一套配教学视频的项目开发案例类丛书，累计销售160000册

10套
完整项目
开发案例

网站、QQ等技术支持，让学习畅通无阻

22小时
教学视频

模块库、题库、素材库登录本书网站

王国辉 宋禹蒙 编著

Java 项目开发全程实录 第3版

- 进销存管理系统（视频讲解：1小时44分）
- 企业内部通信系统（视频讲解：1小时42分）
- 企业人事管理系统（视频讲解：2小时55分）
- 酒店管理系统（视频讲解：2小时42分）
- 图书馆管理系统（视频讲解：2小时18分）
- 企业快信（视频讲解：1小时38分）
- 欣想电子商城（视频讲解：1小时58分）
- 超市管理系统（视频讲解：2小时10分）
- 企业门户网站（视频讲解：1小时8分）
- 棋牌游戏系统之网络五子棋（视频讲解：2小时10分）

清華大學出版社

丛书简介

《软件项目开发全程实录》丛书是一套以展现项目开发完整过程为主，同时配备完整教学视频和源程序的软件开发“案例类”图书。该丛书第1版自2008年出版以来，受到广大读者尤其是高校师生和求职人员的热烈欢迎，2011年进行了改版，截至目前，该丛书已经累计销售近16万册，成为近年来最畅销的编程案例类丛书之一。应读者朋友的要求，我们根据最新的市场变化并结合广大读者的建议，编写了该丛书的第3版。本书的特点有：

★ 10 套完整项目开发案例，项目开发全程展现

10 个项目开发案例涵盖了常用的不同类别的应用系统，每个案例都从需求分析到程序实现，基本完整地展现了项目开发全过程。

★ 22 小时教学视频，让学习更加轻松、快捷

几乎覆盖了本书全部内容，先看教学视频，再对照图书操作，学习一定更加轻松、更加快捷。

★ 提供模块库、题库、素材库等海量学习资料

本书服务网站 www.rjkflm.com，提供了模块库、题库、素材库等海量学习资料，读者可以索取、查阅相关资料。

★ 提供多种形式技术支持，让学习畅通无阻

为了帮助读者快速学习，本书提供以下形式技术支持：
QQ：4006751066
官方网站：www.rjkflm.com

本书项目案例及视频时间

进销存管理系统（视频讲解：1 小时 44 分）
企业内部通信系统（视频讲解：1 小时 42 分）
企业人事管理系统（视频讲解：2 小时 55 分）
酒店管理系统（视频讲解：2 小时 42 分）
图书馆管理系统（视频讲解：2 小时 18 分）
企业快信（视频讲解：1 小时 38 分）
欣想电子商城（视频讲解：1 小时 58 分）
超市管理系统（视频讲解：2 小时 10 分）
企业门户网站（视频讲解：1 小时 8 分）
棋牌游戏系统之网络五子棋（视频讲解：2 小时 10 分）

软件项目开发全程实录

Java 项目开发全程实录（第 3 版）

王国辉　宋禹蒙　编著

清華大學出版社
北 京

内 容 简 介

《Java 项目开发全程实录（第 3 版）》以进销存管理系统、企业内部通信系统、企业人事管理系统、酒店管理系统、图书馆管理系统、企业快信、欣想电子商城、超市管理系统、企业门户网站、棋牌游戏系统之网络五子棋等 10 个实际项目开发程序为案例，从软件工程的角度出发，按照项目的开发顺序，系统、全面地介绍了 J2SE 和 J2EE 项目的开发流程。从开发背景、需求分析、系统功能分析、数据库分析、数据库建模、网站开发和网站发布或者程序打包与运行，每一过程都进行了详细的介绍。

本书及光盘特色还有：10 套项目开发完整案例，项目开发案例的同步视频和其源程序。登录网站还可获取各类资源库（模块库、题库、素材库）等项目案例常用资源，网站还提供技术论坛支持等。

本书案例涉及行业广泛，实用性非常强。通过对本书的学习，读者可以了解各个行业的特点，能够针对某一行业进行软件开发，也可以通过光盘中提供的案例源代码和数据库进行二次开发，以减少开发系统所需要的时间。

图书在版编目（CIP）数据

Java 项目开发全程实录/王国辉，宋禹蒙编著．—3 版．—北京：清华大学出版社，2013
（软件项目开发全程实录）

ISBN 978-7-302-33741-6

I．①J…　II．①王…　②宋…　III．①JAVA 语言-程序设计　IV．①TP312

中国版本图书馆 CIP 数据核字（2013）第 204485 号

责任编辑：赵洛育
封面设计：陈　敏
版式设计：文森时代
责任校对：王　云
责任印制：
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：203mm×260mm　**印　　张：**28.75　**字　　数：**808 千字
（附 DVD 光盘 1 张）
版　　次：2008 年 6 月第 1 版　2013 年 10 月第 3 版　**印　　次：**2013 年 10 月第 1 次印刷
印　　数：23501～27500
定　　价：69.80 元

产品编号：051403-01

前言（第 3 版）

Preface 3rd Edition

编写目的与背景

众所周知，当前社会需求和高校课程设置严重脱节，一方面企业找不到可迅速上手的人才，另一方面大学生就业难。如果有一些面向工作应用的案例参考书，让大学生得以参考，并能亲手去做，势必能缓解这种矛盾。本书就是这样一本书：项目开发案例型的、面向工作应用的软件开发类图书。编写本书的首要目的就是架起让学生从学校走向社会的桥梁。

其次，本书以完成小型项目为目的，让学生切身感受到软件开发给工作带来实实在在的用处和方便，并非只是枯燥的语法和陌生的术语，从而激发学生学习软件的兴趣，让学生变被动学习为自主自发学习。

第三，本书的项目开发案例过程完整，不但适合在学习软件开发时作为小型项目开发的参考书，而且可以作为毕业设计的案例参考书。

第四，丛书第 1 版于 2008 年 6 月出版，于 2011 年 1 月改版，因为编写细腻，配备全程视频备受读者瞩目，丛书累计销售 16 万册，成为近年来最受欢迎的软件开发项目案例类丛书之一。

在以上背景下，我们根据读者朋友的反馈，与时俱进，对丛书进行了改版。

本书特点

视频讲解

对于初学者来说，视频讲解是最好的导师，它能够引导初学者快速入门，使初学者感受到编程的快乐和成就感，增强进一步学习的信心。鉴于此，本书为每一个案例都配备了视频讲解，初学者可以通过视频讲解实现案例中的功能。

典型案例

本书案例均从实际应用角度出发，应用了当前流行的技术，涉及的知识广泛，读者可以从每个案例中积累丰富的实战经验。

代码注释

为了便于读者阅读程序代码，书中的代码均提供了详细的注释，并且整齐地纵向排列，可使读者快速领略作者意图。

代码贴士

案例类书籍通常会包含大量的程序代码，冗长的代码往往令初学者望而生畏。为了方便读者阅读和理解代码，本书避免了连续大篇幅的代码，将其分割为多个部分，并对重要的变量、方法和知识点设计了独具特色的代码贴士。

知识扩展

为了增加读者的编程经验和技巧，书中每个案例都标记有注意、技巧等提示信息，并且在每章中都提供有一项专题技术。

本书约定

由于篇幅有限，本书每章并不能逐一介绍案例中的各模块。作者选择了基础和典型的模块进行介绍，对于功能重复的模块，由于技术、设计思路和实现过程基本雷同，因此没有在书中体现。读者在学习过程中若有相关疑问，请登录本书官方网站。本书中涉及的功能模块在光盘中都附带有视频录像，方便读者学习。

适合读者

本书适合作为计算机相关专业的大学生、软件开发相关求职者和爱好者的毕业设计和项目开发的参考书。

本书服务

为了给读者提供更为方便快捷的服务，读者可以登录本书官方网站：www.rjkflm.com，或者加入 QQ：4006751066 进行交流。

本书作者

本书由明日科技软件开发团队组织编写，主要由王国辉、宋禹蒙执笔，如下人员也参与了本书的编写工作，他们是：王小科、张鑫、杨丽、顾彦玲、赛奎春、高春艳、陈英、刘佳、辛洪郁、刘莉莉、王雨竹、隋光宇、郭鑫、刘志铭、李伟、张金辉、李慧、刘欣、李继业、潘凯华、赵永发、寇长梅、赵会东、王敬洁、李浩然、苗春义、刘清怀、张世辉、张领等，在此一并感谢！

在编写本书的过程中，我们本着科学、严谨的态度，力求精益求精，但错误、疏漏之处在所难免，敬请广大读者批评指正。

感谢您购买本书，希望本书能成为您的良师益友，成为您步入编程高手之路的踏脚石。

宝剑锋从磨砺出，梅花香自苦寒来。祝读书快乐！

编　　者

目　录

Contents

第 1 章

进销存管理系统

（Swing+SQL Server 2000 实现）

实现企业信息化管理是现代社会中小企业稳步发展的必要条件，它可以提高企业的管理水平和工作效率，最大限度地减少手工操作带来的失误。进销存管理系统正是一个信息化管理软件，可以实现企业的进货、销售、库存管理等各项业务的信息化管理。本章将介绍如何使用 Java Swing 技术和 SQL Server 2000 数据库开发跨平台的应用程序。

通过阅读本章，可以学习到：

- 如何进行项目的可行性分析
- 如何设计系统
- 如何进行数据库分析和数据库建模
- 企业进销存主要功能模块的开发过程
- 如何设计公共类
- 如何将程序打包

1.1 开 发 背 景

加入 WTO 之后，随着国内经济的高速发展，中小型的商品流通企业越来越多，其所经营的商品种类繁多，难以管理，而进销存管理系统逐渐成为企业经营和管理中的核心环节，也是企业取得效益的关键。×××有限公司是一家以商业经营为主的私有企业，为了完善管理制度，增强企业的竞争力，公司决定开发进销存管理系统，以实现商品管理的信息化。现需要委托其他单位开发一个企业进销存管理系统。

1.2 系 统 分 析

1.2.1 需求分析

通过与×××有限公司的沟通，要求系统具有以下功能：

- ☑ 操作简单，界面友好。
- ☑ 规范、完善的基础信息设置。
- ☑ 支持多人操作，要求有权限分配功能。
- ☑ 为了方便用户，要求系统支持多条件查询。
- ☑ 对销售信息提供销售排行。
- ☑ 支持销售退货和入库退货功能。
- ☑ 批量填写进货单及销售单。
- ☑ 支持库存价格调整功能。
- ☑ 当外界环境（停电、网络病毒）干扰本系统时，系统可以自动保护原始数据的安全。

1.2.2 可行性分析

根据《GB8567－88 计算机软件产品开发文件编制指南》中可行性分析的要求，制定的可行性研究报告如下。

1. 引言

☑ 编写目的

以文件的形式给企业的决策层提供项目实施的参考依据，其中包括项目存在的风险、项目需要的投资和能够收获的最大效益。

☑ 背景

×××有限公司是一家以商业经营为主的私有企业。为了完善管理制度、增强企业的竞争力、实现信息化管理，公司决定开发进销存管理系统。

2．可行性研究的前提

☑　要求

企业进销存管理系统必须提供商品信息、供应商信息和客户信息的基础设置；提供强大的多条件搜索功能和商品的进货、销售和库存管理功能；可以分不同权限、不同用户对该系统进行操作。另外，该系统还必须保证数据的安全性、完整性和准确性。

☑　目标

企业进销存管理系统的目标是实现企业的信息化管理，减少盲目采购，降低采购成本，合理控制库存，减少资金占用并提升企业市场竞争力。

☑　条件、假定和限制

为实现企业的信息化管理，必须对操作人员进行培训，而且将原有的库存、销售、入库等信息转换为信息化数据，需要操作员花费大量的时间和精力来完成。为了不影响企业的正常运行，进销存管理系统必须在两个月的时间内交付用户使用。

系统分析人员需要两天内到位，用户需要 5 天时间确认需求分析文档。去除其中可能出现的问题，例如用户可能临时有事，占用 6 天时间确认需求分析。那么程序开发人员需要在 1 个月零 15 天的时间内进行系统设计、程序编码、系统测试、程序调试和网站部署工作。其间，还包括了员工每周的休息时间。

☑　评价尺度

根据用户的要求，项目主要以企业进货、销售和查询统计功能为主，对于库存、销售和进货的记录信息应该及时、准确地保存，并提供相应的查询和统计。由于库存商品数量太多，不易盘点，传统的盘点方式容易出错，系统中的库存盘点功能要准确地计算出每种商品的损益数量，减少企业不必要的损失。

3．投资及效益分析

☑　支出

根据系统的规模及项目的开发周期（两个月），公司决定投入 7 个人。为此，公司将直接支付 9 万元的工资及各种福利待遇。在项目安装及调试阶段，用户培训、员工出差等费用支出需要 2 万元。在项目维护阶段预计需要投入 4 万元的资金。累计项目投入需要 15 万元资金。

☑　收益

用户提供项目资金 32 万元。对于项目运行后进行的改动，采取协商的原则根据改动规模额外提供资金。因此从投资与收益的效益比上，公司可以获得约 18 万元的利润。

项目完成后，会给公司提供资源储备，包括技术、经验的积累，其后再开发类似的项目时，可以极大地缩短项目开发周期。

4．结论

根据上面的分析，在技术上不会存在问题，因此项目延期的可能性很小。在效益上公司投入 7 个人、两个月的时间获利 18 万元，效益比较可观。在公司今后发展上，可以储备网站开发的经验和资源。因此认为该项目可以开发。

1.2.3 编写项目计划书

根据《GB8567－88 计算机软件产品开发文件编制指南》中的项目开发计划要求，结合单位实际情况，设计项目计划书如下。

1．引言

☑ 编写目的

为了保证项目开发人员按时、保质地完成预定目标，更好地了解项目实际情况，按照合理的顺序开展工作，现以书面的形式将项目开发生命周期中的项目任务范围、项目团队组织结构、团队成员的工作责任、团队内外沟通协作方式、开发进度、检查项目工作等内容描述出来，作为项目相关人员之间的共识和约定以及项目生命周期内的所有项目活动的行动基础。

☑ 背景

企业进销存管理系统是由×××有限公司委托我公司开发的大型管理系统，主要功能是实现企业进销存的信息化管理，包括统计查询、进货、销售、库存盘点及系统管理等功能。项目周期两个月。项目背景规划如表 1.1 所示。

表 1.1 项目背景规划

项 目 名 称	项目委托单位	任务提出者	项目承担部门
企业进销存管理系统	×××有限公司	陈经理	策划部门 研发部门 测试部门

2．概述

☑ 项目目标

项目目标应当符合 SMART 原则，把项目要完成的工作用清晰的语言描述出来。企业进销存管理系统的项目目标如下：

企业进销存管理系统的主要目的是实现企业进销存的信息化管理，主要的业务就是商品的采购、销售和入库，另外还需要提供统计查询功能，其中包括商品查询、供应商查询、客户查询、销售查询、入库查询和销售排行等。项目实施后，能够降低采购成本、合理控制库存、减少资金占用并提升企业市场竞争力，整个项目需要在两个月的时间内交付用户使用。

☑ 产品目标

时间就是金钱，效率就是生命。项目实施后，企业进销存管理系统能够为企业节省大量人力资源，减少管理费用，从而间接为企业节约成本，提高企业效益。

☑ 应交付成果

- 在项目开发完后，交付内容有企业进销存管理系统的源程序、系统的数据库文件和系统使用说明书。
- 将开发的进销存管理系统打包并安装到企业的网络计算机中。
- 企业进销存管理系统交付用户之后，进行系统无偿维护和服务 6 个月，超过 6 个月进行系统有偿维护与服务。

☑　项目开发环境

操作系统为 Windows 7、Windows XP 或 Windows 2003，使用集成开发工具 Eclipse，数据库采用 SQL Server 2000，项目运行环境为 JDK 7。

☑　项目验收方式与依据

项目验收分为内部验收和外部验收两种方式。在项目开发完成后，首先进行内部验收，由测试人员根据用户需求和项目目标进行验收。项目在通过内部验收后，交给客户进行验收，验收的主要依据为需求规格说明书。

3．项目团队组织

☑　组织结构

为了完成进销存管理系统的项目开发，公司组建了一个临时的项目团队，由公司副经理、项目经理、系统分析员、软件工程师、美工设计师和测试人员构成，如图 1.1 所示。

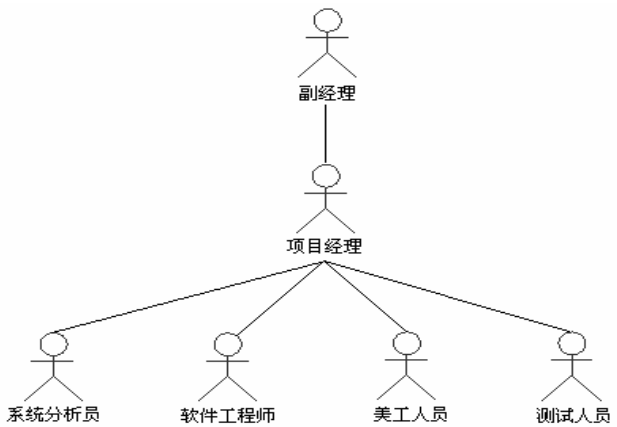


图 1.1　项目团队组织结构图

☑　人员分工

为了明确项目团队中每个人的任务分工，现制定人员分工，如表 1.2 所示。

表 1.2　人员分工

姓　名	技 术 水 平	所 属 部 门	角　色	工 作 描 述
陈××	MBA	经理部	副经理	负责项目的审批、决策的实施
侯××	MBA	项目开发部	项目经理	负责项目的前期分析、策划、项目开发进度的跟踪、项目质量的检查
钟××	高级系统分析员	项目开发部	系统分析员	负责系统功能分析、系统框架设计
李××	高级美术工程师	美工设计部	美术工程师	负责软件美术设计
梁××	高级软件工程师	项目开发部	系统分析员	负责软件设计与编码
马××	高级软件工程师	项目开发部	软件工程师	负责软件设计与编码
王××	中级软件工程师	软件评测部	测试人员	负责软件测试与评定

1.3　系 统 设 计

1.3.1　系统目标

根据需求分析的描述以及与用户的沟通，现制定系统实现目标如下：

☑　界面设计简洁友好、美观大方。

☑　操作简单、快捷方便。

☑　数据存储安全、可靠。

☑　信息分类清晰、准确。

- ☑ 强大的查询功能，保证数据查询的灵活性。
- ☑ 提供销售排行榜，为管理员提供真实的数据信息。
- ☑ 提供灵活、方便的权限设置功能，使整个系统的管理分工明确。
- ☑ 对用户输入的数据，系统进行严格的数据检验，尽可能排除人为的错误。

1.3.2 系统功能结构

本系统包括进货管理、基础信息管理、销售管理、库存管理、查询统计、系统管理 6 大部分。系统结构如图 1.2 所示。

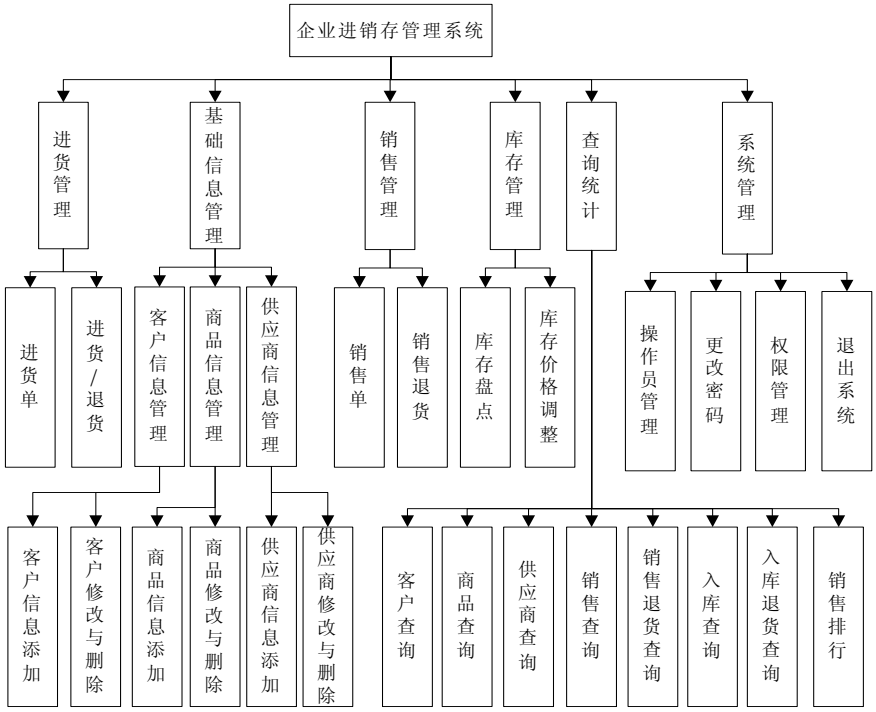


图 1.2 企业进销存管理系统功能结构

1.3.3 业务逻辑编码规则

遵守程序编码规则所开发的程序，代码清晰整洁、方便阅读，并且可以提高程序的可读性，要做到见其名知其意才能编写出优雅的程序代码。本节从数据库设计和程序编码两个方面介绍程序开发中的编码规则。

