中，具体代码如下：

```
JButton insertButton = new JButton("添加");
insertButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        StockDao dao = new StockDao();                             //定义操作数据表方法
        String oId = orderIdTextField.getText();                   //获取用户添加的订单号
        String wname = nameTextField.getText();                    //获取用户添加的客户名称
        String wDate = dateTextField.getText();                    //获取用户添加的交货日期
        String count = countTextField.getText();                   //获取用户添加的商品数量
        String bName = wNameTextField.getText();                   //获取用户添加的货品名称
        String money = moneyTextField.getText();                   //获取用户添加的货品金额
        int countIn = 0;
        float fmoney = 0;
        if((oId.equals(""))||(wname.equals("")) ||(wDate.equals("")) ||
              (count.equals("")) || (money.equals(""))){           //判断用户添加的信息是否完整
        JOptionPane.showMessageDialog(getContentPane(), "请将带星号的内容填写完整！",
                          "信息提示框", JOptionPane.INFORMATION_MESSAGE);
        //给出提示信息
```

```
                return;                                                    //退出程序
            }
            try{
                countIn = Integer.parseInt(count);                         //将用户添加的数量转换为整型
                fmoney = Float.parseFloat(money);
            }catch (Exception ee) {
                JOptionPane.showMessageDialog(getContentPane(), "要输入数字！",
                        "信息提示框", JOptionPane.INFORMATION_MESSAGE);
                return;
            }
            Stock stock = new Stock();                                     //定义与数据表对应的 JavaBean 对象
            stock.setsName(wname);                                         //设置对象属性
            stock.setBaleName(bName);
            stock.setConsignmentDate(wDate);
            stock.setCount(count);
            stock.setMoney(fmoney);
            stock.setOrderId(oId);
            dao.insertStock(stock);
            //调用数据库添加方法
            JOptionPane.showMessageDialog(getContentPane(), "数据添加成功！",
                    "信息提示框", JOptionPane.INFORMATION_MESSAGE);        //提示信息
        }
});
```

20.6.4 搜索采购订货信息实现过程

在采购订货模块中，添加了按指定条件搜索采购订货信息功能，用户可按照自己的需求指定查询条件。搜索采购订货窗体运行结果如图 20.14 所示。

图 20.14　搜索采购订货窗体

下面介绍搜索采购订货信息的具体实现过程。

（1）在搜索采购订货窗体中，为用户提供按“货品名称”“订单号”“交货时间”搜索指定采购订

货信息。下面以按货品名称查询采购订货信息为例，为大家介绍查询数据库方法，具体代码如下：

```
public List selectStockBySName(String sName) {
    List list = new ArrayList<Stock>();                          //定义保存查询结果的 List 对象
    conn = connection.getCon();                                  //获取数据库连接
    int id = 0;
    try {
        Statement statement = conn.createStatement();            //实例化 Statement 对象
        //定义查询语句，获取查询结果集
        ResultSet rest = statement.executeQuery("select * from tb_stock where sName ='"+sName+"'");
        while (rest.next()) {                                    //循环遍历查询结果集
            Stock stock = new Stock();                           //定义与数据表对象的 JavaBean 对象
            stock.setId(rest.getInt(1));                         //应用查询结果设置 JavaBean 属性
            stock.setsName(rest.getString(2));
            stock.setOrderId(rest.getString(3));
            stock.setConsignmentDate(rest.getString(4));
            stock.setBaleName(rest.getString(5));
            stock.setCount(rest.getString(6));
            stock.setMoney(rest.getFloat(7));
            list.add(stock);                                     //将 JavaBean 对象添加到集合
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;                                                 //返回查询集合
}
```