1．数据库对象命名规则

☑ 数据库命名规则

数据库命名以字母“db”开头（小写），后面加数据库相关英文单词或缩写。下面将举例说明，如表 1.3 所示。

表 1.3　数据库命名

数据库名称	描　　述
db_JXC	企业进销存管理系统数据库
db_library	图书馆管理系统数据库

在设计数据库时，为使数据库更容易理解，数据库命名时要注意大小写。

☑　数据表命名规则

数据表以字母“tb”开头（小写），后面加数据库相关英文单词或缩写和数据表名，多个单词间用“_”分隔。下面将举例说明，如表 1.4 所示。

表 1.4　数据表命名

数据表名称	描　　述
tb_sell_main	销售主表
tb_sell_detail	销售明细表

☑　字段命名规则

字段一律采用英文单词或词组（可利用翻译软件）命名，如找不到专业的英文单词或词组可以用相同意义的英文单词或词组代替。下面将举例说明，如表 1.5 所示。

表 1.5　字段命名

字 段 名 称	描　　述
ID	流水号
Name	名称
ProductInfo	商品信息

在命名数据表的字段时，应注意字母的大小写。

2．业务编码规则

☑　供应商编号

供应商的 ID 编号是进销存管理系统中供应商的唯一标识，不同的供应商可以通过该编号来区分。该编号是供应商信息表的主键。在本系统中对该编号的编码规则为：以字符串“gys”为编号前缀，加上 4 位数字作编号的后缀，这 4 位数字从 1000 开始。例如，gys1001。

☑　客户编号

和供应商编号类似，客户的 ID 编号也是客户的唯一标识，不同的客户将以该编号进行区分。该编号作为客户信息表的主键，有数据的、唯一性的约束条件，所以，在客户信息表中不可能有两个相同的客户编号。企业进销存管理系统对客户编号的编码规则为：以字符串“kh”为编号的前缀，加上 4 位数字作编号的后缀，这 4 位数字从 1000 开始。例如，kh1002。

☑　商品编号

商品编号是商品的唯一标识，它是商品信息表的主键，用于区分不同的商品。即使商品名称、单

价、规格等信息相同，其 ID 编号也是不可能相同的，因为主键约束不可以存在相同的 ID 值。商品编号的编码规则和客户编号、供应商编号的编码规则相同，但是前缀使用了“sp”字符串。例如，sp2045。

☑ 销售票号

销售票号用于区分不同的销售凭据。销售票号的命名规则为：以“XS”字符串为前缀，加上销售单的销售日期，再以 3 位数字作后缀。例如，XS20071205001。

☑ 进货票号

进货票号用于区分不同的商品入库信息。进货票号的命名规则为：以“RK”字符串为前缀，加上商品的入库日期，再以 3 位数字作后缀。例如，RK20071109003。

☑ 退货票号

退货票号用于区分不同的入库退货信息。入库退货票号的命名规则为：以“RT”字符串为前缀，加上商品入库的退货日期，再以 3 位数字作后缀。例如，RT20071109001。

1.3.4 系统流程图

进销存管理系统的系统流程如图 1.3 所示。

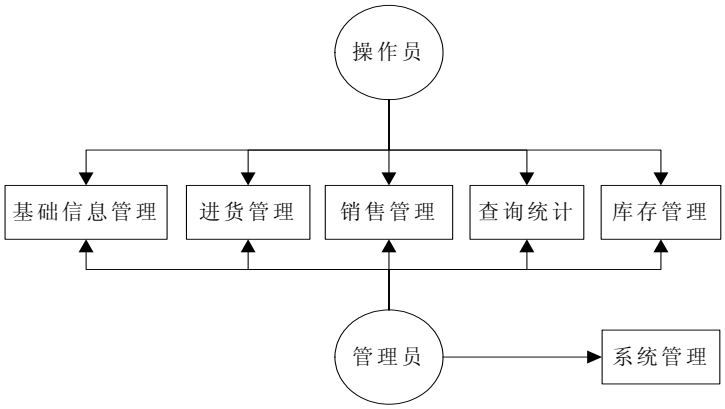


图 1.3 系统流程图

1.3.5 构建开发环境

在开发企业进销存管理系统时，使用了下面的软件环境。

☑ 操作系统：Windows 7。

☑ Java 开发包：JDK 7。

☑ 数据库：SQL Server 2000。

☑ 分辨率：最佳效果为 1024×768 像素。

1.3.6 系统预览

企业进销存管理系统由多个程序界面组成，下面仅列出几个典型界面的预览，其他界面参见光盘中的源程序。

进销存管理系统的主界面如图 1.4 所示，该界面是所有功能模块的父窗体，其中包含调用所有功能模块的导航面板。商品进货单界面如图 1.5 所示，该界面将进货单数据添加到数据库中，其中进货单的编号由系统自动生成。

操作员管理界面如图 1.6 所示，该界面由系统管理调用，主要用于操作员的添加、查看和删除。商品管理界面如图 1.7 所示，该界面包括商品的添加、修改和删除等功能。

由于路径太长，因此省略了部分路径，省略的路径是“TM\01\JXCManager\src”。

图 1.4　主窗体（光盘\…\com\lzw\JXCFrame.java）

图 1.5　进货单界面（光盘\…\internalFrame\JinHuoDan.java）

图 1.6　操作员管理界面（光盘\…\internalFrame\CzyGL.java）

图 1.7　商品管理界面（光盘\…\internalFrame\ShangPinGuanLi.java）

1.3.7　文件夹组织结构

在进行系统开发之前，需要规划文件夹组织结构，也就是说，建立多个文件夹，对各个功能模块进行划分，实现统一管理。这样做的好处为易于开发、管理和维护。本系统的文件夹组织结构如图 1.8 所示。

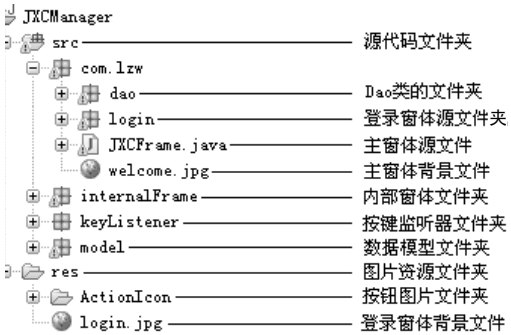


图 1.8　文件夹组织结构图

1.4 数据库设计

1.4.1 数据库分析

本系统是一个桌面应用程序，它可以直接在本地计算机运行，而不需要像 Web 应用那样部署到指定的服务器中，所以这个进销存管理系统在本地计算机上安装了 SQL Server 2000 数据服务器，将数据库和应用程序放在同一个计算机中，可以节省开销、提升系统安全性。另外，本系统也可以在网络内的其他计算机中运行，但是这需要将数据库对外开放，会降低数据安全性。其数据库运行环境如下。

☑ 硬件平台
- CPU：P4 3.2GHz。
- 内存：512MB 以上。
- 硬盘空间：80GB。

☑ 软件平台
- 操作系统：Windows 2003 以上。
- 数据库：SQL Server 2000。

1.4.2 进销存管理系统的 E-R 图

企业进销存管理系统主要实现从进货、库存到销售的一体化信息管理，涉及商品信息、商品的供应商、购买商品的客户等多个实体。下面简单介绍几个关键的实体 E-R 图。

☑ 客户实体 E-R 图

企业进销存管理系统将记录所有的客户信息，在执行销售、退货等操作时，将直接引用该客户的实体属性。客户实体包括客户编号、客户名称、简称、地址、电话、邮政编码、联系人、联系人电话、传真、开户行和账号等属性。客户实体 E-R 图如图 1.9 所示。

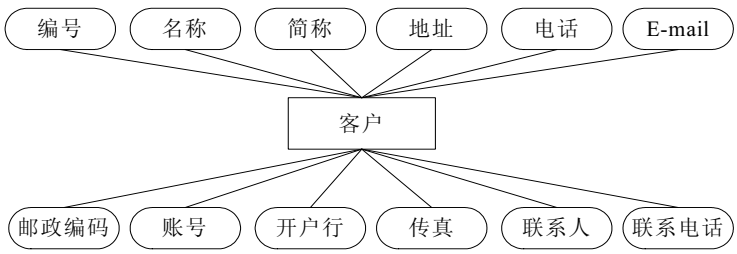


图 1.9 客户实体 E-R 图

☑ 供应商实体 E-R 图

不同的供应商可以为企业提供不同的商品，在商品信息中将引用商品供应商的实体属性。供应商实体包括编号、名称、简称、地址、电话、邮政编码、传真、联系人、联系电话、开户行和 E-mail 属性。供应商实体 E-R 图如图 1.10 所示。

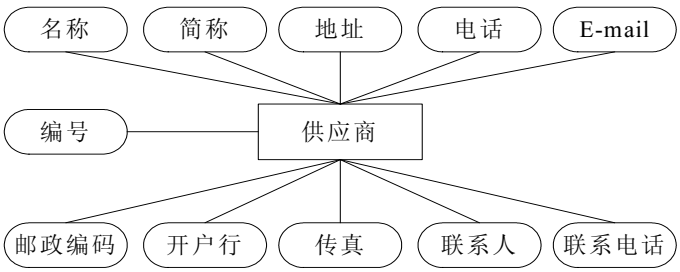


图 1.10　供应商实体 E-R 图

☑　商品实体 E-R 图

商品信息是进销存管理系统中的基本信息，系统将维护商品的进货、退货、销售、入库等操作。商品实体包括编号、商品名称、商品简称、产地、单位、规格、包装、批号、批准文号、商品简介和供应商属性。商品实体 E-R 图如图 1.11 所示。

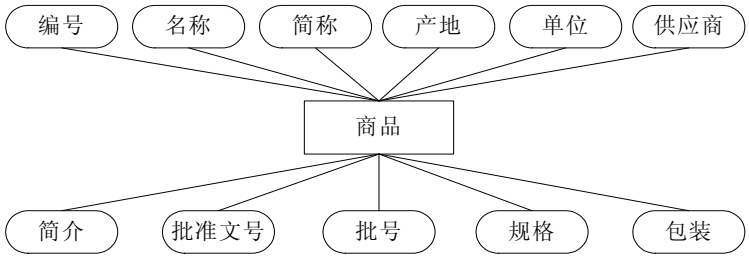


图 1.11　商品实体 E-R 图

1.4.3　使用 PowerDesigner 建模

在数据库概念设计中已经分析了本系统中主要的数据实体对象，通过这些实体可以得出数据表结构的基本模型，最终实施到数据库中，形成完整的数据结构。本系统将使用 PowerDesigner 工具完成数据库建模，使用的版本为 12.5。使用该工具建模的步骤如下：

（1）运行 PowerDesigner，并在 PowerDesigner 主窗口中选择主菜单中的 File/New 命令，在打开的 New 对话框左侧的 Model type 列表框中选择 Physical Data Model（物理数据模型，简称 PDM）选项，在右侧的 Model name 文本框中输入模型名称 JXCManager，在 DBMS 下拉列表框中选择数据库管理系统。PowerDesigner 支持的数据库管理系统非常多，例如，常用的 MySQL 5.0、Microsoft SQL Server 2005、Oracle Version 10gR2 等。企业进销存管理系统选择 Microsoft SQL Server 2000 作为数据库服务器，单击“确定”按钮，如图 1.12 所示。

（2）打开新建的 PDM 窗口。在该窗口的中心空白区域是模型编辑器，下方为输出窗口。另外还有一个浮动的工具面板，其中包括常用的建表工具、建视图工具和主外键引用工具，如图 1.13 所示。

（3）在图 1.13 中单击“建表工具”按钮，这时鼠标指针将显示为，在模型编辑器的合适位置单击，此时在图形窗口中将显示如图 1.14 所示的数据表模型。

说明　细心的读者可以发现，此时的鼠标指针仍然是。如果再次单击还将出现类似图 1.14 所示的表符号。如果想取消该指针，可以单击工具面板中的按钮或单击鼠标右键。

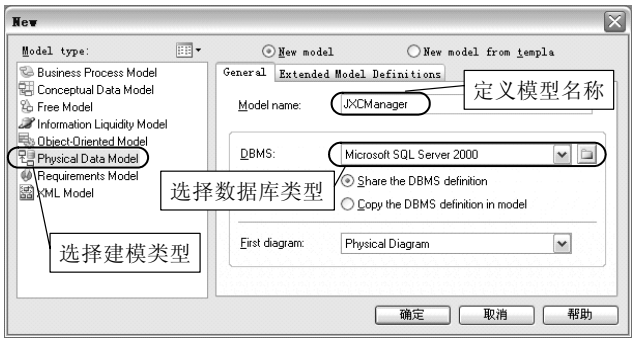


图 1.12　New 对话框

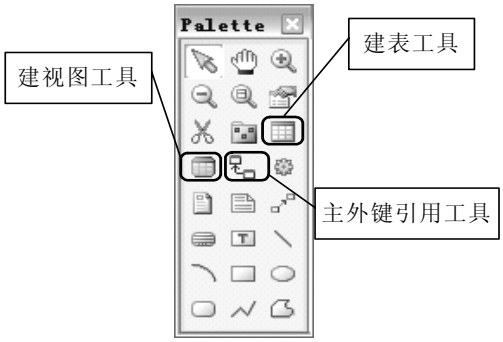


图 1.13　工具面板

（4）在图 1.14 所示的表符号上双击鼠标左键，将打开 Table Properties（表属性）对话框。默认情况下选择的是 General 选项卡，在该选项卡的 Name 文本框中输入表的名称 tb_manager，此时在 Code 文本框中也将自动显示 tb_manager，其他选择默认即可。

（5）选择 Columns 选项卡，首先单击输入列表的第一行，将自动转换第一行为编辑状态，然后在 Name 列输入字段名称为 ID，同时 Code 列也将自动显示为 ID，再在 Data Type 列中选择 int 选项，最后选中 P 列的复选框将该数据表字段设置为主键，此时 M 列的复选框也将自动被选中，它约束字段值不能为空。

（6）按照步骤（5）的方法再添加两个列：name 和 PWD，但是不需要选中 P 列复选框设置主键，如图 1.15 所示。

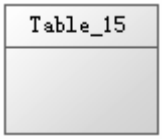


图 1.14　表符号

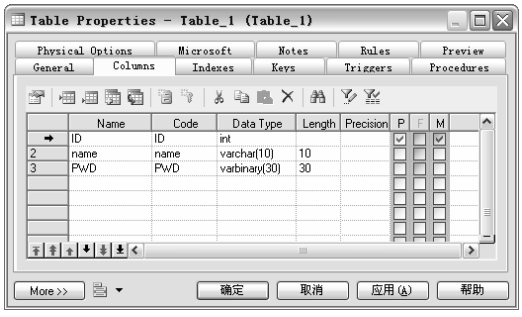


图 1.15　Columns（列）选项卡

（7）在图 1.15 中单击“应用”按钮后，选择 ID 字段，单击左上角的按钮，将打开 Column Properties（列属性）对话框。默认选择 General 选项卡，在其中选中 Identity 复选框，此项操作用于设置 ID 字段使用自动编号。

（8）单击“应用”按钮后，再单击“确定”按钮，关闭 Column Properties 对话框。

（9）单击“确定”按钮，关闭 Table Properties 对话框，完成 tb_manager 表的创建。

（10）按照步骤（3）～步骤（9）的方法创建本系统中的其他数据表，并通过主外键引用工具建立各表间的依赖关系。创建完成的模型如图 1.16 所示。

在默认情况下，创建后的表模型中的全部文字均为常规样式的宋体 8 号字，如果想修改文字的格式，可以选中全部表符号，按 Ctrl+T 组合键，在打开的 Symbol Format 对话框中选择 Font 选项卡，从中设置相关内容的字体及样式和字号等。

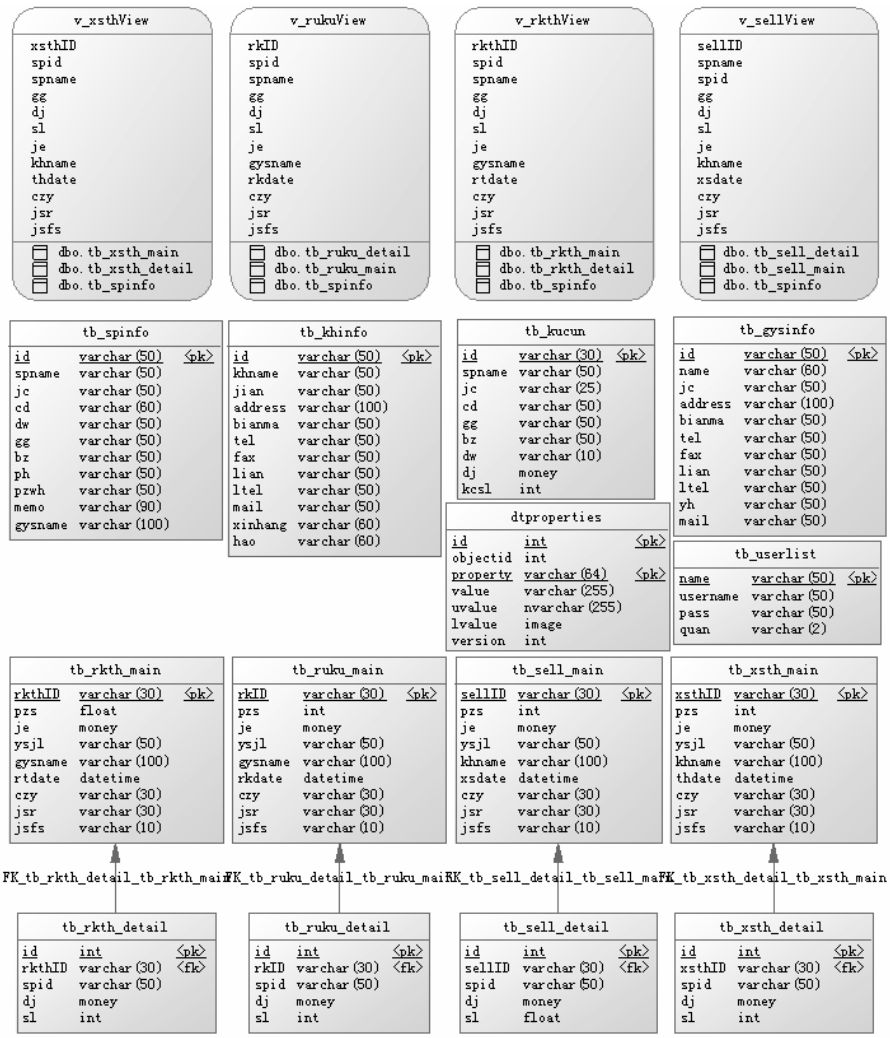


图 1.16　企业进销存管理系统的模型

（11）选择 PowerDesigner 主菜单中的 Database/Generate Database 命令，将打开 Database Generation 对话框。在该对话框中设置导出的脚本文件的名称（如 jxc.sql）及保存路径（如 D:\JXC），选中 Script generation 单选按钮，如图 1.17 所示。单击“确定”按钮，将会在指定的路径中生成数据库脚本文件。

（12）在图 1.17 所示的对话框中选中 Direct generation 单选按钮，可以使用 ODBC 数据源直接在数据库管理系统中生成数据表和视图。但是必须先创建数据库的数据源，然后单击按钮选择指定的数据源，并单击“确定”按钮。

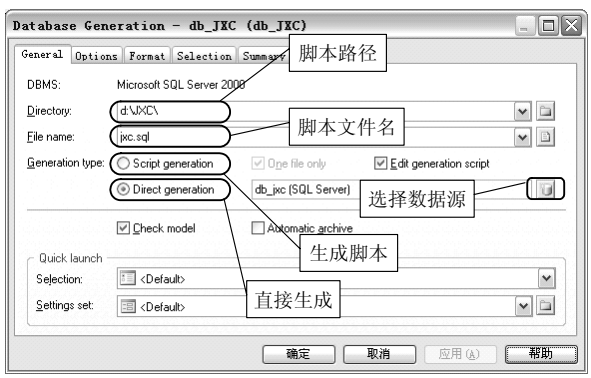


图 1.17　Database Generation 对话框

1.4.4 创建数据库

在导出数据库脚本文件后，即可在查询分析器中执行该脚本来创建数据库及数据表。具体步骤如下：

（1）选择“开始”/“所有程序”/Microsoft SQL Server/“查询分析器”命令，在弹出的“连接到 SQL Server”对话框中输入访问数据库的“登录名”和“密码”，如图 1.18 所示。单击“确定”按钮。

（2）在打开的 SQL 查询分析器中选择“文件”/“打开”命令，在弹出的对话框中选择数据库脚本文件，然后单击“打开”按钮，返回查询分析器，选择执行脚本的数据库，然后单击▶按钮执行脚本中的命令创建数据库的表结构，如图 1.19 所示。

图 1.18 “连接到 SQL Server”对话框

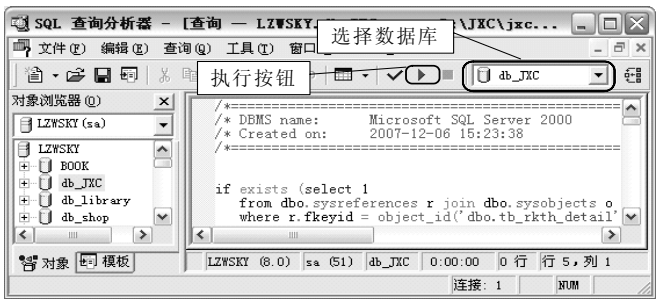


图 1.19 查询分析器运行效果

1.5 主窗体设计

主窗体界面也是该系统的欢迎界面。应用程序的主窗体必须设计层次清晰的系统菜单和工具栏，其中系统菜单包含系统中所有功能的菜单项，而工具栏主要提供常用功能的快捷访问按钮。企业进销存管理系统采用导航面板综合了系统菜单和工具栏的优点，而且导航面板的界面更加美观，操作更快捷。主窗体的界面效果如图 1.20 所示。

图 1.20 程序主窗体界面效果

1.5.1 创建主窗体

创建主窗体的步骤如下：

（1）创建 JXCFrame 类，在类中创建并初始化窗体对象，为窗体添加桌面面板，并设置背景图片。关键代码如下：

例程 01　代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
private JDesktopPane desktopPane;
private JFrame frame;
private JLabel backLabel;
private Preferences preferences;
//创建窗体的 Map 类型集合对象
private Map<String, JInternalFrame> ifs = new HashMap<String, JInternalFrame>();
public JXCFrame() {
    frame = new JFrame("企业进销存管理系统");                                    //创建窗体对象
    frame.addComponentListener(new FrameListener());                          //添加窗体事件监听器
    frame.getContentPane().setLayout(new BorderLayout());                     //设置布局管理器
    frame.setBounds(100, 100, 800, 600);                                      //设置窗体位置和大小
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);                     //设置窗体默认的关闭方式
    backLabel = new JLabel();                                                 //背景标签
    backLabel.setVerticalAlignment(SwingConstants.TOP);                       //设置背景标签垂直对齐方式
    backLabel.setHorizontalAlignment(SwingConstants.CENTER);                  //设置背景标签水平对齐方式
    updateBackImage();                                                        //调用初始化背景标签的方法
    desktopPane = new JDesktopPane();                                         //创建桌面面板
    desktopPane.add(backLabel, new Integer(Integer.MIN_VALUE));               //将背景标签添加到桌面面板中
    frame.getContentPane().add(desktopPane);                                  //添加桌面面板到窗体中
    JTabbedPane navigationPanel = createNavigationPanel();                    //创建导航面板
    frame.getContentPane().add(navigationPanel, BorderLayout.NORTH);          //添加导航面板到窗体中
    frame.setVisible(true);                                                   //显示窗体
}
```

（2）编写 updateBackImage()方法，在该方法中初始化背景标签，背景标签使用 HTML 超文本语言设置了主窗体的背景图片，该图片将随主窗体的大小自动缩放。关键代码如下：

例程 02　代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
private void updateBackImage() {
    if (backLabel != null) {
        int backw = JXCFrame.this.frame.getWidth();
        int backh = frame.getHeight();
        backLabel.setSize(backw, backh);                                      //初始化背景标签的大小
        backLabel.setText("<html><body><image width='" + backw
                + "' height='" + (backh - 110) + "' src="
                + JXCFrame.this.getClass().getResource("welcome.jpg")
                + "'></img></body></html>");                                  //设置背景标签的图像
    }
}
```

（3）在类的静态代码段中设置进销存管理系统的外观样式。Swing 支持跨平台特性，它可以在不同的操作系统中保持一致的外观风格，但是本系统使用 UIManager 类的 setLookAndFeel()方法设置程序界面使用本地外观，这样可以使程序更像本地应用程序。关键代码如下：

例程 03　代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
static {
    try {
```

```
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (Exception e) {
            e.printStackTrace();
        }
}
```

（4）编写主窗体的 main()入口方法，在该方法中创建登录窗体对象，登录窗体会验证登录信息，并显示主窗体界面。关键代码如下：

例程 04 代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new Login();
        }
    });
}
```

1.5.2 创建导航面板

创建导航面板的步骤如下：

（1）在 JXCFrame 类中编写 createNavigationPanel()方法，在该方法中创建 JTabbedPane 选项卡面板对象。为突出选项卡的立体效果，设置该选项卡使用 BevelBorder 边框效果，然后依次创建基础信息管理、库存管理、销售管理、查询统计、进货管理和系统管理的选项卡。关键代码如下：

例程 05 代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
private JTabbedPane createNavigationPanel() {                          //创建导航面板的方法
    JTabbedPane tabbedPane = new JTabbedPane();
    tabbedPane.setFocusable(false);
    tabbedPane.setBackground(new Color(211, 230, 192));
    tabbedPane.setBorder(new BevelBorder(BevelBorder.RAISED));
    JPanel baseManagePanel = new JPanel();                             //基础信息管理面板
    baseManagePanel.setBackground(new Color(215, 223, 194));
    baseManagePanel.setLayout(new BoxLayout(baseManagePanel, BoxLayout.X_AXIS));
    baseManagePanel.add(createFrameButton("客户信息管理", "KeHuGuanLi"));
    baseManagePanel.add(createFrameButton("商品信息管理", "ShangPinGuanLi"));
    baseManagePanel.add(createFrameButton("供应商信息管理", "GysGuanLi"));
    JPanel depotManagePanel = new JPanel();                            //库存管理面板
    depotManagePanel.setBackground(new Color(215, 223, 194));
    depotManagePanel.setLayout(new BoxLayout(depotManagePanel, BoxLayout.X_AXIS));
    depotManagePanel.add(createFrameButton("库存盘点", "KuCunPanDian"));
    depotManagePanel.add(createFrameButton("价格调整", "JiaGeTiaoZheng"));
    JPanel sellManagePanel = new JPanel();                             //销售管理面板
    sellManagePanel.setBackground(new Color(215, 223, 194));
    sellManagePanel.setLayout(new BoxLayout(sellManagePanel, BoxLayout.X_AXIS));
    sellManagePanel.add(createFrameButton("销售单", "XiaoShouDan"));
    sellManagePanel.add(createFrameButton("销售退货", "XiaoShouTuiHuo"));
```

```
    JPanel searchStatisticPanel = new JPanel();                                    //查询统计面板
    searchStatisticPanel.setBounds(0, 0, 600, 41);
    searchStatisticPanel.setName("searchStatisticPanel");
    searchStatisticPanel.setBackground(new Color(215, 223, 194));
    searchStatisticPanel.setLayout(new BoxLayout(searchStatisticPanel, BoxLayout.X_AXIS));
    searchStatisticPanel.add(createFrameButton("客户信息查询", "KeHuChaXun"));
    searchStatisticPanel.add(createFrameButton("商品信息查询", "ShangPinChaXun"));
    searchStatisticPanel.add(createFrameButton("供应商信息查询","GongYingShangChaXun"));
    searchStatisticPanel.add(createFrameButton("销售信息查询", "XiaoShouChaXun"));
    searchStatisticPanel.add(createFrameButton("销售退货查询","XiaoShouTuiHuoChaXun"));
    searchStatisticPanel.add(createFrameButton("入库查询", "RuKuChaXun"));
    searchStatisticPanel.add(createFrameButton("入库退货查询", "RuKuTuiHuoChaXun"));
    searchStatisticPanel.add(createFrameButton("销售排行", "XiaoShouPaiHang"));
    JPanel stockManagePanel = new JPanel();                                        //进货管理面板
    stockManagePanel.setBackground(new Color(215, 223, 194));
    stockManagePanel.setLayout(new BoxLayout(stockManagePanel, BoxLayout.X_AXIS));
    stockManagePanel.add(createFrameButton("进货单", "JinHuoDan"));
    stockManagePanel.add(createFrameButton("进货退货", "JinHuoTuiHuo"));
    JPanel sysManagePanel = new JPanel();                                          //系统管理面板
    sysManagePanel.setBackground(new Color(215, 223, 194));
    sysManagePanel.setLayout(new BoxLayout(sysManagePanel, BoxLayout.X_AXIS));
    sysManagePanel.add(createFrameButton("操作员管理", "CzyGL"));
    sysManagePanel.add(createFrameButton("更改密码", "GengGaiMiMa"));
    sysManagePanel.add(createFrameButton("权限管理", "QuanManager"));
    //将所有面板添加到导航面板中
    tabbedPane.addTab("    基础信息管理    ", null, baseManagePanel, "基础信息管理");
    tabbedPane.addTab("    进货管理    ", null, stockManagePanel, "进货管理");
    tabbedPane.addTab("    销售管理    ", null, sellManagePanel, "销售管理");
    tabbedPane.addTab("    查询统计    ", null, searchStatisticPanel, "查询统计");
    tabbedPane.addTab("    库存管理    ", null, depotManagePanel, "库存管理");
    tabbedPane.addTab("    系统管理    ", null, sysManagePanel, "系统管理");
    return tabbedPane;
}
```

（2）编写 createFrameButton()方法，该方法负责创建 Action 对象，该对象用于创建并显示窗体对象，另外，它还包含图标、文本等属性，如果将 Action 对象添加到系统菜单栏或者工具栏中，会直接创建相应的菜单项和工具按钮，而且这些菜单项和工具按钮将显示 Action 对象中的文本和图标属性。本系统没有使用系统菜单，所以该方法直接创建按钮对象。关键代码如下：

例程 06　代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
private JButton createFrameButton(String fName, String cname) {    //为内部窗体添加 Action 的方法
    String imgUrl = "res/ActionIcon/" + fName + ".png";
    String imgUrl_roll = "res/ActionIcon/" + fName   + "_roll.png";
    String imgUrl_down = "res/ActionIcon/" + fName       + "_down.png";
    Icon icon = new ImageIcon(imgUrl);                                  //创建按钮图标
    Icon icon_roll = null;
    if (imgUrl_roll != null)
        icon_roll = new ImageIcon(imgUrl_roll);                         //创建鼠标经过按钮时的图标
    Icon icon_down = null;
    if (imgUrl_down != null)
```

```
        icon_down = new ImageIcon(imgUrl_down);                    //创建按钮按下的图标
    Action action = new openFrameAction(fName, cname, icon);       //用 openFrameAction 类创建 Action 对象
    JButton button = new JButton(action);
❶   button.setMargin(new Insets(0, 0, 0, 0));
❷   button.setHideActionText(true);
❸   button.setFocusPainted(false);
❹   button.setBorderPainted(false);
❺   button.setContentAreaFilled(false);
    if (icon_roll != null)
❻       button.setRolloverIcon(icon_roll);
    if (icon_down != null)
❼       button.setPressedIcon(icon_down);
    return button;
```

代码贴士

❶ setMargin()：该方法用于设置按钮的四周边界大小。

❷ setHideActionText()：该方法用于设置按钮隐藏 Action 对象中的文本信息，例如一个只显示图标的按钮可以取消文本使按钮更加美观。

❸ setFocusPainted()：该方法用于设置按钮获取焦点时，是否绘制焦点样式。导航面板取消了这个焦点样式，因为它破坏了按钮图标美观性。

❹ setBorderPainted()：该方法设置是否绘制按钮的边框样式，导航面板取消了边框样式，因为按钮的图标需要覆盖整个按钮。

❺ setContentAreaFilled()：该方法设置是否绘制按钮图形，在不同的操作系统，甚至系统不同的皮肤样式中都有不同的图形。导航面板取消了按钮的图形效果，因为导航面板要使用图标绘制整个按钮。

❻ setRolloverIcon()：该方法用于设置鼠标经过按钮时，按钮所使用的图标。

❼ setPressedIcon()：该方法用于设置鼠标按下按钮时，按钮所使用的图标。

（3）编写内部类 openFrameAction，它必须继承 AbstractAction 类实现 Action 接口。该类用于创建导航按钮的 Action 对象，并为每个导航按钮定义创建并显示不同窗体对象的动作监听器，这个监听器在按钮被按下时，调用 getIFrame()方法获取相应的窗体对象，并显示在主窗体中。关键代码如下：

例程 07 代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
protected final class openFrameAction extends AbstractAction {          //主窗体菜单项的单击事件监听器
    private String frameName = null;
    private openFrameAction() {
    }
    public openFrameAction(String cname, String frameName, Icon icon) {
        this.frameName = frameName;
        putValue(Action.NAME, cname);                                    //设置 Action 的名称
        putValue(Action.SHORT_DESCRIPTION, cname);                       //设置 Action 的提示文本框
        putValue(Action.SMALL_ICON, icon);                               //设置 Action 的图标
    }
    public void actionPerformed(final ActionEvent e) {
        JInternalFrame jf = getIFrame(frameName);                        //调用 getIFrame() 方法
        //在内部窗体关闭时，从内部窗体容器 ifs 对象中清除该窗体
        jf.addInternalFrameListener(new InternalFrameAdapter() {
            public void internalFrameClosed(InternalFrameEvent e) {
```

```
                ifs.remove(frameName);
            }
        });
        if (jf.getDesktopPane() == null) {
            desktopPane.add(jf);                                    //将窗体添加到主窗体中
            jf.setVisible(true);                                    //显示窗体
        }
        try {
            jf.setSelected(true);                                   //使窗体处于被选择状态
        } catch (PropertyVetoException e1) {
            e1.printStackTrace();
        }
    }
}
```

（4）编写 getIFrame()方法，该方法负责创建指定名称的窗体对象，在方法中使用了 Java 的反射技术，调用不同窗体类的默认构造方法创建窗体对象。关键代码如下：

例程 08　代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
private JInternalFrame getIFrame(String frameName) {                //获取内部窗体的唯一实例对象
    JInternalFrame jf = null;
    if (!ifs.containsKey(frameName)) {
        try {
            Class fClass = Class.forName("internalFrame." + frameName);
            Constructor constructor = fClass.getConstructor(null);
            jf = (JInternalFrame) constructor.newInstance(null);
            ifs.put(frameName, jf);
        } catch (Exception e) {
            e.printStackTrace();
        }
    } else
        jf = ifs.get(frameName);
    return jf;
}
```

1.6　公共模块设计

在本系统的项目空间中，有部分模块是公用的，或者是多个模块甚至整个系统的配置信息，它们被多个模块重复调用完成指定的业务逻辑，本节将这些公共模块提出来作单独介绍。

1.6.1　编写 Dao 公共类

Dao 类主要负责有关数据库的操作，该类在静态代码段中驱动并连接数据库，然后将所有的数据库访问方法定义为静态的。本节将介绍 Dao 类中有关数据库操作的关键方法。Dao 类的定义代码如下：

例程 09 代码位置：光盘\TM\01\JXCManager\src\com\lzw\dao\Dao.java

```
public class Dao {
❶   protected static String dbClassName = "com.microsoft.jdbc.sqlserver.SQLServerDriver";
❷   protected static String dbUrl = "jdbc:microsoft:sqlserver://localhost:1433;"
            + "DatabaseName=db_JXC;SelectMethod=Cursor";
❸   protected static String dbUser = "sa";
❹   protected static String dbPwd = "";
    protected static String second = null;
❺   public static Connection conn = null;
    static {
        try {
            if (conn == null) {
                Class.forName(dbClassName).newInstance();                    //加载数据库驱动类
                conn = DriverManager.getConnection(dbUrl, dbUser, dbPwd);  //获取数据库连接
            }
        } catch (Exception ee) {
            ee.printStackTrace();
        }
    }
}
```

代码贴士

❶ dbClassName：该成员变量用于定义数据库驱动类的名称。

❷ dbUrl：该成员变量用于定义访问数据库的 URL 路径。

❸ dbUser：该成员变量用于定义访问数据库的用户名称。

❹ dbPwd：该成员变量用于定义访问数据库的用户密码。

❺ conn：该成员变量用于定义连接数据库的对象。

1．addGys()方法

该方法用于添加供应商的基础信息，它接收供应商的实体类 TbGysinfo 作方法参数，然后把实体对象中的所有属性存入供应商数据表中。关键代码如下：

例程 10 代码位置：光盘\TM\01\JXCManager\src\com\lzw\dao\Dao.java

```
//添加供应商信息的方法
public static boolean addGys(TbGysinfo gysInfo) {
    if (gysInfo == null)                                                    //如果供应商实体对象为空
        return false;                                                       //则返回 false
    return insert("insert tb_gysinfo values('" + gysInfo.getId() + "','"   //执行供应商添加
            + gysInfo.getName() + "','" + gysInfo.getJc() + "','"
            + gysInfo.getAddress() + "','" + gysInfo.getBianma() + "','"
            + gysInfo.getTel() + "','" + gysInfo.getFax() + "','"
            + gysInfo.getLian() + "','" + gysInfo.getLtel() + "','"
            + gysInfo.getMail() + "','" + gysInfo.getYh() + "')");
}
```

2．getGysInfo()方法

该方法将根据 Item 对象中封装的供应商 ID 编号和供应商名称获取指定供应商的数据，并将该供应商的数据封装到实体对象中，然后返回该实体对象。关键代码如下：

例程 11　代码位置：光盘\TM\01\JXCManager\src\com\lzw\dao\Dao.java

```
//读取指定供应商信息
public static TbGysinfo getGysInfo(Item item) {
    String where = "name='" + item.getName() + "' ";                    //默认的查询条件以供应商名称为主
    if (item.getId() != null)                                           //如果 Item 对象中存有 ID 编号
        where = "id='" + item.getId() + "' ";                           //则以 ID 编号为查询条件
    TbGysinfo info = new TbGysinfo();
    ResultSet set = findForResultSet("select * from tb_gysinfo where "+ where);
    try {
        if (set.next()) {
            info.setId(set.getString("id").trim());                     //封装供应商数据到实体对象中
            info.setAddress(set.getString("address").trim());
            info.setBianma(set.getString("bianma").trim());
            info.setFax(set.getString("fax").trim());
            info.setJc(set.getString("jc").trim());
            info.setLian(set.getString("lian").trim());
            info.setLtel(set.getString("ltel").trim());
            info.setMail(set.getString("mail").trim());
            info.setName(set.getString("name").trim());
            info.setTel(set.getString("tel").trim());
            info.setYh(set.getString("yh").trim());
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return info;                                                        //返回供应商实体对象
}
```

3．updateGys()方法

该方法用于更新供应商的基础信息，它接收供应商的实体类 TbGysinfo 作方法参数，在方法中直接解析供应商实体对象中的属性，并将这些属性更新到数据表中。关键代码如下：

例程 12　代码位置：光盘\TM\01\JXCManager\src\com\lzw\dao\Dao.java

```
//修改供应商信息的方法
public static int updateGys(TbGysinfo gysInfo) {
    return update("update tb_gysinfo set jc='" + gysInfo.getJc()
            + "',address='" + gysInfo.getAddress() + "',bianma='"
            + gysInfo.getBianma() + "',tel='" + gysInfo.getTel()
            + "',fax='" + gysInfo.getFax() + "',lian='" + gysInfo.getLian()
            + "',ltel='" + gysInfo.getLtel() + "',mail='"
            + gysInfo.getMail() + "',yh='" + gysInfo.getYh()
            + "' where id='" + gysInfo.getId() + "'");
}
```

4．insertRukuInfo()方法

该方法负责完成入库单信息的添加，它涉及库存表、入库主表和入库详细表等多个数据表的操作。为保证数据的完整性，该方法将入库信息的添加操作放在事务中完成，方法将接收入库主表的实体类 TbRukuMain 作参数，该实体类中包含了入库详细表的引用。关键代码如下：

例程 13 代码位置：光盘\TM\01\JXCManager\src\com\lzw\dao\Dao.java

```
public static boolean insertRukuInfo(TbRukuMain ruMain) {                    //在事务中添加入库信息
    try {
❶       boolean autoCommit = conn.getAutoCommit();
❷       conn.setAutoCommit(false);                                           //取消自动提交模式
        insert("insert into tb_ruku_main values('" + ruMain.getRkId()        //添加入库主表记录
                + "','" + ruMain.getPzs() + "'," + ruMain.getJe() + ",'"
                + ruMain.getYsjl() + "','" + ruMain.getGysname() + "','"
                + ruMain.getRkdate() + "','" + ruMain.getCzy() + "','"
                + ruMain.getJsr() + "','" + ruMain.getJsfs() + "')");
        Set<TbRukuDetail> rkDetails = ruMain.getTabRukuDetails();
        for (Iterator<TbRukuDetail> iter = rkDetails.iterator(); iter.hasNext();) {
            TbRukuDetail details = iter.next();
            insert("insert into tb_ruku_detail values('" + ruMain.getRkId()  //添加入库详细表记录
                    + "','" + details.getTabSpinfo() + "'," + details.getDj() + "," + details.getSl() + ")");
            Item item = new Item();
            item.setId(details.getTabSpinfo());
            TbSpinfo spInfo = getSpInfo(item);
            if (spInfo.getId() != null && !spInfo.getId().isEmpty()) {
                TbKucun kucun = getKucun(item);
                if (kucun.getId() == null || kucun.getId().isEmpty()) {      //添加或修改库存表记录
                    insert("insert into tb_kucun values('" + spInfo.getId()
                            + "','" + spInfo.getSpname() + "','"+ spInfo.getJc() + "','" + spInfo. getCd()
                            + "','" + spInfo.getGg() + "','"+ spInfo.getBz() + "','" + spInfo.getDw()
                            + "'," + details.getDj() + ","+ details.getSl() + ")");
                } else {
                int sl = kucun.getKcsl() + details.getSl();
                update("update tb_kucun set kcsl=" + sl + ",dj="+ details.getDj() + " where id='"+ kucun.
getId() + "'");
                }
            }
        }
❸       conn.commit();                                                       //提交事务
        conn.setAutoCommit(autoCommit);                                      //恢复自动提交模式
    } catch (SQLException e) {
        try {
❹           conn.rollback();                                                 //如果出错，回退事务
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
        e.printStackTrace();
    }
    return true;
}
```

代码贴士

❶ getAutoCommit()：该方法用于获取事务的自动提交模式。

❷ setAutoCommit()：该方法用于设置事务的自动提交模式。

❸ commit()：该方法用于执行事务提交。

❹ rollback ()：该方法在事务执行失败时，执行回退操作。

5．getKucun()方法

该方法用于获取指定商品 ID 编号或名称的库存信息。方法接收一个 Item 对象作参数，该对象中封装了商品的 ID 编号和商品名称信息，如果库存表中存在该商品的库存记录，就获取该记录并将记录中的数据封装到库存表的实体对象中，然后将该实体对象作为方法的返回值。关键代码如下：