（2）当用户单击“搜索”按钮时，首先将表格中的数据全部删除，再将满足条件的数据填写到表格中，关键代码如下：

```
JButton findButton = new JButton("搜索");
findButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        dm.setRowCount(0);                                       //将表格内容清空
        String condition = comboBox.getSelectedItem().toString();
        //获取用户选择的查询条件
        String conditionText = conditionTextField.getText();     //获取用户添加的查询条件
        if(conditionText.equals("")){                            //如果用户没有添加查询条件
            JOptionPane.showMessageDialog(getParent(), "请输入查询条件！",
                    "信息提示框", JOptionPane.INFORMATION_MESSAGE); //给出提示信息
            return;                                              //退出程序
        }
        if(condition.equals("货品名称")){                         //如果用户选择按货品名称进行搜索
            List list = dao.selectStockBySName(conditionText);
            //调用按货品名称查询数据方法
            for(int i= 0;i<list.size();i++){                     //循环遍历查询结果
                Stock stock = (Stock)list.get(i);
                String oid = stock.getOrderId();                 //获取订单号信息
                int id = dao.selectJoinStockByOid(oid);          //根据订单号查询入库信息
                if(id <=0){                                      //如果该订单的货品在入库表中不存在
```

```
                    dm.addRow(new Object[]{"入库",stock.getId(),stock.getsName(),stock.getOrderId(),
                            stock. getConsignmentDate(),stock.getBaleName(),
                            stock.getMoney(),stock.getCount()});   //向表格中添加数据
                }
                else{                                              //如果指定订单号的货品名称在入库表中存在
                    dm.addRow(new Object[]{"已经入库",stock.getId(),stock.getsName(),stock.getOrderId(),
                            stock.getConsignmentDate(),stock.getBaleName(),
                            stock.getMoney(),stock.getCount()});
                }
            }
        }
```

20.6.5 修改采购订货信息实现过程

采购订货模块中提供了修改采购订货信息功能，当用户在显示采购订货信息的表格中选择要修改的信息后，单击窗体中的“修改”按钮，即可打开修改采购订单窗体，运行结果如图 20.15 所示。

图 20.15 修改采购订单窗体

下面详细地介绍修改采购订单窗体的实现过程。

（1）创建修改采购订货信息方法 updateStock()，该方法以 Stock 对象作为参数，具体代码如下：

```
public void updateStock(Stock stock) {
    conn = connection.getCon();                                    //获取数据库连接
    try {
        String sql = "update tb_stock set sName=?,orderId=?,consignmentDate=?," +
                "baleName=?,count=?,money=? where id =?";          //定义修改数据表方法
        PreparedStatement statement = conn.prepareStatement(sql);
                                                                   //获取 PreparedStatement 对象
        statement.setString(1, stock.getsName());                  //设置预处理语句参数值
        statement.setString(2, stock.getOrderId());
        statement.setString(3, stock.getConsignmentDate());
        statement.setString(4, stock.getBaleName());
        statement.setString(5, stock.getCount());
        statement.setFloat(6, stock.getMoney());
        statement.setInt(7, stock.getId());
        statement.executeUpdate();                                 //执行更新语句
    } catch (SQLException e) {
```

```
            e.printStackTrace();
        }
}
```

（2）要实现修改采购订货信息，首先将要修改的内容查询出来，并显示在窗体中。这样才能实现修改操作，首先编写按编号查询采购订货信息方法 selectStockByid()，具体代码如下：

```
public Stock selectStockByid(int id) {
        Stock stock = new Stock();                                  //定义与数据库对应的 JavaBean 对象
        conn = connection.getCon();                                 //获取数据库连接
        try {
            Statement statement = conn.createStatement();
            String sql = "select * from tb_stock where id = " + id;  //定义查询 SQL 语句
            ResultSet rest = statement.executeQuery(sql);           //执行查询语句获取查询结果集
            while (rest.next()) {                                   //循环遍历查询结果集
                stock.setId(id);                                    //应用查询结果设置对象属性
                stock.setsName(rest.getString(2));
                stock.setOrderId(rest.getString(3));
                stock.setConsignmentDate(rest.getString(4));
                stock.setBaleName(rest.getString(5));
                stock.setCount(rest.getString(6));
                stock.setMoney(rest.getFloat(7));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return stock;                                               //返回 Stock 对象
}
```

（3）由于显示采购订单窗体与修改采购订单窗体是两个独立的窗体，用户需要在显示采购订单窗体中选择要修改的信息，系统会将指定采购订货信息的编号写入文本文件中，之后在修改采购订单窗体中读取出来，这样就可实现在修改采购订单窗体中显示要修改的订货信息。在显示采购订单窗体中，将用户选择的采购订货信息保存在文本文件中，具体代码如下：

```
JButton updateButton = new JButton("修改");
updateButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                int row = table.getSelectedRow();                   //获取用户选中表格的行数
                if (row < 0) {
                    JOptionPane.showMessageDialog(getParent(), "没有选择要修改的数据！",
                            "信息提示框", JOptionPane.INFORMATION_MESSAGE);
                    return;
                } else {
                    File file = new File("filedd.txt");             //创建文件对象
                    try {
                        String column = dm.getValueAt(row, 1).toString();
                                                                    //获取表格中的数据
                        file.createNewFile();                       //新建文件
                        FileOutputStream out = new FileOutputStream(file);
```

```
                        out.write((Integer.parseInt(column)));          //将数据写入文件中
                        UpdateStockFrame frame = new UpdateStockFrame();
                                                                        //创建修改信息窗体
                        frame.setVisible(true);
                        out.close();                                    //将流关闭
                        repaint();
                    } catch (Exception ee) {
                        ee.printStackTrace();
                    }
                }
        }
});
```