例程 14　代码位置：光盘\TM\01\JXCManager\src\com\lzw\dao\Dao.java

```
//获取库存商品信息
public static TbKucun getKucun(Item item) {
    String where = "spname='" + item.getName() + "'";
    if (item.getId() != null)
        where = "id='" + item.getId() + "'";
    ResultSet rs = findForResultSet("select * from tb_kucun where " + where);
    TbKucun kucun = new TbKucun();
    try {
        if (rs.next()) {
            kucun.setId(rs.getString("id"));
            kucun.setSpname(rs.getString("spname"));
            kucun.setJc(rs.getString("jc"));
            kucun.setBz(rs.getString("bz"));
            kucun.setCd(rs.getString("cd"));
            kucun.setDj(rs.getDouble("dj"));
            kucun.setDw(rs.getString("dw"));
            kucun.setGg(rs.getString("gg"));
            kucun.setKcsl(rs.getInt("kcsl"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return kucun;
}
```

1.6.2　编写 Item 类

Item 类是系统的公共类之一，主要用于封装和传递参数信息，这是典型命令模式的实现。在 Dao 类中经常使用该类作为方法参数，另外，在各个窗体界面中也经常使用该类作组件数据，其 toString() 方法将返回 name 属性值，所以显示到各个组件上的内容就是 Item 类的对象所代表的商品、供应商或者客户等信息中的名称。定义该类的关键代码如下：

例程 15 代码位置：光盘\TM\01\JXCManager\src\internalFrame\guanli\Item.java

```
public class Item {
    public String id;                          //定义 ID 属性
    public String name;                        //定义名称属性
    public String getId() {                    //定义暴露 ID 属性的方法
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {                  //定义暴露名称属性的方法
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String toString() {                 //定义该类的字符串表现形式
        return getName();
    }
}
```

1.7 基础信息模块设计

基础信息模块用于管理企业进销存管理系统中的客户、商品和供应商信息，其功能主要是对这些基础信息进行添加、修改和删除。

1.7.1 基础信息模块概述

企业进销存管理系统中的基础信息模块主要包括客户管理、商品管理和供应商管理 3 部分，由于它们的实现方法基本相似，本节将以供应商管理部分为主，介绍基础信息模块对本系统的意义和实现的业务逻辑。

1．供应商添加

供应商添加功能主要负责为系统添加新的供应商记录。在企业进销存管理系统中，商品是主要的管理对象，而系统中所有的商品都由不同的供应商提供，这就需要把不同的供应商信息添加到系统中，在商品信息中会关联系统中对应的供应商信息。供应商添加功能的程序界面如图 1.21 所示。

2．供应商修改与删除

供应商的修改与删除功能主要用于维护系统中的供应商信息。在供应商的联系方式发生改变时，必须更新系统中的记录，以提供供应商的最新信息。另外，当不再与某家供应商合作时，需要从系统中删除供应商的记录信息。程序运行界面如图 1.22 所示。

图 1.21　供应商添加界面

图 1.22　供应商修改与删除功能界面

1.7.2　基础信息模块技术分析

基础信息模块中使用了 Java Swing 的 JTabbedPane 选项卡面板组件分别为客户信息管理、商品信息管理和供应商信息管理提供多个操作界面，例如供应商信息管理中分别存在供应商添加和供应商修改与删除界面，而这两个界面都存在于一个窗体中，可以通过选择顶部的两个选项卡，在不同的界面之间来回切换。

1.7.3　供应商添加的实现过程

供应商添加使用的数据表：tb_gysinfo

开发供应商添加的步骤如下：

（1）创建 GysTianJiaPanel 类，用于实现本系统的供应商添加功能。该类将在界面中显示多个用于输入供应商信息的文本框。界面中定义的主要控件如表 1.6 所示。

表 1.6　供应商添加界面中的主要控件

控 件 类 型	控 件 名 称	主要属性设置	用　途
JtextField	quanChengF	无	供应商全称
	JianChengF	无	简称
	BianMaF	无	邮政编码
	DiZhiF	无	地址
	DianHuaF	无	电话
	ChuanZhenF	无	传真
	LianXiRenF	无	联系人
	lianXiRenDianHuaF	无	联系人电话
	YinHangF	无	开户银行
	EmailF	无	电子信箱
Jbutton	TjButton	设置按钮文本为“添加”，设置动作监听器为 TjActionListener 类的实例对象	添加
	ResetButton	设置按钮文本为“重填”，设置动作监听器为 ResetActionListener 类的实例对象	重填

（2）创建 ResetActionListener 类，该类是“重填”按钮的事件监听器，它必须实现 ActionListener 接口，并在 actionPerformed()方法中清除界面中的所有文本框内容。关键代码如下：

例程 16 代码位置：光盘\TM\01\JXCManager\src\internalFrame\gysGuanLi\GysTianJiaPanel.java

```
❶ class ResetActionListener implements ActionListener {        //“重填”按钮的事件监听类
❷       public void actionPerformed(
❸                                   final ActionEvent e) {
            diZhiF.setText("");                                  //将文本框中的内容设置为空字符串
            bianMaF.setText("");
            chuanZhenF.setText("");
            jianChengF.setText("");
            lianXiRenF.setText("");
            lianXiRenDianHuaF.setText("");
            EMailF.setText("");
            quanChengF.setText("");
            dianHuaF.setText("");
            yinHangF.setText("");
        }
    }
```

代码贴士

❶ ActionListener：该接口是控件的动作监听器接口，实现该接口的类可以成为按钮和菜单项等控件的监听器。

❷ actionPerformed()：该方法是监听器 ActionListener 接口定义的方法，当事件产生时，将调用监听器实现类的 actionPerformed()方法处理相应的业务逻辑。

❸ ActionEvent：该类是动作事件类，当用户单击按钮时，将产生该事件，这个事件会被监听器捕获并执行相应的业务逻辑。

（3）创建 TjActionListener 类，该类是“添加”按钮的事件监听器，它必须实现 ActionListener 接口，并在 actionPerformed()方法中实现用户输入的验证和供应商信息的保存。关键代码如下：

例程 17 代码位置：光盘\TM\01\JXCManager\src\internalFrame\gysGuanLi\GysTianJiaPanel.java

```
class TjActionListener implements ActionListener {                  //“添加”按钮的事件监听类
    public void actionPerformed(final ActionEvent e) {
        if (diZhiF.getText().equals("") || quanChengF.getText().equals("")      //验证用户输入
                || chuanZhenF.getText().equals("")|| jianChengF.getText().equals("")
                || yinHangF.getText().equals("")|| bianMaF.getText().equals("")
                || diZhiF.getText().equals("")|| lianXiRenF.getText().equals("")
                || lianXiRenDianHuaF.getText().equals("")
                || EMailF.getText().equals("")|| dianHuaF.getText().equals("")) {
            JOptionPane.showMessageDialog(GysTianJiaPanel.this, "请填写全部信息");
            return;
        }
        try {                                                            //验证是否存在同名供应商
            ResultSet haveUser = Dao.query("select * from tb_gysinfo where name='"
                                    + quanChengF.getText().trim() + "'");
            if (haveUser.next()) {
```

```
                JOptionPane.showMessageDialog(GysTianJiaPanel.this,
                        "供应商信息添加失败，存在同名供应商", "供应商添加信息",
                        JOptionPane.INFORMATION_MESSAGE);
                return;
            }
            ResultSet set = Dao.query("select max(id) from tb_gysinfo");    //获取供应商的最大 ID 编号
            String id = null;
            if (set != null && set.next()) {                                //创建新的供应商编号
                String sid = set.getString(1).trim();
                if (sid == null)
                    id = "gys1001";
                else {
                    String str = sid.substring(3);
                    id = "gys" + (Integer.parseInt(str) + 1);
                }
            }
            TbGysinfo gysInfo = new TbGysinfo();                            //创建供应商实体对象
            gysInfo.setId(id);                                              //初始化供应商对象
            gysInfo.setAddress(diZhiF.getText().trim());
            gysInfo.setBianma(bianMaF.getText().trim());
            gysInfo.setFax(chuanZhenF.getText().trim());
            gysInfo.setYh(yinHangF.getText().trim());
            gysInfo.setJc(jianChengF.getText().trim());
            gysInfo.setName(quanChengF.getText().trim());
            gysInfo.setLian(lianXiRenF.getText().trim());
            gysInfo.setLtel(lianXiRenDianHuaF.getText().trim());
            gysInfo.setMail(EMailF.getText().trim());
            gysInfo.setTel(dianHuaF.getText().trim());
            Dao.addGys(gysInfo);                                            //调用 addGys()方法存储供应商
            JOptionPane.showMessageDialog(GysTianJiaPanel.this, "已成功添加客户",
                    "客户添加信息", JOptionPane.INFORMATION_MESSAGE);
            resetButton.doClick();                                          //触发“重填”按钮的单击动作
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
    }
}
```

1.7.4　供应商修改与删除的实现过程

供应商修改与删除使用的数据表：tb_gysinfo

开发供应商修改与删除的步骤如下：

（1）创建 GysXiuGaiPanel 类，用于实现本系统的供应商修改功能。在程序界面中有多个用于输入供应商信息的文本框，这些文本框的内容会根据所选供应商自动填充内容，修改部分或全部内容后，单击“修改”按钮将修改供应商数据。界面中定义的主要控件如表 1.7 所示。

表 1.7　供应商修改与删除界面中的主要控件

控 件 类 型	控 件 名 称	主要属性设置	用　　途
JtextField	quanChengF	无	供应商全称
	jianChengF	无	简称
	bianMaF	无	邮政编码
	diZhiF	无	地址
	dianHuaF	无	电话
	chuanZhenF	无	传真
	lianXiRenF	无	联系人
	lianXiRenDianHuaF	无	联系人电话
	yinHangF	无	开户银行
	EMailF	无	电子信箱
JcomboBox	Gys	设置初始大小为（230, 21），调用 initComboBox()方法初始化下拉列表，设置组件的选择事件调用 doGysSelectAction()方法	选择供应商
Jbutton	tjButton	设置按钮文本为“修改”，设置动作监听器为 ModifyActionListener 类的实例对象	修改供应商信息
	resetButton	设置按钮文本为“删除”，设置动作监听器为 DelActionListener 类的实例对象	删除供应商信息

（2）编写 initComboBox()方法，用于初始化选择供应商的下拉列表框。该方法调用 Dao 类的 getGysInfos()方法获取数据库中所有的供应商信息，然后将供应商的 ID 编号和供应商名称封装成 Item 对象并添加到选择供应商的下拉列表框中，在下拉列表框中 Item 的 toString()方法将显示供应商的名称。initComboBox()方法的关键代码如下：

例程 18　代码位置：光盘\TM\01\JXCManager\src\internalFrame\gysGuanLi\GysXiuGaiPanel.java

```
public void initComboBox() {                                    //初始化供应商下拉列表框的方法
    List gysInfo = Dao.getGysInfos();                           //调用 getGysInfos()方法获取供应商列表
    List<Item> items = new ArrayList<Item>();                   //创建 Item 列表
    gys.removeAllItems();                                       //清除下拉列表框中原有的选项
    for (Iterator iter = gysInfo.iterator(); iter.hasNext();) {
        List element = (List) iter.next();
        Item item = new Item();                                 //封装供应商信息
        item.setId(element.get(0).toString().trim());
        item.setName(element.get(1).toString().trim());
        if (items.contains(item))                               //如果 Item 列表中包含该供应商的封装对象
            continue;                                           //跳出本次循环
        items.add(item);
        gys.addItem(item);                                      //否则添加该对象到下拉列表框中
    }
    doGysSelectAction();                                        //doGysSelectAction() 方法
}
```

（3）编写 doGysSelectAction()方法，它在更改下拉列表框中的供应商信息时被调用，主要用于根据选择的供应商名称，把供应商的其他信息填充到相应的文本框中。关键代码如下：

例程 19　代码位置：光盘\TM\01\JXCManager\src\internalFrame\gysGuanLi\GysXiuGaiPanel.java

```
private void doGysSelectAction() {                          //处理供应商选择事件
    Item selectedItem;
    if (!(gys.getSelectedItem() instanceof Item)) {
        return;
    }
    selectedItem = (Item) gys.getSelectedItem();            //获取 Item 对象
    TbGysinfo gysInfo = Dao.getGysInfo(selectedItem);//通过 Item 对象调用 getGysInfo() 方法获取供应商信息
    quanChengF.setText(gysInfo.getName());                  //填充供应商信息到文本框中
    diZhiF.setText(gysInfo.getAddress());
    jianChengF.setText(gysInfo.getJc());
    bianMaF.setText(gysInfo.getBianma());
    dianHuaF.setText(gysInfo.getTel());
    chuanZhenF.setText(gysInfo.getFax());
    lianXiRenF.setText(gysInfo.getLian());
    lianXiRenDianHuaF.setText(gysInfo.getLtel());
    EMailF.setText(gysInfo.getMail());
    yinHangF.setText(gysInfo.getYh());
}
```

（4）创建 ModifyActionListener 类，该类是“修改”按钮的事件监听器，它必须实现 ActionListener 接口，并在 actionPerformed()方法中获取所有文本框的内容，其中包括修改后的信息，并通过调用 updateGys()方法将这些供应商信息更新到数据库中。关键代码如下：

例程 20　代码位置：光盘\TM\01\JXCManager\src\internalFrame\gysGuanLi\GysXiuGaiPanel.java

```
class ModifyActionListener implements ActionListener {                  //“修改”按钮的事件监听器
    public void actionPerformed(ActionEvent e) {
        Item item = (Item) gys.getSelectedItem();
        TbGysinfo gysInfo = new TbGysinfo();                             //创建供应商实体对象
        gysInfo.setId(item.getId());                                     //初始化供应商实体对象
        gysInfo.setAddress(diZhiF.getText().trim());
        gysInfo.setBianma(bianMaF.getText().trim());
        gysInfo.setFax(chuanZhenF.getText().trim());
        gysInfo.setYh(yinHangF.getText().trim());
        gysInfo.setJc(jianChengF.getText().trim());
        gysInfo.setName(quanChengF.getText().trim());
        gysInfo.setLian(lianXiRenF.getText().trim());
        gysInfo.setLtel(lianXiRenDianHuaF.getText().trim());
        gysInfo.setMail(EMailF.getText().trim());
        gysInfo.setTel(dianHuaF.getText().trim());
        if (Dao.updateGys(gysInfo) == 1)                                 //更新供应商信息
            JOptionPane.showMessageDialog(GysXiuGaiPanel.this, "修改完成");
        else
            JOptionPane.showMessageDialog(GysXiuGaiPanel.this, "修改失败");
    }
}
```

（5）创建 DelActionListener 类，该类是“删除”按钮的事件监听器，它必须实现 ActionListener 接口，并在 actionPerformed()方法中获取当前选择的供应商，然后调用 Dao 类的 delete()方法从数据库

中将该供应商删除。关键代码如下：

例程 21 代码位置：光盘\TM\01\JXCManager\src\internalFrame\gysGuanLi\GysXiuGaiPanel.java

```
class DelActionListener implements ActionListener {                    //“删除”按钮的事件监听器
    public void actionPerformed(ActionEvent e) {
        Item item = (Item) gys.getSelectedItem();                      //获取当前选择的供应商
        if (item == null || !(item instanceof Item))
            return;
        int confirm = JOptionPane.showConfirmDialog(                   //弹出确认删除对话框
                GysXiuGaiPanel.this, "确认删除供应商信息吗？");
        if (confirm == JOptionPane.YES_OPTION) {                       //如果确认删除
            int rs = Dao.delete("delete tb_gysInfo where id='"         //调用 delete()方法
                    + item.getId() + "'");
            if (rs > 0) {
                JOptionPane.showMessageDialog(GysXiuGaiPanel.this,     //显示删除成功对话框
                        "供应商：" + item.getName() + "。删除成功");
                gys.removeItem(item);
            } else {
                JOptionPane.showMessageDialog(GysXiuGaiPanel.this,
                        "无法删除客户：" + item.getName() + "。");
            }
        }
    }
}
```

1.7.5 单元测试

在现代软件开发过程中，测试不再作为一个独立的生命周期，单元测试成为与编写代码同步进行的开发活动。单元测试能够提高程序员对程序的信心，保证程序的质量，加快软件开发速度，使程序易于维护。

1．单元测试概述

单元测试是在软件开发过程中要进行的最低级别的测试活动，在单元测试活动中，软件的独立工作单元将在与程序的其他部分相隔离的情况下进行测试。

在一种传统的结构化编程语言中，如 Java 语言，要进行测试的工作单元一般是方法。在像 C++这样的面向对象的语言中，要进行测试的基本单元是类。单元测试不仅仅是作为无错编码的一种辅助手段在一次性的开发过程中使用，单元测试还必须是可重复的，无论是在软件修改或是移植到新的运行环境的过程中。因此，所有的测试都必须在整个软件系统的生命周期中进行。

2．什么是单元测试

☑ 它是一种验证行为

程序中的每一项功能都可以通过单元测试来验证其正确性。它为以后的开发提供支持。就算是开发后期，也可以轻松地增加功能或更改程序结构，而不用担心这个过程中会破坏重要的东西。而且它为代码的重构提供了保障。这样，开发者就可以更自由地对程序进行改进。

☑　它是一种设计行为

编写单元测试将使开发者从调用者的角度观察、思考。特别是先写测试（test-first），迫使开发者把程序设计成易于调用和可测试的，即迫使解除软件中的耦合。

☑　它是一种编写文档的行为

单元测试是一种无价的文档，它是展示函数或类如何使用的最佳文档。这份文档是可编译、可运行的，它永远保持与代码同步。

3．越到项目后期，单元测试为何越难进行

在很多项目的初期，项目中的大部分程序员都能够自觉地编写单元测试。随着项目的进展、任务的加重，离交付时间越来越近，不能按时完成项目的风险越来越大，单元测试就往往成为牺牲品了。项目经理因为进度的压力也不重视了，程序员也因为编码的压力和无人看管而不再为代码编写单元测试了。笔者亲身经历的项目都或多或少地发生过类似这样的事情。越是在项目的后期，能够坚持编写单元测试的程序员在整个项目组中所占比例越来越低。

为了追赶项目进度，多数程序员将没有经过任何测试的程序代码上传到版本控制系统，项目经理也不再追问，照单全收。这样做的结果就是在项目后期，技术骨干人员只好加班加点进行系统集成。集成完了之后，下发给测试人员测试时，Bug 的报告数量翻倍增长。程序员开始修改 Bug，但有非常多的 Bug 隐藏得很深，一直潜伏到生产环境中去。

4．JUnit 单元测试工具的介绍与使用

☑　JUnit 使用介绍

JUnit 是一个单元测试框架，专门用于测试 Java 开发的程序，同类产品还包括 Nunit（.Net）、CPPUnit（C++），都属于 JUnit 中的成员。目前 JUnit 的最新版本是 JUnit 4.0，在 Eclipse 开发工具中已经集成了 JUnit 的多个版本。

在正式讲解 JUnit 之前，先来看一下单元测试的运行效果，如图 1.23 和图 1.24 所示。

图 1.23　单元测试通过效果

在图 1.23 和图 1.24 中很容易发现不同颜色的警示条，图 1.23 所示是绿色的，图 1.24 所示是红色的。如果所有测试案例运行成功，就为绿色；反之，如果有一个不成功，则为红色。

☑　使用 JUnit 进行单元测试

下面开始按步骤讲解如何在 Eclipse 中使用 JUnit 工具。

（1）为单元测试代码创建一个 Java 项目，将其命名为 JUnitTest。

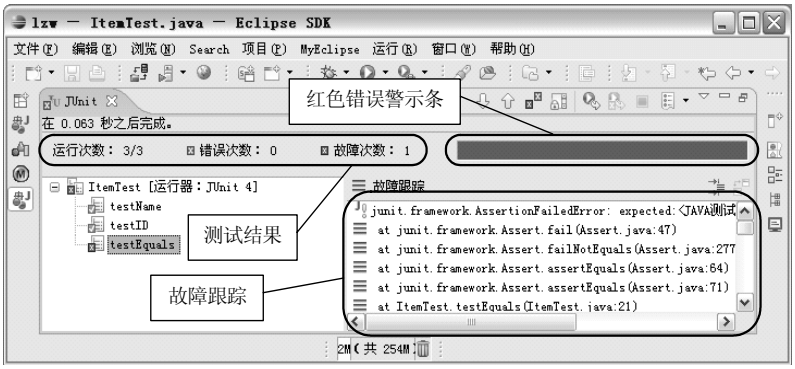


图 1.24　单元测试失败效果

（2）创建 ItemTest 类，该类用于测试公共类 Item 的行为（即方法）。在“新建 Java 类”对话框中设置该类的超类为 TestCase，也就是继承 JUnit 框架的测试用例编写单元测试，单击“完成”按钮，如图 1.25 所示。

（3）在项目的构建路径中添加 JUnit 类库。右击项目名称，在弹出的快捷菜单中选择“构建路径”/“添加库”命令，在弹出的“添加库”对话框中选择 JUnit 选项，单击“下一步”按钮，如图 1.26 所示。

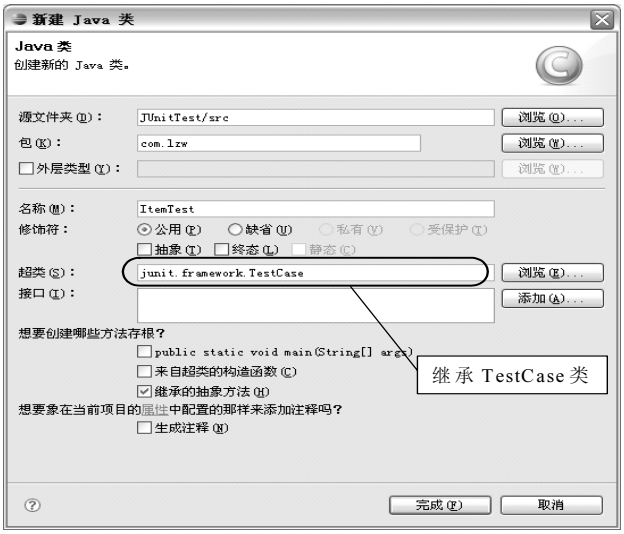


图 1.25　新建测试用例类

图 1.26　“添加库”对话框

（4）在弹出的对话框中选择 JUnit 的版本为 JUnit 4，单击“完成”按钮。

（5）在创建的 ItemTest 中，对 Item 类进行单元测试。Item 是本系统的公共类之一，要实现该类的单元测试，需要编写以 test 作方法名称的前缀，创建 testName()、testID()和 testEquals()方法。另外还要重写父类的 setUp()方法，在该方法中创建并初始化测试用例中需要的数据。

完整代码如下：

```
    import internalFrame.guanli.Item;
    import junit.framework.TestCase;
❶   public class ItemTest extends TestCase{
    private Item item;
```

```
❷       protected void setUp() throws Exception {
            item=new Item();
            item.setId("007");
            item.setName("JAVA 测试");
        }
❸       public void testID(){
            assertEquals(item.getId(), "007");
        }
❹       public void testName(){
            assertEquals(item.getName(), "JAVA 测试");
        }
❺       public void testEquals(){
            Item newItem=new Item();
            newItem.setId("007");
            newItem.setName("JAVA 测试");
            assertEquals(item, newItem);
        }
    }
```

代码贴士

❶ TestCase：该类是 JUnit 框架的测试用例类，所有的单元测试都需要继承该类。

❷ setUp()：该方法将在单元测试之前，为本类的所有单元测试提供测试数据。

❸ testID()：该方法用于测试 Item 类的 getId()方法。

❹ testName()：该方法用于测试 Item 类的 getName()方法。

❺ testEquals()：该方法用于测试 Item 类的相等性。

（6）在该类上单击鼠标右键，在弹出的快捷菜单中选择“运行方式”/“JUnit 测试”命令，运行 Item 类的单元测试，根据警示条中的颜色，即可判断单元测试的成功与失败，如图 1.23 和图 1.24 所示。因为在本系统中不需要判断 Item 实例的相等性，所以 Item 类没有实现父类的 equals()方法，可以不进行该测试，否则在判断两个 Item 类的实例对象是否相等时，将出现判断失败。

1.8　进货管理模块设计

进货管理模块是进销存管理系统中不可缺少的重要组成部分，它主要负责为系统记录进货单及其退货信息，相应的进货商品会添加到库存管理中。

1.8.1　进货管理模块概述

企业进销存管理系统中的进货管理模块主要包括进货单和进货退货两个部分。由于它们的实现方法基本相似，本节将以进货单功能为主，介绍进货管理模块对本系统的意义和实现的业务逻辑。

1．进货单

进货单功能主要负责记录企业的商品进货信息，可以单击“添加”按钮，在商品表中添加进货的

商品信息。在“供应商”下拉列表框中选择不同的供应商，将会改变商品表中可以添加的商品。进货单的程序界面如图 1.27 所示。

2．入库退货

入库退货功能主要负责记录进货管理中的退货信息，界面效果如图 1.28 所示。在选择了退货的商品后，单击“退货”按钮，将把表格中的商品退货信息更新到数据库中。

图 1.27　进货单程序界面

图 1.28　入库退货程序界面

1.8.2　进货管理模块技术分析

进货管理模块使用 JDBC 实现事务操作，因为进货和退货的业务逻辑涉及 3 个数据表，为保证数据的完整性，将 3 个数据表的操作放在事务中实现，如果对任何一个数据表的操作出现错误或是不可执行的操作，那么整个事务中的所有操作都将取消，并恢复到事务执行之前的数据状态；否则 3 个数据表的操作全部执行。下面介绍使用 JDBC 实现事务操作的关键方法。

1．setAutoCommit()方法

该方法用于设置连接对象的自动提交模式。如果连接处对象的自动提交模式为 true，则它的所有 SQL 语句将被执行并作为单个事务提交；否则，该连接对象的 SQL 语句将聚集到事务中，直到调用 commit()或 rollback()方法为止。默认情况下，新连接的自动提交模式为 true。其语法格式如下：

```
void setAutoCommit(boolean autoCommit)
```

autoCommit：该参数为 true，表示启用连接对象的自动提交模式；为 false，表示禁用连接对象的自动提交模式。

2．getAutoCommit()方法

该方法用于判断此连接对象是否启用了自动提交模式。其语法格式如下：

```
boolean getAutoCommit()
```

3．commit()方法

该方法将执行提交 SQL 语句执行数据库操作，并释放此连接对象当前持有的所有数据库锁。此方法只在禁用自动提交模式情况下使用。其语法格式如下：

```
void commit()
```

4．rollback()方法

该方法将取消在当前事务中进行的所有更改，并释放此连接对象当前持有的所有数据库锁。此方法只在禁用自动提交模式情况下使用。其语法格式如下：

```
void rollback()
```

1.8.3　进货单的实现过程

进货单使用的数据表：tb_ruku_main、tb_ruku_detail、tb_kucun

开发进货单的步骤如下：

（1）创建 JinHuoDan 类，用于实现本系统的进货单功能的界面和业务逻辑。界面中定义的主要控件如表 1.8 所示。

表 1.8　进货单界面中的主要控件

控 件 类 型	控 件 名 称	主要属性设置	用　　途
JTextField	PiaoHao	设置该控件不接受输入焦点	进货单票号
	Pzs	设置该控件不接受输入焦点	品种数
	Hpzs	设置该控件不接受输入焦点	货品总数
	Hjje	设置该控件不接受输入焦点	合计金额
	Ysjl	无	验收结论
	Czy	设置该控件不接受输入焦点	操作员
	Jhsj	设置该控件不接受输入焦点	进货时间
	Jsr	无	经手人
	Lian	设置该控件不接受输入焦点	联系人
JComboBox	Jsfs	添加“现金”和“支票”两个下拉列表项	结算方式
	Gys	从数据库中获取供应商并添加到下拉列表中	供应商
	Sp	从数据库中获取商品并添加到下拉列表中	商品
JTable	Table	调用 initTable()方法初始化表格，取消表格列大小的自动调整	商品表格
Jbutton	TjButton	设置按钮文本为“添加”，设置动作监听器为 TjActionListener 类的实例对象	添加
	RkButton	设置按钮文本为“入库”，设置动作监听器为 RkActionListener 类的实例对象	入库

（2）编写 initTable()方法，该方法用于初始化商品表格的表头、列编辑器等。设置表格中第一个列的编辑器，使用下拉列表框样式的编辑器，通过该编辑器选择商品的名称，其他的商品信息将自动填充。关键代码如下：

例程 22 代码位置：光盘\TM\01\JXCManager\src\internalFrame\JinHuoDan.java

```
private void initTable() {                                          //初始化表格
    String[] columnNames = {"商品名称", "商品编号", "产地", "单位", "规格", "包装", "单价",
            "数量", "批号", "批准文号"};
    ((DefaultTableModel) table.getModel())
            .setColumnIdentifiers(columnNames);                     //设置表格的表头
    TableColumn column = table.getColumnModel().getColumn(0);       //获取第一个表格列
    final DefaultCellEditor editor = new DefaultCellEditor(sp);     //创建表格列编辑器
    editor.setClickCountToStart(2);
    column.setCellEditor(editor);
}
```

（3）编写 initSpBox()方法，该方法用于初始化表格中的商品下拉列表框。它首先调用 Dao 类的 query()方法获取指定供应商所提供的所有商品信息，然后将这些商品信息封装成商品对象，并把这些对象添加到商品下拉列表框中。关键代码如下：

例程 23 代码位置：光盘\TM\01\JXCManager\src\internalFrame\JinHuoDan.java

```
private void initSpBox() {                                          //初始化商品下拉列表框
    List list = new ArrayList();
    ResultSet set = Dao.query("select * from tb_spinfo where gysName='"
            + gys.getSelectedItem() + "'");                         //调用 query()方法
    sp.removeAllItems();
    sp.addItem(new TbSpinfo());
    for (int i = 0; table != null && i < table.getRowCount(); i++) {
        TbSpinfo tmpInfo = (TbSpinfo) table.getValueAt(i, 0);
        if (tmpInfo != null && tmpInfo.getId() != null)
            list.add(tmpInfo.getId());
    }
    try {
        while (set.next()) {
            TbSpinfo spinfo = new TbSpinfo();                       //创建商品对象
            spinfo.setId(set.getString("id").trim());               //初始化商品对象
            //如果表格中已存在同样商品，商品下拉列表框中就不再包含该商品
            if (list.contains(spinfo.getId()))
                continue;
❶           spinfo.setSpname(set.getString("spname").trim());       //封装商品信息
❷           spinfo.setCd(set.getString("cd").trim());
❸           spinfo.setJc(set.getString("jc").trim());
❹           spinfo.setDw(set.getString("dw").trim());
❺           spinfo.setGg(set.getString("gg").trim());
❻           spinfo.setBz(set.getString("bz").trim());
❼           spinfo.setPh(set.getString("ph").trim());
❽           spinfo.setPzwh(set.getString("pzwh").trim());
❾           spinfo.setMemo(set.getString("memo").trim());
❿           spinfo.setGysname(set.getString("gysname").trim());
            sp.addItem(spinfo);                                     //将商品对象添加到下拉列表框
        }
    } catch (SQLException e) {
```

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

代码贴士

❶ setSpname()：该方法用于设置商品实体类的商品名称。
❷ setCd()：该方法用于设置商品实体类的商品产地。
❸ setJc()：该方法用于设置商品实体类的商品简称。
❹ setDw()：该方法用于设置商品实体类的商品单位。
❺ setGg()：该方法用于设置商品实体类的商品规格。
❻ setBz()：该方法用于设置商品实体类的商品包装。
❼ setPh()：该方法用于设置商品实体类的商品批号。
❽ setPzwh()：该方法用于设置商品实体类的商品批准文号。
❾ setMemo()：该方法用于设置商品实体类的商品简介信息。
❿ setGysname()：该方法用于设置商品实体类的供应商名称。

（4）编写“入库”按钮的事件监听器 RkActionListener 类，该类必须实现 ActionListener 接口和接口中的 actionPerformed()方法，并在 actionPerformed()方法中获取界面中的商品表格数据，然后将这些数据封装到进货数据表的实体对象中，最后调用 Dao 类的 insertRukuInfo()方法在事务中保存进货单数据。关键代码如下：

例程 24　代码位置：光盘\TM\01\JXCManager\src\internalFrame\JinHuoDan.java

```
class RkActionListener implements ActionListener {                    //“入库”按钮的事件监听器
    public void actionPerformed(ActionEvent e) {
        stopTableCellEditing();                                        //结束表格中没有编写的单元
        clearEmptyRow();                                               //清除空行
        String hpzsStr = hpzs.getText();                               //货品总数
        String pzsStr = pzs.getText();                                 //品种数
        String jeStr = hjje.getText();                                 //合计金额
        String jsfsStr = jsfs.getSelectedItem().toString();            //结算方式
        String jsrStr = jsr.getText().trim();                          //经手人
        String czyStr = czy.getText();                                 //操作员
        String rkDate = jhsjDate.toLocaleString();                     //入库时间
        String ysjlStr = ysjl.getText().trim();                        //验收结论
        String id = piaoHao.getText();                                 //票号
        String gysName = gys.getSelectedItem().toString();             //供应商名称
        if (jsrStr == null || jsrStr.isEmpty()) {
            JOptionPane.showMessageDialog(JinHuoDan.this, "请填写经手人");
            return;
        }
        if (ysjlStr == null || ysjlStr.isEmpty()) {
            JOptionPane.showMessageDialog(JinHuoDan.this, "添写验收结论");
            return;
        }
        if (table.getRowCount() <= 0) {
            JOptionPane.showMessageDialog(JinHuoDan.this, "添加入库商品");
```

```
            return;
        }
        TbRukuMain ruMain = new TbRukuMain(id, pzsStr, jeStr, ysjlStr,
                gysName, rkDate, czyStr, jsrStr, jsfsStr);          //创建入库主表实体对象
        Set<TbRukuDetail> set = ruMain.getTabRukuDetails();         //获取入库主表的详细表集合
        int rows = table.getRowCount();
        for (int i = 0; i < rows; i++) {
            TbSpinfo spinfo = (TbSpinfo) table.getValueAt(i, 0);    //获取商品对象
            String djStr = (String) table.getValueAt(i, 6);         //获取商品的单价
            String slStr = (String) table.getValueAt(i, 7);         //获取商品的数量
            Double dj = Double.valueOf(djStr);
            Integer sl = Integer.valueOf(slStr);
            TbRukuDetail detail = new TbRukuDetail();               //创建商品详细表实体对象
            detail.setTabSpinfo(spinfo.getId());                    //初始化商品详细表对象
            detail.setTabRukuMain(ruMain.getRkId());
            detail.setDj(dj);
            detail.setSl(sl);
            set.add(detail);
        }
        boolean rs = Dao.insertRukuInfo(ruMain);                    //调用 insertRukuInfo()方法保存进货单
        if (rs) {
            JOptionPane.showMessageDialog(JinHuoDan.this, "入库完成");
            DefaultTableModel dftm = new DefaultTableModel();
            table.setModel(dftm);
            initTable();
            pzs.setText("0");
            hpzs.setText("0");
            hjje.setText("0");
        }
    }
}
```

1.9 查询统计模块设计

查询统计模块是进销存管理系统中不可缺少的重要组成部分，它主要包括基础信息、进货信息、销售信息、退货信息的查询和销售排行功能。

1.9.1 查询统计模块概述

企业进销存管理系统中的查询统计模块包括客户查询、商品查询、供应商查询、销售查询、销售退货查询、入库查询、入库退货查询和销售排行功能。由于本书的篇幅所限，本节将以销售查询功能为主，介绍查询统计模块对本系统的意义和实现的业务逻辑。

销售查询功能主要用于查询系统中的销售信息，其查询方式可以按照客户全称、销售票号进行匹配查询和模糊查询。另外，还可以指定销售日期查询。程序界面如图 1.29 所示。

图 1.29　销售查询界面

1.9.2　查询统计模块技术分析

查询统计模块主要以丰富的查询条件为主要技术，当查询一个商品销售或者退货等信息时，需要提供按客户全称、销售票号、退货票号、指定日期等多种查询条件和查询对象进行普通查询或者模糊查询。对于普通查询条件可以简单地使用 SQL 语句的“=”进行判断，但是模糊查询稍微复杂一些，需要使用 SQL 语句中的 LIKE 关键字。LIKE 关键字需要使用通配符在字符串内查找指定的模式，所以读者需要了解通配符及其含义。通配符的含义如表 1.9 所示。

表 1.9　LIKE 关键字中的通配符及其含义

通　配　符	说　　明
%	由 0 个或更多字符组成的任意字符串
_	任意单个字符
[ ]	用于指定范围，例如[A~F]，表示 A~F 范围内的任何单个字符
[^]	表示指定范围之外的，例如[^ A~F]，表示 A~F 范围以外的任何单个字符

☑　“%”通配符

“%”通配符能匹配 0 个或更多个字符的任意长度的字符串。

☑　“_”通配符

“_”通配符表示任意单个字符，该符号只能匹配一个字符，利用“_”符号可以作为通配符组成匹配模式进行查询。

☑　“[ ]”通配符

在模糊查询中可以使用“[ ]”通配符来查询一定范围内的数据。“[ ]”符号用于表示一定范围内的任意单个字符，它包括两端数据。

☑　“[^]”通配符

在模式查询中可以使用“[^]”通配符来查询不在指定范围内的数据。“[^]”符号用于表示不在某

范围内的任意单个字符，它包括两端数据。

1.9.3 销售查询的实现过程

销售查询使用的数据表：v_sellView

销售查询的开发步骤如下：

（1）创建 XiaoShouChaXun 类，用于实现本系统的销售查询功能界面和业务逻辑。界面中定义的主要控件如表 1.10 所示。

表 1.10 销售查询功能界面中的主要控件

控件类型	控件名称	主要属性设置	用途
JtextField	StartDate	设置文本内容为当年的 1 月 1 日	起始日期
	EndDate	设置文本内容为当前日期	截止日期
	Content	添加按键监听器，当按下 Enter 键时执行查询	查询内容
JcomboBox	Operation	添加“等于”和“包含”两项内容	条件方式
	Condition	添加“客户全称”和“销售票号”两项内容	查询条件
Jtable	Table	设置表头，取消表格列大小的自动调整	商品表格
Jbutton	queryButton	设置按钮文本为“查询”，设置动作监听器为 QueryActionListener 类的实例对象	查询
	showAllButton	设置按钮文本为“显示全部数据”，设置动作监听器为 ShowAllActionListener 类的实例对象	显示全部数据

（2）编写 updateTable()方法，用于更新表格数据。该方法必须接收一个 Iterator 迭代器对象，通过遍历该迭代器中的数据来初始化界面中的表格。关键代码如下：

例程 25 代码位置：光盘\TM\01\JXCManager\src\internalFrame\XiaoShouChaXun.java

```
private void updateTable(Iterator iterator) {                    //更新表格数据
    int rowCount=dftm.getRowCount();
    for(int i=0;i<rowCount;i++) {                                //清除原内容
        dftm.removeRow(0);
    }
    while(iterator.hasNext()) {                                  //更新表格数据
        Vector vector=new Vector();
        List view=(List) iterator.next();
        vector.addAll(view);
        dftm.addRow(vector);
    }
}
```

（3）创建 ShowAllActionListener 类，使该类实现 ActionListener 接口，并实现该接口的 actionPerformed()方法。该方法在用户单击“显示全部数据”按钮时，执行无条件的数据查询，也就是说，该按钮将读取数据库中所有的销售信息，并显示到表格中。关键代码如下：

例程 26　代码位置：光盘\TM\01\JXCManager\src\internalFrame\XiaoShouChaXun.java

```
class ShowAllActoinListener implements ActionListener {          //“显示全部数据”按钮的动作监听器
    public void actionPerformed(final ActionEvent e) {
        content.setText("");
        List list=Dao.findForList("select * from v_sellView");    //调用 findForList()方法执行查询
        Iterator iterator=list.iterator();
        updateTable(iterator);                                    //调用 updateTable()方法更新表格
    }
}
```

（4）创建“查询”按钮的事件监听器 QueryActionListener 类，该类必须实现 ActionListener 接口，并实现该接口的 actionPerformed()方法。在该方法中编写查询销售信息的业务逻辑，并将查询结果更新到表格控件中，其查询条件由 condition、operation 下拉列表框和一个 content 文本框组成。关键代码如下：

例程 27　代码位置：光盘\TM\01\JXCManager\src\internalFrame\XiaoShouChaXun.java

```
class QueryActionListener implements ActionListener {
    public void actionPerformed(final ActionEvent e) {
        boolean selDate = selectDate.isSelected();
        if(content.getText().equals("")) {
            JOptionPane.showMessageDialog(getContentPane(), "请输入查询内容！");
            return;
        }
        if(selDate) {
            if(startDate.getText()==null||startDate.getText().equals("")) {
                JOptionPane.showMessageDialog(getContentPane(), "请输入查询的开始日期！");
                return;
            }
            if(endDate.getText()==null||endDate.getText().equals("")) {
                JOptionPane.showMessageDialog(getContentPane(), "请输入查询的结束日期！");
                return;
            }
        }
        List list=null;
        String con = condition.getSelectedIndex() == 0                    //获取查询字段
                ? "khname "
                : "sellId ";
        int oper = operation.getSelectedIndex();                          //定义查询方式
        String opstr = oper == 0 ? "= " : "like ";
        String cont = content.getText();                                  //获取查询内容
        list = Dao.findForList("select * from v_sellView where "          //调用 findForList()方法查询数据
                + con
                + opstr
                + (oper == 0 ? "'"+cont+"'" : "'%" + cont + "%'")
                + (selDate ? " and xsdate>'" + startDate.getText()
                        + "' and xsdate<='" + endDate.getText()+" 23:59:59'" : ""));
        Iterator iterator = list.iterator();
        updateTable(iterator);                                            //调用 updateTable()方法更新表格
    }
}
```

1.10 库存管理模块设计

1.10.1 库存管理模块概述

企业进销存管理系统中的库存管理模块包括库存盘点和价格调整两个功能。由于本书的篇幅所限，本节将以价格调整功能为主，介绍库存管理模块对本系统的意义和实现的业务逻辑。

价格调整功能主要用于调整库存中指定商品的单价，当用户选择了指定的商品，价格调整功能的界面会显示该商品在库存中的单价、库存数量、库存金额、单位、产地等信息。程序界面如图 1.30 所示。用户可以修改商品价格并单击“确定”按钮，调整该商品在库存中的单价。

图 1.30　价格调整界面

1.10.2 库存管理模块技术分析

企业进销存管理系统中的库存管理模块包括库存盘点和价格调整两个功能，其中库存盘点涉及的技术比较简单，它将库存信息显示在表格中，由操作员输入盘点的商品数量，然后程序自动计算损益值。价格调整功能涉及下拉列表框的选择事件监听和事件处理技术，这在使用 Java Swing 技术进行程序开发的过程中非常重要。为防止用户的错误输入，程序界面经常需要将可枚举的输入内容封装在下拉列表框中，限制用户的输入。但是，要知晓下拉列表框的改变，还需要为下拉列表框添加相应的事件监听器。下面就来介绍相关的语法。

addItemListener()方法可以为下拉列表框添加 ItemListener 监听器。当更改下拉列表框中的选项时，将产生相应的事件，这个事件会被添加的 ItemListener 监听器捕获，并处理相应的业务逻辑。其语法格式如下：

```
public void addItemListener(ItemListener aListener)
```

aListener：要通知的 ItemListener 监听器。

1.10.3 价格调整的实现过程

价格调整使用的数据表：tb_kucun

价格调整的开发步骤如下：

（1）创建 JiaGeTiaoZheng 类，用于实现本系统的价格调整功能界面和业务逻辑。界面中定义的主要控件如表 1.11 所示。

表 1.11　价格调整功能界面中的主要控件

控 件 类 型	控 件 名 称	主要属性设置	用　　途
JTextField	KuCunJinE	取消编辑状态	库存金额
	KuCunShuLiang	取消编辑状态	库存数量
	DanJia	添加按键监听器，当输入改变时调用 updateJinE()方法更新库存金额	库存单价
JComboBox	ShangPinMingCheng	读取库存表数据并初始化该控件内容	商品名称
Jlabel	GuiGe	设置前景色为蓝色	规格
	ChanDi	设置前景色为蓝色	产地
	JianCheng	设置前景色为蓝色	简称
	BaoZhuang	设置前景色为蓝色	包装
	DanWei	设置前景色为蓝色	单位
Jbutton	OkButton	设置按钮文本为“确定”，设置动作监听器为 OkActionListener 类的实例对象	确定
	CloseButton	设置按钮文本为“关闭”，设置动作监听器为 CloseActionListener 类的实例对象	关闭