（4）在修改采购订单窗体 UpdateStockFrame 中，读取用户写在文本文件中保存的要修改的采购订货信息的编号，再按照这个编号查询要修改的采购订货信息对象，将该对象的信息显示在窗体中，关键代码如下：

```
try {
        File file = new File("filedd.txt");                             //创建文件对象
        FileInputStream fin = new FileInputStream(file);                //创建文件输入流对象
        int count =   fin.read();                                       //读取文件中数据
        stock = dao.selectStockByid(count);                             //调用按编号查询数据方法
        file.delete();                                                  //删除文件
 } catch (Exception e) {
            e.printStackTrace();
}
JLabel orderIdLabel = new JLabel("订单号：");
orderIdLabel.setBounds(59, 55, 60, 15);
contentPane.add(orderIdLabel);
orderIdTextField = new JTextField();                                    //创建文本框对象
orderIdTextField.setText(stock.getOrderId());                           //设置文本框对象内容
orderIdTextField.setBounds(114, 50, 164, 25);
contentPane.add(orderIdTextField);                                      //将文本框对象添加到面板中
orderIdTextField.setColumns(10);
...//省略了设置窗体其他内容的代码
```

（5）在修改采购订单窗体的“修改”按钮中，调用修改采购订货信息方法，将用户修改的信息保存到数据库中，具体代码如下：

```
JButton insertButton = new JButton("修改");
insertButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            StockDao dao = new StockDao();                              //创建保存有修改方法的类对象
            String oId = orderIdTextField.getText();                    //获取用户填写订单数据
            String wname = nameTextField.getText();                     //获取用户填写的客户名信息
            String wDate = dateTextField.getText();                     //获取用户填写的交货日期信息
            String count = countTextField.getText();
            String bName = wNameTextField.getText();
            String money = moneyTextField.getText();
```

```
            int countIn = 0;
            float fmoney = 0;
            if((oId.equals(""))||(wname.equals("")) ||(wDate.equals("")) ||
                  (count.equals("")) || (money.equals(""))){                    //判断用户是否将信息添加完整
                  JOptionPane.showMessageDialog(getContentPane(), "请将带星号的内容填写完整！",
                              "信息提示框", JOptionPane.INFORMATION_MESSAGE);     //给出提示信息
                  return;
            }
            try{
                  countIn = Integer.parseInt(count);                            //将用户填写的数量转换为整数
                  fmoney = Float.parseFloat(money);
            }catch (Exception ee) {
                  JOptionPane.showMessageDialog(getContentPane(), "要输入数字！",
                              "信息提示框", JOptionPane.INFORMATION_MESSAGE);
                  //如果有异常抛出给出提示信息
                  return;
            }
            stock.setsName(wname);                                              //将设置采购订货信息属性
            stock.setBaleName(bName);
            stock.setConsignmentDate(wDate);
            stock.setCount(count);
            stock.setMoney(fmoney);
            stock.setOrderId(oId);
            dao.updateStock(stock);                                             //调用修改信息方法
            JOptionPane.showMessageDialog(getContentPane(), "数据添加成功！",
                        "信息提示框", JOptionPane.INFORMATION_MESSAGE);
      }
});
```

20.6.6　删除采购订货信息实现过程

如果要删除某采购订货信息，可以在采购订货信息表格中选中要删除的内容，再单击页面中的“删除”按钮，即可实现删除操作。实现删除功能的具体实现步骤如下。

（1）定义删除数据 deleteStock()方法，该方法有一个 int 类型参数，用于指定要删除采购订货信息的编号，具体代码如下：

```
public void deleteStock(int id){
      conn = connection.getCon();                                               //获取数据库连接
      String sql = "delete from tb_stock where id ="+id;                        //定义删除数据 SQL 语句
      try {
            Statement statement = conn.createStatement();                       //实例化 Statement 对象
            statement.executeUpdate(sql);                                       //执行 SQL 语句
      } catch (SQLException e) {
            e.printStackTrace();
      }
}
```

（2）在“删除”按钮的单击事件中，获取用户选择的表格中选择的要删除的采购订货信息的编号，