（2）编写 updateJinE()方法，用于更新库存金额。该方法将“单价”文本框的内容转换为 Double 类型，将“库存数量”文本框的内容转换为 Integer 类型，然后用它们的乘积更新“库存金额”文本框的内容。关键代码如下：

例程 28　代码位置：光盘\TM\01\JXCManager\src\internalFrame\JiaGeTiaoZheng.java

```
private void updateJinE() {                          //更新库存金额的方法
    Double dj = Double.valueOf(danJia.getText());
    Integer sl = Integer.valueOf(kuCunShuLiang.getText());
    kuCunJinE.setText((dj * sl) + "");
}
```

（3）创建 ItemActionListener 类，它必须实现 ItemListener 接口和接口中的 itemStateChanged()方法，成为下拉列表框的事件监听器。当改变界面中选择的商品时，相应的 ItemEvent 事件会通知该监听器处理业务逻辑，也就是根据选择的商品名称更新其他的控件内容。关键代码如下：

例程 29　代码位置：光盘\TM\01\JXCManager\src\internalFrame\JiaGeTiaoZheng.java

```
❶  class ItemActionListener implements ItemListener {                    //商品选择事件监听器
❷      public void ItemStateChanged(
❸                                          final ItemEvent e) {
           Object selectedItem = shangPinMingCheng.getSelectedItem();     //获取选择的商品对象
           if (selectedItem == null)
               return;
           Item item = (Item) selectedItem;
           kcInfo = Dao.getKucun(item);                                   //调用 getKucun()方法
           if(kcInfo.getId()==null)
               return;
           int dj, sl;
           dj = kcInfo.getDj().intValue();
```

```
                sl = kcInfo.getKcsl().intValue();
                chanDi.setText(kcInfo.getCd());                    //更新界面控件的内容
                jianCheng.setText(kcInfo.getJc());
                baoZhuang.setText(kcInfo.getBz());
                danWei.setText(kcInfo.getDw());
                danJia.setText(kcInfo.getDj() + "");
                kuCunShuLiang.setText(kcInfo.getKcsl() + "");
                kuCunJinE.setText(dj * sl + "");
                guiGe.setText(kcInfo.getGg());
            }
        }
```

代码贴士

❶ ItemListener：下拉列表框的事件监听器必须实现的分接口。

❷ ItemStateChanged()：当下拉列表框的选中项发生改变时将触发该方法。

❸ ItemEvent：这是选项事件类，在用户更改带有多项选择内容的组件选项时，将产生该事件。例如下拉选择框组件。

（4）创建 OkActionListener 类，它必须实现 ActionListener 接口和接口中的 actionPerformed()方法，在这个方法中获取新的库存商品价格，然后调用 Dao 类的 updateKucunDj()方法更新库存价格。关键代码如下：

例程 30 代码位置：光盘\TM\01\JXCManager\src\internalFrame\JiaGeTiaoZheng.java

```
class OkActionListener implements ActionListener {
    public void actionPerformed(final ActionEvent e) {
        kcInfo.setDj(Double.valueOf(danJia.getText()));
        kcInfo.setKcsl(Integer.valueOf(kuCunShuLiang.getText()));
        int rs = Dao.updateKucunDj(kcInfo);
        if (rs > 0)
            JOptionPane.showMessageDialog(getContentPane(), "价格调整完毕。",
                    kcInfo.getSpname() + "价格调整",
                    JOptionPane.QUESTION_MESSAGE);
    }
}
```

1.10.4 单元测试

在价格调整界面中输入单价时，如果输入“1l33”，程序将抛出 NumberFormatException 异常，如图 1.31 所示。这是因为输入单价的数字格式不对，注意输入值“1l33”的第二个“l”字符并不是数字，而是英文字母 L 的小写形式，字母当然不能用作数字，所以产生了这个错误，导致程序无法执行价格调整。

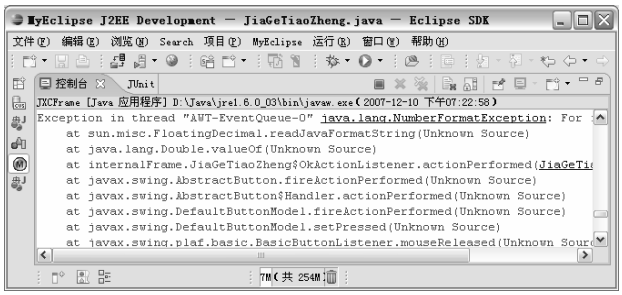


图 1.31 非数字单价产生的错误

解决这一问题的方法是在执行价格调整前对输入的单价进行数字格式验证。但非要等

操作员输入单价后，再验证输入单价的正确与否吗？如果利用按键监听器，监听“单价”文本框中的每一次按键，当按键是数字时，继续接收输入；反之，当按键不是数字或小数点时（那它就应该是字母或其他的什么，反正不是数字）就取消本次按键的输入。这样在用户输入时，就能够有效地屏蔽非数字格式的输入，它比之前的数字格式验证更有效。关键代码如下：

例程 31　代码位置：光盘\TM\01\JXCManager\src\internalFrame\JiaGeTiaoZheng.java

```
danJia.addKeyListener(new KeyAdapter() {                    //添加按键监听器
    public void keyTyped(KeyEvent e) {
        String numStr = "0123456789." + (char) 8;//数字格式的字符串，其中(char)8 是回退键用于删除字符
        if (numStr.indexOf(e.getKeyChar()) < 0)             //如果按键字符不在数字格式字符串中
            e.consume();                                    //销毁按键对象
        else                                                //否则
            updateJinE();                                   //更新库存金额
    }
});
```

1.11　系统打包发布

Java 应用程序可以打包成 JAR 文件，JAR 文件是一个简单的 ZIP 格式的文件，它包含程序中的类文件和执行程序的其他资源文件。在程序发布之前，需要将所有编译好的 Java 文件封装到一个程序打包文件中，然后将这个程序的打包文件提交给客户使用。一旦程序打包后，就可以使用简单的命令来执行它。另外，如果配置好 Java 环境或使用 JDK 的安装程序构建 Java 环境，那么就可以像运行本地可执行文件一样去执行 JAR 文件。本节将介绍如何使用 Eclipse 开发工具将程序打包成 JAR 文件。

（1）创建描述文件。JAR 文件需要一个描述文件，该文件以 MANIFEST.MF 命名，它描述了 JAR 的配置信息，如指定主类名称、类路径等。程序代码如下：

```
❶ Manifest-Version: 1.0                                          //指定描述文件的版本
❷ Main-Class: com.lzw.JXCFrame                                   //指定程序主类
❸ Class-Path: . lib\msbase.jar lib\mssqlserver.jar lib\msutil.jar //配置类路径
❹                                                                //添加空行结尾
```

代码贴士

❶ 描述文件的版本号是每个描述文件的基本信息。

❷ Main-Class 用于指定程序执行的主类。

❸ Class-Path 用于指定程序执行的类路径，多个路径使用空格符号分割。

❹ 在描述文件的结尾插入一个空行，这代表描述文件的结束。

注意　在“:”符号和后面的定义值之间一定要有一个空格作分隔符，否则程序会因为无法识别而导致程序出错。

（2）在 Eclipse 的资源包管理器中右击项目的 src 文件夹，在弹出的快捷菜单中选择“导出”命令。

（3）在弹出的“导出”对话框中单击 Java/“JAR 文件”子节点，单击“下一步”按钮。

（4）在弹出的“JAR 导出”对话框中选择要导出的文件夹，本系统的程序代码都在 src 文件夹中，在步骤（2）中是右击 src 文件夹启动导出功能的，在该对话框中已经默认选择 src 文件夹中的所有内容，包括子文件夹。然后在“JAR 文件”下拉列表框中输入生成的 JAR 文件名和路径，如图 1.32 所示。单击两次“下一步”按钮。

图 1.32 “JAR 导出”对话框

（5）在弹出的对话框中选中“从工作空间中使用现有清单”单选按钮，在“清单文件”文本框的右侧单击“浏览”按钮，选择步骤（1）建立的清单文件 MANIFEST.MF，单击“完成”按钮。

（6）现在 JAR 文件已经创建并保存在 C 盘 product 文件夹中，程序的清单描述文件中指定连接 SQL Server 2000 数据库的 JDBC 驱动包放在 lib 文件夹中，所以，必须在 product 文件夹中创建 lib 文件夹，然后将相应的类包复制到 lib 文件夹中，最后将本系统所用到的 res 图片资源文件夹复制到 product 文件夹中，就可以双击 JXCManager.jar 文件运行程序了。

1.12 开发技巧与难点分析

本系统使用的是 MDI 窗体模式开发的程序界面，它使用一个主窗体包含多个子窗体，子窗体只能在主窗体规定的范围内移动。这些子窗体由导航面板上的按钮调用，这些按钮需要添加事件监听器，在单击某个按钮时，由事件监听器创建并初始化相应的子窗体，然后显示该子窗体。

如果为每个按钮创建新的事件监听器对象，那至少需要 20 个事件监听器类，因为导航面板上定义的按钮总数和子窗体的数量是对应的，而子窗体的数量正好是 20 个，所以需要定义相应数量的按钮和事件监听器，这些繁琐的工作会占用大量的程序开发时间，影响工程进度。

从不同的按钮监听器所实现的业务逻辑中不难发现，它们所完成的工作基本相同，都是创建并初始化子窗体，然后显示它们。如果它们能够使用同一个事件监听器类就可以实现代码重用，同时也节省了代码工作量，提高程序开发速度。

这样的开发思路存在很多优点，但是实现起来并不容易，子窗体的名称、类名都可以获取，但是如何根据指定的类名去创建子窗体对象呢？

Java 的反射功能为这个思路提供了可行性。在 java.lang.reflect 包中有 Field、Method 和 Constructor 3 个类，分别描述类的字段、方法和构造方法。这里需要的就是类的构造方法，只有调用类的构造方法才能创建该类的实例对象。可以通过 Class 类的 getConstructor()方法获取 Constructor 类的实例对象，然后调用该对象的 newInstance()方法创建类的实例对象。关键代码如下：

例程 32　代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
try {
❶    Class fClass = Class.forName("internalFrame." + frameName);
❷    Constructor constructor = fClass.getConstructor(null);
❸    jf = (JInternalFrame) constructor.newInstance(null);
     ifs.put(frameName, jf);
} catch (Exception e) {
     e.printStackTrace();
}
```

代码贴士

❶ 调用 Class 类的 forName()方法加载指定的 Java 类，该方法将返回该类的 Class 实例对象。

❷ 调用指定类的 getConstructor()方法获取指定的构造器。方法中使用 null 作参数，是调用该类的默认构造器，因为类的默认构造器没有任何参数。

❸ 调用构造器的 newInstance()方法，同样传递参数 null，这样就可以调用默认的构造方法创建子窗体对象。

1.13　使用 PowerDesigner 逆向生成数据库 E-R 图

在开发一个新的程序时，为提高开发速度，经常修改现有的与将要开发的程序相类似的旧程序。同样，功能相似的程序，它们的数据库也基本相似，甚至完全相同，那么可以直接使用原有的数据库从而节省数据库设计的时间和工作量。而要分析一个数据库的数据结构和连接关系，E-R 图是最好的数据库资料。但是原有的数据库也许是多年以前的，或者是借鉴同事的，资料不一定完整，也不一定存在 E-R 图。这就给数据库分析带来了很大的不便。

如果能够使用相应的设计工具将数据库的结构和关系抽象成 E-R 图，就可以为系统分析员提供相应的数据库资料，从而分析或修改原有数据库。本节将介绍如何使用 PowerDesigner 工具实现数据库 E-R 图的逆向生成。

（1）在开始逆向生成 E-R 图之前，需要为指定的数据库创建 ODBC 数据源。以 Windows 2003 操作系统为例，选择“开始”/“运行”命令，在打开的“运行”对话框中输入“odbcad32.exe”，单击“确定”按钮，启动数据源管理器。

（2）在“ODBC 数据源管理器”对话框中单击“添加”按钮。

（3）在弹出的“创建新数据源”对话框中选择 SQL Server 选项，单击“完成”按钮，如图 1.33 所示。

（4）在弹出的“创建到 SQL Server 的新数据源”对话框的“名称”文本框中输入新建数据源的名称，如 db_jxcOdbc。在“描述”文本框中可以输入该数据源的描述信息，因为数据源的名称经常使用单词的缩写形式，随着时间的流逝很容易忘记其含义，如果搭配相应的描述信息，会使该数据源的含义更明确。在“服务器”下拉列表框中输入“localhost”，单击“下一步”按钮，如图 1.34 所示。

（5）在弹出的对话框中选中“使用用户输入登录 ID 和密码的 SQL Server 验证”单选按钮，然后选中“连接 SQL Server 以获得其他配置选项的默认设置”复选框，在“登录 ID”文本框中输入访问数据库的用户名，例如 sa，在“密码”文本框中输入访问数据库的密码，单击“下一步”按钮，如图 1.35 所示。

图 1.33 “创建新数据源”对话框

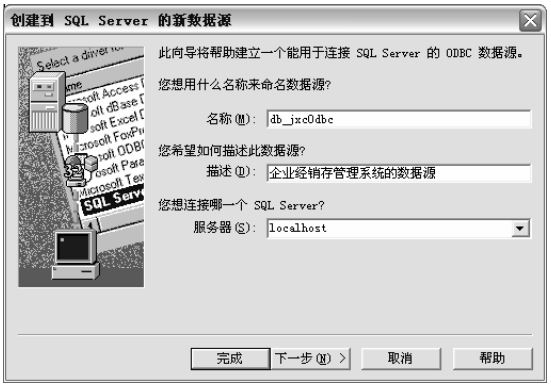


图 1.34 “创建到 SQL Server 的新数据源”对话框（1）

（6）在弹出的对话框的“更改默认的数据库”下拉列表框中选择操作的数据库，例如本系统的 db_JXC，单击“下一步”按钮，然后在弹出的对话框中单击“完成”按钮创建数据源。

（7）建立完数据源后，启动 PowerDesigner。重复 1.4.3 节中的步骤（1）建立一个空的物理数据模型。

（8）选择 Database/Reverse Engineer Database 命令，在弹出的对话框中选中 Using a data source 单选按钮，单击右侧的按钮选择刚刚建立的 db_jxcOdbc 数据源，单击“确定”按钮，如图 1.36 所示。

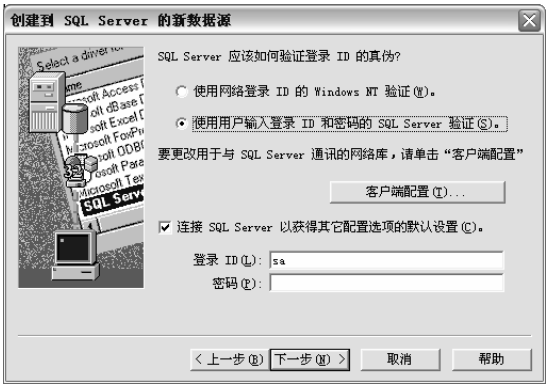


图 1.35 “创建到 SQL Server 的新数据源”对话框（2）

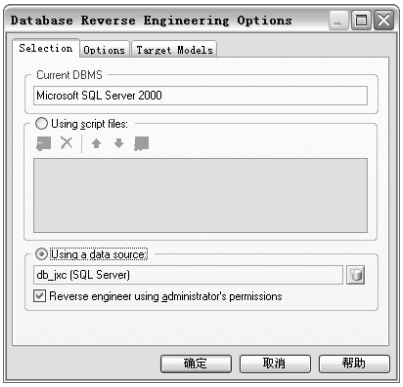


图 1.36 Database Reverse Engineering Options 对话框

（9）在弹出的对话框中选择需要生成 E-R 图的数据表、视图、系统表等，单击“确定”按钮生成数据库的 E-R 图。本系统的数据库 E-R 图在 1.4.3 节中已经介绍过，效果如图 1.16 所示。

1.14 本 章 小 结

本章运用软件工程的设计思想，通过一个完整的企业进销存管理系统为读者详细讲解了一个系统的开发流程。通过本章的学习，读者可以了解 Java 应用程序的开发流程及 Java Swing 的窗体设计、事件监听等技术。另外，本章还介绍了 PowerDesigner 工具的数据库建模和逆向生成数据库 E-R 图以及 Java 应用程序的系统打包等技术，希望对读者日后的程序开发有所帮助。

第 2 章

企业内部通信系统

（Swing+JavaDB 实现）

近年来，各种企业内部通信系统得到了飞速发展。它可以不用连接 Internet，直接在局域网内实现信息通信、工作交流、提交计划等业务。这种通信系统广泛应用于中、小型企业的内部通信，可以大大提高职工的工作效率，在方便企业内部职工交流的同时，也创造了一个安静的工作环境，是现代企业不可缺少的辅助工具。

本章将介绍如何使用 Java Swing 技术和 Java 6.0 新增的 JavaDB 数据库开发跨平台的应用程序。

通过阅读本章，可以学习到：

- JavaDB 数据库的应用
- 如何使用 Java 6.0 的系统托盘
- 如何实现多点通信
- Java 如何调用其他程序
- 使用 UDP 通信协议

2.1 开 发 背 景

×××有限公司是一个中型的私营企业，企业内部的员工经常需要沟通和交流工作中的常见问题，频繁地使用电话会影响其他工作人员，另外，在实验室、档案室等需要安静气氛的环境中，使用电话沟通更不方便。为了便于职工之间的交流和工作信息的传递，企业内部通信系统的开发就显得迫切而重要。于是，该公司决定根据企业的内部结构，开发一个符合本企业工作流程的通信系统。它可以帮助企业快速搭建内部即时通信结构，大幅度提高企业的工作效率，使上级与下级之间的交流更方便。

2.2 系 统 分 析

2.2.1 需求分析

通过与×××有限公司的沟通和需求分析，要求企业内部通信系统具有以下功能：

- ☑ 操作简单，界面友好。
- ☑ 规范、完善的基础信息设置。
- ☑ 支持网络通信。
- ☑ 支持系统托盘和程序最小化功能，避免影响其他工作。
- ☑ 使用独立的本地数据库。
- ☑ 自动搜索和手动添加网络内的通信用户。
- ☑ 提供用户的更名、删除等操作。
- ☑ 支持自动更新，始终保持程序的最高版本。
- ☑ 支持系统公告。

2.2.2 可行性分析

根据《GB8567－88 计算机软件产品开发文件编制指南》中可行性分析的要求，制定的可行性研究报告如下。

1. 引言

☑ 编写目的

以文件的形式给企业的决策层提供项目实施的参考依据，其中包括项目存在的风险、项目需要的投资和能够收获的最大效益。

☑ 背景

×××有限公司是一家中型的私有企业，为了提高企业的工作效率、实现信息化管理，公司决定开发企业内部通信系统。

2．可行性研究的前提

☑　要求

企业内部通信系统必须提供网络通信功能，在通信过程中禁止使用聊天表情、文件传送等功能，避免资料外泄，或因发送错误而导致上级资料的丢失以及其他损失。最重要的是必须适应任何操作系统，即实现跨平台技术，因为由于企业内部的工作需要，工作环境中使用了多个操作系统来完成不同的工作。另外，系统不需要使用服务器中转和记录通信内容，可以独立完成通信任务，排除职工对领导监视工作进度等逆反心理。

☑　目标

企业内部通信系统的目标是实现企业的信息化通信，提高企业通信能力，提高任务理解和执行能力，减少不必要的人员流动和资金损耗，以最快的速度提升企业的市场竞争力。

☑　条件、假定和限制

为实现企业的信息化通信，必须对操作人员进行培训，需要花费部分时间和精力来完成。为不影响企业的正常运行，企业内部通信系统必须在两个月的时间内交付用户使用。

系统分析人员需要两天内到位，用户需要 3 天时间确认需求分析文档。去除其中可能出现的问题，例如用户可能临时有事，占用 4 天时间确认需求分析。那么程序开发人员需要在 1 个月零 19 天的时间内进行系统设计、程序编码、系统测试、程序调试和网站部署工作。其间，还包括了员工每周的休息时间。

☑　评价尺度

根据用户的要求，项目主要以企业通信功能为主，对于通信信息仅提供本次系统启动后的通信内容。由于职工人数过多，而公司在楼内公告板上的公告信息，难以及时通知每位职工，系统中公告功能要及时地通知所有员工最新的公告内容。

3．投资及效益分析

☑　支出

根据系统的规模及项目的开发周期（两个月），公司决定投入 4 个人。为此，公司将直接支付 3 万元的工资及各种福利待遇。在项目安装及调试阶段，用户培训、员工出差等费用支出需要 1 万元。在项目维护阶段预计需要投入 2 万元的资金。累计项目投入需要 6 万元资金。

☑　收益

用户提供项目资金 12 万元。对于项目运行后进行的改动，采取协商的原则根据改动规模额外提供资金。因此从投资与收益的效益比上，公司可以获得 6 万元的利润。

项目完成后，会给公司提供资源储备，包括技术、经验的积累，其后再开发类似的项目时，可以极大地缩短项目开发周期。

4．结论

根据上面的分析，在技术上不会存在问题，因此项目延期的可能性很小。在效益上公司投入 4 个人、两个月的时间获利 6 万元，效益比较可观。在公司今后发展上可以储备网站开发的经验和资源。因此认为该项目可以开发。

2.2.3 编写项目计划书

根据《GB8567－88 计算机软件产品开发文件编制指南》中的项目开发计划要求，结合单位实际情况，设计项目计划书如下。

1．引言

☑ 编写目的

为了保证项目开发人员按时、保质地完成预定目标，更好地了解项目实际情况，按照合理的顺序开展工作，现以书面的形式将项目开发生命周期中的项目任务范围、团队组织结构、团队成员的工作责任、团队内外沟通协作方式、开发进度、检查项目工作等内容描述出来，作为项目相关人员之间的共识和约定及项目生命周期内的所有项目活动的行动基础。

☑ 背景

企业内部通信系统是由×××有限公司委托本公司开发的通信系统，主要功能是实现企业内部通信和企业公告，项目周期两个月。项目背景规划如表 2.1 所示。

表 2.1 项目背景规划

项 目 名 称	项目委托单位	任务提出者	项目承担部门
企业内部通信系统	×××有限公司	陈经理	策划部门 研发部门 测试部门

2．概述

☑ 项目目标

项目目标应当符合 SMART 原则，把项目要完成的工作用清晰的语言描述出来。企业内部通信系统的项目目标如下：

企业内部通信系统的主要目的是实现企业的内部通信，主要的业务就是信息交流和企业公告。项目实施后，能够避免无谓的支出、合理控制任务计划、减少资金占用并提升企业工作效率。整个项目需要在两个月的时间内交付用户使用。

☑ 产品目标

时间就是金钱，效率就是生命。项目实施后，企业内部通信系统能够为企业节省大量人力资源，减少管理费用，从而间接为企业节约了成本，并提高了企业的竞争力。

☑ 应交付成果

- 在项目开发完后，交付内容有企业内部通信系统的源程序、系统的数据库文件、系统使用说明书。
- 将开发的企业内部通信系统打包并安装到企业的网络计算机中。
- 企业内部通信系统交付用户后，进行系统无偿维护和服务 6 个月，超过 6 个月进行系统有偿维护与服务。

☑　项目开发环境

操作系统为 Windows 7、Windows XP 或 Windows 2003，使用集成开发工具 Eclipse，数据库采用 SQL Server 2000，项目运行环境为 JDK 7。

☑　项目验收方式与依据

项目验收分为内部验收和外部验收两种方式。在项目开发完成后，首先进行内部验收，由测试人员根据用户需求和项目目标进行验收。项目在通过内部验收后，交给客户进行验收，验收的主要依据为需求规格说明书。

3．项目团队组织

☑　组织结构

为了完成企业内部通信系统的项目开发，公司组建了一个临时的项目团队，由公司项目经理、软件工程师、美工设计师和测试人员构成，如图 2.1 所示。

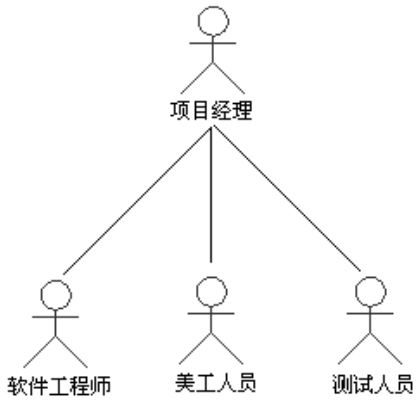


图 2.1　项目团队组织结构图

☑　人员分工

为了明确项目团队中每个人的任务分工，现制定人员分工表如表 2.2 所示。

表 2.2　人员分工表

姓　名	技 术 水 平	所 属 部 门	角　色	工 作 描 述
李××	MBA	策划部门	项目经理	负责项目的前期分析、策划、项目开发进度的跟踪、项目质量的检查
钟××	高级软件工程师	软件研发部	软件工程师	负责软件分析、设计与编码
尉××	中级美术设计师	美术设计部	美工人员	负责界面的设计和美工
赵××	高级软件工程师	测试部门	测试人员	负责软件测试与评定

2.3　系 统 设 计

2.3.1　系统目标

根据企业对内部通信系统的要求，本系统可以实现以下目标：

☑　操作简单方便，界面简洁美观。
☑　更方便访问企业公共资源。
☑　及时显示企业公共信息。
☑　在通信窗口显示对方 IP 信息。
☑　局域网内用户自动搜索。
☑　系统运行稳定、安全可靠。

2.3.2　系统功能结构

企业内部通信系统的功能结构如图 2.2 所示。

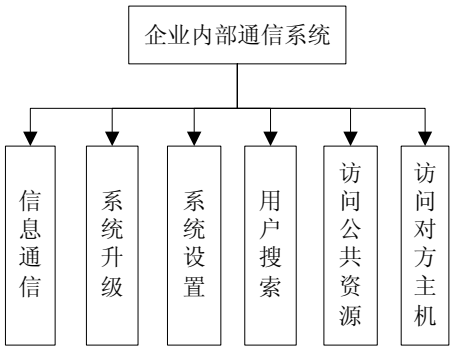


图 2.2　企业内部通信系统的功能结构

2.3.3 数据库设计

1．数据库分析

本系统是一个桌面应用程序，它可以直接在本地计算机上运行，而不需要像 Web 应用那样部署到指定的服务器中，所以企业内部通信系统的数据库应该随系统存在，即数据库和企业内部通信系统在同一个计算机中，将数据库和应用程序捆绑在一起，可以节省开销，提升系统安全性。本系统采用 JavaDB 数据库。其数据库运行环境如下。

☑ 硬件平台
- CPU：P4 1.6GHz。
- 内存：128MB 以上。
- 硬盘空间：100MB。

☑ 软件平台
- 操作系统：Windows 2003 以上。
- 数据库：JavaDB。
- Java 虚拟机：JDK 6.0 以上。

2．企业内部通信系统的 E-R 图

企业内部通信系统包含用户和窗体位置两个实体，这两个实体分别用于记录用户信息和通信窗体的当前位置。

☑ 用户实体

用户实体是企业内部通信系统的通信用户，它记载了系统搜索或添加的所有用户信息，主要包括用户 IP 地址、主机名称、用户名称、提示信息和头像信息，如图 2.3 所示。

☑ 窗体位置实体

窗体位置实体是窗体的定位参数，它将记录窗体最后的移动位置、窗体大小等信息，主要包括窗体位置的 X 坐标和 Y 坐标、窗体的宽度及高度，如图 2.4 所示。

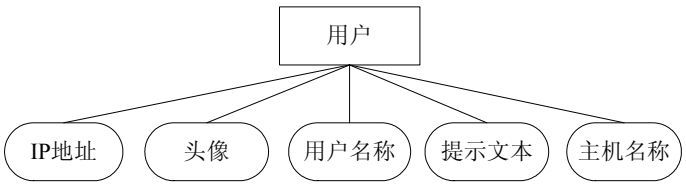


图 2.3 用户实体 E-R 图

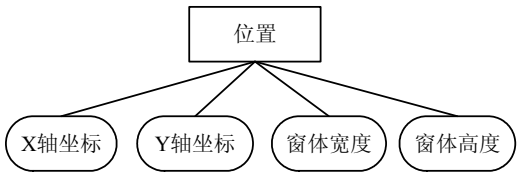


图 2.4 窗体位置实体 E-R 图

3．数据库逻辑结构设计

在本系统中创建了一个数据库 db_EQ，一共包含了两个数据表，下面分别介绍这两个数据表的逻辑结构。

☑　tb_users（用户信息表）

用户信息表主要用来保存企业内的通信用户，即职工信息。表 tb_users 的结构如表 2.3 所示。

表 2.3　表 tb_users 的结构

字　段　名	数 据 类 型	是 否 为 空	是 否 主 键	默　认　值	描　　述
ip	Varchar(16)	No	Yes		用户 IP 地址
host	Varchar(30)	Yes	No	NULL	主机名称
name	Varchar(20)	Yes	No	NULL	姓名
tooltip	Varchar(50)	Yes	No	NULL	提示文本
icon	Varchar(50)	Yes	No	NULL	头像

☑　tb_location（窗体位置信息表）

窗体位置信息表主要用来保存通信窗体的位置和窗体大小。表 tb_ location 的结构如表 2.4 所示。

表 2.4　表 tb_ location 的结构

字　段　名	数 据 类 型	是 否 为 空	是 否 主 键	默　认　值	描　　述
xLocation	Int	Yes	No	NULL	X 轴坐标
yLocation	Int	Yes	No	NULL	Y 轴坐标
width	Int	Yes	No	NULL	窗体宽度
heigth	Int	Yes	No	NULL	窗体高度

2.3.4　系统预览

企业内部通信系统由多个程序界面组成，下面仅列出几个典型界面的预览，其他界面参见光盘中的源程序。

企业内部通信系统的主界面如图 2.5 所示，该界面包含调用所有功能模块的控件。通信窗体界面如图 2.6 所示，该界面用于发送和接收通信信息；另外，还可以在对方未开启企业通信系统的情况下，向对方发送信使信息。

图 2.5　主窗体（光盘\…\EQ.java）

图 2.6　通信窗体界面（光盘\…\frame\TelFrame.java）

系统工具界面如图 2.7 所示，该界面主要用于更换界面外观、搜索用户和系统更新。系统设置界面如图 2.8 所示，该界面用于设置系统路径、系统登录、IP 搜索范围。

由于路径太长，因此省略了部分路径，省略的路径是“TM\02\EQ\src\com\lzw”。

2.3.5 文件夹组织结构

在进行系统开发前，需要规划文件夹组织结构，即建立多个文件夹，对各个功能模块进行划分，实现统一管理。这样做的好处为易于开发、管理和维护。本系统的文件夹组织结构如图 2.9 所示。

图 2.7 系统工具界面（光盘\…\EQ.java）

图 2.8 系统设置界面（光盘\…\EQ.java）

图 2.9 文件夹组织结构

2.4 主窗体设计

主窗体界面也是企业内部通信系统的用户列表，它由用户列表、公告提示、系统选项卡等组成。其中系统选项卡用于切换不同管理界面，包括系统工具和系统设计界面。主窗体的运行效果如图 2.10 所示。

图 2.10 主窗体界面

2.4.1 创建主窗体

创建主窗体的步骤如下：

（1）创建 JXCFrame 类，在类中创建窗体对象，为窗体添加选项卡面板，并添加用户列表、系统工具、系统设置 3 个选项卡和状态栏标签、公告按钮等属性。关键代码如下：

例程 01　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
public class EQ extends Dialog {
    private JTextField ipEndTField;                                          //“起始 IP”文本框
    private JTextField ipStartTField;                                        //“终止 IP”文本框
    private JTextField userNameTField;                                       //“用户名”文本框
    private JPasswordField passwordTField;                                   //“密码”文本框
    private JTextField placardPathTField;                                    //“系统公告路径”文本框
    private JTextField updatePathTField;                                     //“程序升级路径”文本框
    private JTextField pubPathTField;                                        //“公共程序路径”文本框
    public static EQ frame = null;                                           //主窗体对象
    private ChatTree chatTree;                                               //用户树对象
    private JPopupMenu popupMenu;                                            //弹出菜单对象
    private JTabbedPane tabbedPane;                                          //选项卡
    private JToggleButton searchUserButton;                                  //搜索用户按钮
    private JProgressBar progressBar;                                        //搜索进度条
    private JList faceList;                                                  //搜索列表
    private JButton selectInterfaceOKButton;                                 //界面选择按钮
    private DatagramSocket ss;                                               //数据通信包
    private final JLabel stateLabel;                                         //状态栏标签
    private static String user_dir;                                          //用户当前文件夹
    private static File localFile;                                           //本地文件
    private static File netFile;                                             //升级路径上的网络文件
    private String netFilePath;                                              //升级文件路径
    private JButton messageAlertButton;                                      //公告信息按钮
    private Stack<String> messageStack;                                      //公告信息栈
    private ImageIcon messageAlertIcon;                                      //公告信息图标
    private ImageIcon messageAlertNullIcon;                                  //公告信息空图标
    private Rectangle location;                                              //窗体位置
    public static TrayIcon trayicon;                                         //系统托盘
    private Dao dao;                                                         //数据库操作类
    public final static Preferences preferences = Preferences.systemRoot();; //首选项
    private JButton userInfoButton;                                          //本地用户按钮
…//省略部分代码
}
```

（2）在构造方法中初始化窗体上的控件、数据库操作类、首选项对象等属性，另外还要为窗体和公告信息按钮添加事件监听器等。关键代码如下：

例程 02　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
public EQ() {
    super(new Frame());
    frame = this;                                                  //初始化窗体对象
    dao = Dao.getDao();                                            //初始化 dao 对象
    location = dao.getLocation();                                  //初始化窗体位置对象
    setTitle("EQ 通讯");                                           //设置窗体标题
    setBounds(location);                                           //设置窗体位置
    progressBar = new JProgressBar();                              //初始化搜索用户进度条
    progressBar.setBorder(new BevelBorder(BevelBorder.LOWERED));
```

```
        tabbedPane = new JTabbedPane();                                    //初始化选项卡
        popupMenu = new JPopupMenu();                                      //初始化弹出菜单
        chatTree = new ChatTree(this);
        user_dir = System.getProperty("user.dir");                         //用户当前路径
        localFile = new File(user_dir + File.separator + "EQ.jar");        //本地 EQ 文件
        stateLabel = new JLabel();                                         //状态栏标签
        addWindowListener(new FrameWindowListener());                      //添加窗体监视器
        {                                                                  //初始化公告信息按钮
            messageAlertIcon = new ImageIcon(EQ.class.getResource("/image/messageAlert.gif"));
            messageAlertNullIcon = new ImageIcon(EQ.class.getResource("/image/messageAlertNull20.gif"));
            messageStack = new Stack<String>();
            messageAlertButton = new JButton();
            messageAlertButton.setHorizontalAlignment(SwingConstants.RIGHT);
            messageAlertButton.setContentAreaFilled(false);
            final JPanel BannerPanel = new JPanel();
            BannerPanel.setLayout(new BorderLayout());
            add(BannerPanel, BorderLayout.NORTH);
            userInfoButton = new JButton();
            BannerPanel.add(userInfoButton, BorderLayout.WEST);
            userInfoButton.setMargin(new Insets(0, 0, 0, 10));
            initUserInfoButton();                                          //初始化本地用户头像按钮
            BannerPanel.add(messageAlertButton, BorderLayout.CENTER);
            messageAlertButton.addActionListener(new ActionListener() {    //添加公告信息按钮监听器
                public void actionPerformed(final ActionEvent e) {
                    if (!messageStack.empty()) {
                        showMessageDialog(messageStack.pop());             //显示公告信息
                    }
                }
            });
            messageAlertButton.setIcon(messageAlertIcon);
            showMessageBar();
        }
        add(tabbedPane, BorderLayout.CENTER);                              //初始化选项卡面板
❶       tabbedPane.setTabPlacement(SwingConstants.LEFT);
        ImageIcon userTicon = new ImageIcon(EQ.class
                .getResource("/image/tabIcon/tabLeft.PNG"));               //创建用户列表图标
        tabbedPane.addTab(
❷               null,
❸               userTicon,
❹               createUserList(),
❺               "用户列表");                                               //添加用户列表选项卡
        ImageIcon sysOTicon = new ImageIcon(EQ.class
                .getResource("/image/tabIcon/tabLeft2.PNG"));              //创建系统操作图标
        tabbedPane.addTab(null, sysOTicon, createSysToolPanel(), "系统操作"); //添加系统操作选项卡
        ImageIcon sysSTicon = new ImageIcon(EQ.class
                .getResource("/image/tabIcon/tabLeft3.png"));              //创建系统设置图标
        tabbedPane.addTab(null, sysSTicon, createSysSetPanel(), "系统设置");  //添加系统设置选项卡
        setAlwaysOnTop(true);                                              //使窗体显示在最顶端
    }
```

代码贴士

❶ setTabPlacement()方法用于设置选项卡面板的显示方向，参数 SwingConstants.LEFT 用于设置选项卡显示在左侧。
❷ 该参数是选项卡的文本内容，因为这里使用图标，所以该参数设置为 null。
❸ 该参数是选项卡的图标内容。
❹ 该参数是选项卡存放的控件，这里调用 createUserList()方法创建了用户列表面板。
❺ 该参数用于设置选项卡的提示文本，当光标停留在选项卡上，将显示该参数内容。

（3）初始化 Socket 服务器，指定端口使用 1111，如果初始化失败，提示用户服务器端口被占用，或者是本软件已经运行，并退出程序。该步骤很关键，它用于接收其他用户发送的通信信息，如果启动失败将无法接收信息，所以必须退出系统。关键代码如下：

例程 03　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
try {                                                          //启动通信服务端口
    ss = new DatagramSocket(1111);                             //初始化服务器
} catch (SocketException e2) {
    if (e2.getMessage().startsWith("Address already in use"))  //如果异常提示该信息
        showMessageDialog("服务端口被占用，或者本软件已经运行。");  //显示提示信息
    System.exit(0);                                            //退出系统
}
```

（4）编写 checkPlacard()方法，用于检测系统公告信息，当公告路径中存在公告信息时，该方法将从公告文件中获取完整信息，然后调用 pushMessage()方法将公告信息压入公告信息栈中。关键代码如下：

例程 04　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
private void checkPlacard() {                                  //检测公告信息方法
    String placardDir = preferences.get("placardPath", null);  //获取公告路径
    if (placardDir == null) {                                  //如果没有设置公告路径
        pushMessage("未设置公告路径");                          //提示设置信息
        return;
    }
    File placard = new File(placardDir);                       //创建公告文件对象
    try {
        if (placard.exists() && placard.isFile()) {
            StringBuilder placardStr = new StringBuilder();
            Scanner sc = new Scanner(new FileInputStream(placard));
            while (sc.hasNextLine()) {                         //读取公告文件内容
                placardStr.append(sc.nextLine());
            }
            pushMessage(placardStr.toString());                //堆压公告信息
        }
    } catch (FileNotFoundException e) {
        pushMessage("公告路径错误，或公告文件不存在");          //提示错误信息
    }
}
```

（5）编写 initUserInfoButton()方法，用于初始化本地用户信息，并在主窗体左上角显示本地用户的头像和名称，在用户更改本地用户名称时，它会同步更新。关键代码如下：

例程 05 代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
private void initUserInfoButton() {                                    //初始化用户信息按钮
    try {
        String ip = InetAddress.getLocalHost().getHostAddress();       //获取本地 IP
        User user = dao.getUser(ip);                                   //从数据库获取用户对象
❶       userInfoButton.setIcon(user.getIconImg());
❷       userInfoButton.setText(user.getName());
❸       userInfoButton.setIconTextGap(JLabel.RIGHT);                   //设置文本显示在头像右侧
❹       userInfoButton.setToolTipText(user.getTipText());              //设置提示文本
        userInfoButton.getParent().doLayout();
    } catch (UnknownHostException e1) {
        e1.printStackTrace();
    }
}
```

代码贴士

❶ setIcon()：该方法用于设置按钮的图标。

❷ setText()：该方法用于设置按钮的文本内容。

❸ setIconTextGap()：该方法用于设置按钮中的文本和图像的显示顺序。

❹ setToolTipText()：该方法用于设置按钮的提示文本。

（6）编写 main()方法，该方法是主程序的入口方法，在该方法中首先获取用户设置的界面外观，其中包括“当前系统”和“Java 默认”两种外观，然后调用 UIManager 类的 setLookAndFeel()方法设置指定的外观，并生成主窗体对象，最后初始化服务器端口和系统栏图标。关键代码如下：

例程 06 代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
public static void main(String args[]) {
    try {
        String laf = preferences.get("lookAndFeel", "java 默认");      //获取用户选择的外观
        if (laf.indexOf("当前系统")>-1)
            UIManager.setLookAndFeel(UIManager
                    .getSystemLookAndFeelClassName());                 //设置外观
        EQ frame = new EQ();                                           //创建主窗体对象
        frame.setVisible(true);                                        //显示窗体
        frame.SystemTrayInitial();                                     //初始化系统栏
        frame.server();                                                //启动服务端口
        frame.checkPlacard();                                          //检测系统公告
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

2.4.2 记录窗体位置

主窗体使用的数据表：tb_location

（1）为窗体添加控件监听器，当改变窗体大小或者移动位置时，调用 saveLocation()方法将窗体的当前位置和大小保存到数据库中。关键代码如下：

例程 07　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
addComponentListener(new ComponentAdapter() {
    public void componentResized(final ComponentEvent e) {          //当窗体改变大小
        saveLocation();                                              //保存改变到数据库
    }
    public void componentMoved(final ComponentEvent e) {            //当窗体改变位置
        saveLocation();                                              //保存改变到数据库
    }
});
```

（2）编写 saveLocation()方法，在该方法中调用 Dao 数据库操作类的 updateLocation()方法将窗体的位置和大小保存到数据库中。关键代码如下：

例程 08　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
private void saveLocation() {                                        //保存主窗体位置的方法
    location = getBounds();                                          //获取窗体位置和大小
    dao.updateLocation(location);                                    //调用 updateLocation() 方法
}
```

2.5　公共模块设计

在本系统的项目空间中，有部分模块是公用的，或者是多个模块甚至整个系统的配置信息，它们被多个模块重复调用完成指定的业务逻辑，本节将这些公共模块提出来作单独介绍。

2.5.1　数据库操作类

Dao 类主要负责有关数据库的操作，该类在构造方法中驱动并连接数据库，然后将构造方法设置为 private 私有属性，通过静态的 getDao()方法获取 Dao 类的实例对象，这是典型的单例模式。在连接数据库时，可以指定 create 参数为 true 直接创建数据库，但在此之前需要调用 dbExists()方法判断数据库是否存在。Dao 类的关键代码如下；

例程 09　代码位置：光盘\TM\02\EQ\src\com\lzw\dao\Dao.java

```
public class Dao {
    private static final String driver = "org.apache.derby.jdbc.EmbeddedDriver";   //数据库驱动
    private static String url = "jdbc:derby:db_EQ";                                //数据库 URL
    private static Connection conn = null;                                         //数据库连接
    private static Dao dao = null;
    private Dao() {
        try {
❶           Class.forName(driver);                                                 //加载驱动
            if (!dbExists()) {                                                     //如果数据库不存在
            conn = DriverManager.getConnection(url + ";create=true");              //创建数据库
                createTable();                                                     //创建数据表
            } else
```

```
❷                   conn = DriverManager.getConnection(url);                    //连接数据库
                addDefUser();                                                   //添加本地用户到数据库
            } catch (Exception e) {
                e.printStackTrace();
                JOptionPane.showMessageDialog(null, "数据库连接异常，或者本软件已经运行。");
                System.exit(0);
            }
        }
❸       private boolean dbExists() {                                             //检测数据库的方法
            boolean bExists = false;
            File dbFileDir = new File("db_EQ");
            if (dbFileDir.exists()) {
                bExists = true;
            }
            return bExists;
        }
❹       public static Dao getDao() {                                            //获取 DAO 实例的方法
            if (dao == null)
                dao = new Dao();
            return dao;
        }
        public void createTable() {                                             //创建数据表的方法
            String createUserSql = "CREATE TABLE tb_users ("
                    + "ip varchar(16) primary key," + "host varchar(30),"
                    + "name varchar(20)," + "tooltip varchar(50),"
                    + "icon varchar(50))";
            String createLocationSql = "CREATE TABLE tb_location ("
                    + "xLocation int," + "yLocation int," + "width int,"
                    + "height int)";
            try {
                Statement stmt = conn.createStatement();
                stmt.execute(createUserSql);                                    //创建用户数据表
                stmt.execute(createLocationSql);                                //创建窗体定位数据表
                addDefLocation();
                stmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
}
```

代码贴士

❶ forName()：该方法用于加载指定的类。

❷ getConnection()：该方法用于获取数据库的连接对象。

❸ dbExists()：该自定义方法用于检测数据库文件夹是否存在，如果存在，说明数据库已经建立。

❹ getDao()：该自定义方法用于获取数据库访问类的实例。

1．addDefLocation()方法

该方法用于添加窗体默认位置和大小到数据库，在企业内部通信系统首次运行时，将使用该方法设置窗体的位置和大小。如果用户更改了窗体位置或者窗体大小，该方法所保存的数据将不再起作用。关键代码如下：

例程 10　代码位置：光盘\TM\02\EQ\src\com\lzw\dao\Dao.java

```
public void addDefLocation() {                                    //添加默认窗体位置
    String sql = "insert into tb_location values(?,?,?,?)";      //定义带参数的 SQL 语句
    try {
        PreparedStatement pst = conn.prepareStatement(sql);
        pst.setInt(1, 100);                                       //传递第一个 SQL 参数
        pst.setInt(2, 0);                                         //传递第二个 SQL 参数
        pst.setInt(3, 240);                                       //传递第 3 个 SQL 参数
        pst.setInt(4, 500);                                       //传递第 4 个 SQL 参数
        pst.executeUpdate();                                      //执行 SQL 更新
        pst.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

2．addDefUser()方法

该方法使用本机 IP 地址创建默认用户，并添加到数据库中。除了使用本机 IP 地址外，默认的用户还包括主机名称、姓名、提示文本和头像图标等属性。关键代码如下：

例程 11　代码位置：光盘\TM\02\EQ\src\com\lzw\dao\Dao.java

```
public void addDefUser() {                                        //创建本机用户
    try {
        InetAddress local = InetAddress.getLocalHost();           //获取本机地址对象
        User user = new User();                                   //创建新用户对象
        user.setIp(local.getHostAddress());                       //初始化用户对象
        user.setHost(local.getHostName());
        user.setName(local.getHostName());
        user.setTipText(local.getHostAddress());
        user.setIcon("1.gif");
        if (getUser(user.getIp()) == null) {
            addUser(user);                                        //添加该用户到数据库中
        }
    } catch (UnknownHostException e) {
        e.printStackTrace();
    }
}
```

3．getLocation()方法

该方法用于从数据库中获取窗体位置和窗体大小信息，并将这些信息封装成 Rectangle 类的实例对象，然后作为方法的返回值。关键代码如下：

例程 12 代码位置：光盘\TM\02\EQ\src\com\lzw\dao\Dao.java

```
public Rectangle getLocation() {                                    //获取窗体位置
    Rectangle rec = new Rectangle(100, 0, 240, 500);                //创建 rec 对象并带有默认数据
    String sql = "select * from tb_location";
    try {
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);                      //从数据库读取数据
        if (rs.next()) {
            rec.x = rs.getInt(1);                                   //初始化 rec 对象
            rec.y = rs.getInt(2);
            rec.width = rs.getInt(3);
            rec.height = rs.getInt(4);
        }
        rs.close();
        stmt.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return rec;
}
```

4．addUser()方法

该方法用于添加指定用户到数据库中，方法接收 User 类的实例对象，即用户对象作参数，在 SQL 语句中将用户对象的属性作为参数插入到数据库中。关键代码如下：

例程 13 代码位置：光盘\TM\02\EQ\src\com\lzw\dao\Dao.java

```
public void addUser(User user) {                                    //添加用户
    try {
        String sql = "insert into tb_users values(?,?,?,?,?)";     //添加用户的 SQL 语句
        PreparedStatement ps = null;
        ps = conn.prepareStatement(sql);
        ps.setString(1, user.getIp());                              //填充 SQL 语句的参数
        ps.setString(2, user.getHost());
        ps.setString(3, user.getName());
        ps.setString(4, user.getTipText());
        ps.setString(5, user.getIcon());
        ps.execute();                                               //执行 SQL 语句
        ps.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

5．delUser()方法

该方法用于从数据库中删除指定用户的记录，方法接收 User 类的实例对象参数，并以该用户对象的 IP 属性为查询条件，从数据库中删除指定 IP 的用户信息。关键代码如下：

例程 14　代码位置：光盘\TM\02\EQ\src\com\lzw\dao\Dao.java

```
public void delUser(User user) {                                    //删除用户
    try {
        String sql = "delete from tb_users where ip=?";             //删除用户的 SQL 语句
        PreparedStatement ps = null;
        ps = conn.prepareStatement(sql);
        ps.setString(1, user.getIp());                              //填充 SQL 语句的参数
        ps.executeUpdate();                                         //执行 SQL 语句
        ps.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

6．getUser()方法

该方法用于从数据库获取指定 IP 的用户对象，该用户对象包含了指定 IP 地址的用户的所有属性，并且将这些属性封装到用户对象中，然后作为方法的返回值。关键代码如下：

例程 15　代码位置：光盘\TM\02\EQ\src\com\lzw\dao\Dao.java

```
public User getUser(String ip) {                                    //获取指定 IP 的用户
    String sql = "select * from tb_users where ip=?";               //定义查询用户表的 SQL 语句
    User user = null;
    try {
        PreparedStatement pst = conn.prepareStatement(sql);
        pst.setString(1, ip);
        ResultSet rs = pst.executeQuery();                          //执行用户查询
        if (rs.next()) {
            user = new User();                                      //创建用户对象
            user.setIp(rs.getString(1));                            //初始化用户对象
            user.setHost(rs.getString(2));
            user.setName(rs.getString(3));
            user.setTipText(rs.getString(4));
            user.setIcon(rs.getString(5));
        }
        rs.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return user;                                                    //返回用户对象
}
```

7．updateLocation()方法

该方法用于更新窗体位置和窗体大小信息，将接收 Rectangle 类的实例对象作参数，将位置、宽度和高度等参数保存到数据库中。关键代码如下：

例程 16　代码位置：光盘\TM\02\EQ\src\com\lzw\dao\Dao.java

```
public void updateLocation(Rectangle location) {                   //更新窗体位置
    String sql = "update tb_location set xLocation=?,yLocation=?,width=?,height=?";
```

```
    try {
❶       PreparedStatement pst
❷                                 = conn.prepareStatement(sql);
❸       pst.setInt(1, location.x);                                    //初始化 SQL 语句的参数
        pst.setInt(2, location.y);
        pst.setInt(3, location.width);
        pst.setInt(4, location.height);
❹       pst.executeUpdate();                                          //执行 SQL 更新语句
        pst.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

代码贴士

❶ PreparedStatement：该类用于定义带参数的 SQL 语法对象。

❷ prepareStatement()：该方法用于创建 PreparedStatement 的实例对象。

❸ setInt()：该方法用于设置 PreparedStatement 实例对象的整数参数，还可用于设置其他参数类型的方法。

❹ executeUpdate()：该方法用于执行 PreparedStatement 实例对象所定义的 SQL 语句。

8．updateUser()方法

该方法用于更新用户信息，其中可更新的内容包括除 IP 地址以外的所有信息，如用户的主机名称、头像、姓名等。关键代码如下：

例程 17 代码位置：光盘\TM\02\EQ\src\com\lzw\dao\Dao.java

```
public void updateUser(User user) {                                   //修改用户
    try {
        String sql = "update tb_users set host=?,name=?,tooltip=?,icon=? where ip='"
                + user.getIp() + "'";
        PreparedStatement ps = null;
        ps = conn.prepareStatement(sql);
        ps.setString(1, user.getHost());                              //初始化 SQL 语句的参数
        ps.setString(2, user.getName());
        ps.setString(3, user.getTipText());
        ps.setString(4, user.getIcon());
        ps.executeUpdate();                                           //执行 SQL 更新语句
        ps.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

2.5.2 系统工具类

Resource 类是企业内部通信系统中的工具类，该类中的工具方法都是静态的，可以直接调用，而不用创建 Resource 类的实例对象。这些工具方法包括搜索用户的方法、登录公共资源的方法、信使群发的方法和单条信息发送的方法。

1．searchUsers()方法

该方法用于搜索局域网中的通信用户，即搜索企业中的所有职工。该方法将获取用户指定的 IP 搜索范围，并在该范围内搜索所有可以访问的计算机，如果用户没有指定 IP 范围，那么系统默认的范围是 192.168.0.1～192.168.1.255。searchUsers()方法的关键代码如下：

例程 18　代码位置：光盘\TM\02\EQ\src\com\lzw\system\Resource.java

```
public static void searchUsers(ChatTree tree, JProgressBar progressBar,
        JList list, JToggleButton button) {
    String ipStart = EQ.preferences.get("ipStart", "192.168.0.1");
    String ipEnd = EQ.preferences.get("ipEnd", "192.168.1.255");
    String[] is = ipStart.split("\\.");
    String[] ie = ipEnd.split("\\.");
    int[] ipsInt = new int[4];
    int[] ipeInt = new int[4];
    for (int i = 0; i < 4; i++) {
        ipsInt[i] = Integer.parseInt(is[i]);
        ipeInt[i] = Integer.parseInt(ie[i]);
    }
    progressBar.setIndeterminate(true);
    progressBar.setStringPainted(true);
    DefaultListModel model = new DefaultListModel();
    model.addElement("搜索结果：");
    list.setModel(model);
    try {
        for (int l = ipsInt[0]; l <= ipeInt[0]; l++) {
            boolean b0 = l < ipeInt[0];                          //记录第一层循环的条件
            int k = l != ipsInt[0] ? 0 : ipsInt[1];              //从第二次循环以后 k 赋值 0
            for (; b0 ? k < 256 : k <= ipeInt[1]; k++) {
                boolean b1 = b0 || k < ipeInt[1];                //记录第二层循环的条件
                int j = k != ipsInt[1] ? 0 : ipsInt[2];          //从第二次循环以后 j 赋值 0
                for (; b1 ? j < 256 : j <= ipeInt[2]; j++) {
                    boolean b2 = b1 || b1 ? j < 256 : j < ipeInt[2];
                    int i = j != ipsInt[2] ? 0 : ipsInt[3];
                    for (; b2 ? i < 256 : i <= ipeInt[3]; i++) {
                        if(!button.isSelected()){                //如果搜索新用户按钮没有被选择
                            progressBar.setIndeterminate(false); //取消进度条的滚动
                            return;
                        }
                        Thread.sleep(100);                       //线程休息 100 毫秒
                        String ip = l + "." + k + "." + j + "." + i;
                        progressBar.setString("正在搜索：" + ip); //设置进度条文本
                        if (tree.addUser(ip, "search"))
                            model.addElement("<html><b><font color=green>添加"
                                    + ip + "</font></b></html>");//添加新用户
                    }
                }
            }
        }
    } catch (Exception e) {
```

```
            e.printStackTrace();
        }finally{
            progressBar.setIndeterminate(false);                    //停止进度条
            progressBar.setString("搜索完毕");                       //进度条显示搜索完毕
            button.setText("搜索新用户");                            //恢复搜索新用户按钮文本
            button.setSelected(false);                              //恢复搜索新用户按钮状态
        }
    }
```

2．loginPublic()方法

该方法用于登录程序升级的服务器，它获取用户指定的升级路径、用户名和密码，使用 net use 命令访问服务器，并返回访问成功或者失败的 boolean 值。关键代码如下：

例程 19 代码位置：光盘\TM\02\EQ\src\com\lzw\system\Resource.java

```
public static boolean loginPublic(String user, String pass) {
    try {
        String userName = user;
        String updatePath = EQ.preferences.get("updatePath", null);          //获取升级路径
        if (updatePath == null)                                              //如果未指定路径
            return false;                                                    //返回失败结果
        File file = new File(updatePath);
        if (!file.exists())
            return false;
        if (file.isFile())
            updatePath = file.getParent();
        if (userName != null && !userName.equals(""))
            userName = " /user:" + userName;
        Process process = Runtime.getRuntime().exec(
                "cmd /c %windir%" + File.separator + "System32"
                        + File.separator + "net use " + updatePath + " "
                        + pass + userName);                                  //访问升级服务器
        Scanner sce = new Scanner(process.getErrorStream());
        StringBuilder stre = new StringBuilder();
        while (sce.hasNextLine()) {
            stre.append(sce.nextLine());                                     //获取访问结果
        }
        process.destroy();
        String resulte = stre.toString();
        if (resulte.equals(""))                                              //如果没有访问结果
            return true;                                                     //访问成功
        else                                                                 //否则
            JOptionPane.showMessageDialog(EQ.frame, resulte, "错误信息",
                    JOptionPane.ERROR_MESSAGE);                              //提示错误信息
    } catch (IOException e) {
        e.printStackTrace();
    }
    return false;
}
```

3．sendMessenger()方法

该方法用于发送信使到指定用户的操作系统。当通信对方没有运行企业内部通信系统时，就无法接收到通信系统发送的内容，这时可以调用该方法向对方发送信使，对方的操作系统接收到信使后会以对话框的通知方式显示信使内容。关键代码如下：

例程 20　代码位置：光盘\TM\02\EQ\src\com\lzw\system\Resource.java

```
public static void sendMessenger(User user, String message, TelFrame frame) {    //发送信使信息
    class TheThread implements Runnable {                                         //定义内部线程类
…//省略部分代码
        public void run() {
            try {
                sendButton.setEnabled(false);                                     //取消发送按钮的状态
                Process process = Runtime.getRuntime().exec(
                        "net send " + user.getIp() + " " + message);              //执行信使发送命令
                InputStream is = process.getInputStream();
                int i, j;
                StringBuilder sb = new StringBuilder();
                while ((i = is.read()) != -1) {
                    sb.append((char) i);                                          //获取信使发送结果
                }
                String runIs = new String(sb.toString().getBytes(
                        "iso-8859-1")).trim().replace(user.getIp(),
                        user.getName());                                          //将结果信息转码
                InputStream eis = process.getErrorStream();
                StringBuilder esb = new StringBuilder();
                while ((j = eis.read()) != -1) {
                    esb.append((char) j);                                         //获取信使发送的错误
                }
                String runEis = new String(esb.toString().getBytes(
                        "iso-8859-1")).trim().replace(user.getIp(),
                        user.getName());                                          //将错误信息转码
                frame.appendReceiveText(runIs, new Color(187, 30, 193));//显示信使发送结果
                if (runEis.length() > 0)                                          //如果存在错误信息
                    frame.appendReceiveText(runIs, Color.RED);                    //提示发送失败
                sendButton.setEnabled(true);                                      //恢复发送按钮的状态
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    new Thread(new TheThread(user, message, frame)).start();
}
```

4．sendGroupMessenger()方法

该方法可以向指定用户群体发送信使，如果企业中有针对会议参与人员的信息或者是个别群体职工的信息，可以使用该方法向指定群体发送信使。群发信使的界面效果如图 2.11 所示。

图 2.11　群发信使界面效果

该方法的关键代码如下：

例程 21　代码位置：光盘\TM\02\EQ\src\com\lzw\system\Resource.java

```
public static void sendGroupMessenger(final TreePath[] selectionPaths, final String message) {//群发信使信息
    new Thread(new Runnable() {                                                    //创建内部线程
        int bufferSize = 512;
        public void run() {
            MessageFrame messageFrame = new MessageFrame();                        //创建信使窗口
            try {
                for (TreePath path : selectionPaths) {                             //遍历选择的用户
                    DefaultMutableTreeNode node = (DefaultMutableTreeNode) path.getLastPath
Component();
                    User user = (User) node.getUserObject();
                    messageFrame.setStateBarInfo("<html>正在给<font color=blue>"
                            + user.getName()+ "</font>发送消息……</html>");//提示发送信使
                    Thread.sleep(20);                                              //线程休眠
                    InetAddress addr
❶                           = InetAddress.getByName(user.getIp());
                    if (!addr.getHostAddress().equals(addr.getHostName())) {
❷                       Process process
❸                           = Runtime.getRuntime()
❹                                   .exec(
                            "net send " + user.getIp() + " " + message);       //发送信使
                        InputStream is = process.getInputStream();
                        int i;
                        String sb = null;
                        byte[] data = new byte[bufferSize];
                        if ((i = is.read(data)) != -1) {
                            sb = new String(data, 0, i);                       //获取发送结果
                        }
                        String runIs = sb;
                        runIs = runIs.replace(user.getIp(), user.getName()).trim();
                        process.destroy();
                        if (runIs.indexOf("出错") < 0)                          //如果结果包含错误
                            messageFrame.addMessage(runIs, true);              //显示错误信息
                        else                                                   //否则
                            messageFrame.addMessage(runIs, false);             //显示成功信息
                    } else {
                        messageFrame.addMessage("错误：" + user.getName()
                                + "可能没有开机或启动了防火墙", false); //发送失败的提示
                    }
```

```
                    }
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                messageFrame.setStateBarInfo("消息发送完毕,可以关闭窗口。");//提示信使群发完毕
            }
        }).start();
    }
```

代码贴士

❶ getByName()：该方法用于获取指定主机名称的 InetAddress 类的实例对象，该类用于封装网络通信地址。

❷ Process：该类是本机进程类，它提供了执行进程、获取进程输入/输出流、检查进程状态以及销毁进程的方法。

❸ getRuntime()：该方法用于获取 Process 类的实例对象。

❹ exec()：该方法用于执行指定的外部进程并返回 Process 类的实例。

5．startFolder()方法

该方法用于打开指定的文件夹或者网络共享资源。它通过“cmd /c start”指令打开 str 参数指定的文件夹位置。关键代码如下：

例程 22　代码位置：光盘\TM\02\EQ\src\com\lzw\system\Resource.java

```
public static void startFolder(String str) {
    try {
        Runtime.getRuntime().exec("cmd /c start " + str);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

2.6　系统托盘模块设计

2.6.1　系统托盘模块概述

系统托盘模块用于定义系统栏图标。企业内部通信系统的主窗体是继承对话框窗体编写的，该窗体在系统任务栏不会显示相应的任务标题，如果主窗体最小化之后将会隐藏，这时必须使用快捷键或者系统托盘中的图标执行显示窗体的命令。本系统在系统托盘中的效果如图 2.12 所示。

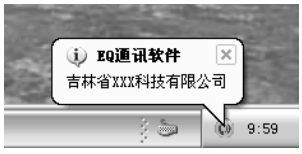


图 2.12　系统托盘效果

2.6.2　系统托盘模块技术分析

系统托盘模块使用了 JDK 6.0 提供的新功能，其中包含 SystemTray 类和 TrayIcon 类，它们分别用于创建系统托盘和系统栏图标对象。另外，系统栏图标使用了弹出菜单技术，为企业内部通信系统提供了部分快捷操作。创建弹出菜单和菜单项由 PopupMenu 类和 MenuItem 类实现，一个 PopupMenu 对

象可以使用 add()方法添加多个 MenuItem 对象，每个 MenuItem 对象必须使用 addActionListener()方法添加实现指定菜单项业务逻辑的监听器，其监听器的实现和普通按钮相同。

2.6.3 系统托盘模块的实现过程

系统托盘模块的开发步骤如下：

（1）在程序主类中编写 SystemTrayInitial()方法，该方法用于初始化系统托盘。在方法中初始化系统托盘中的提示文本、系统栏图标，然后调用 createMenu()方法为系统栏图标创建弹出菜单，同时为系统栏图标添加 SysTrayActionListener 类事件监听器。关键代码如下：

例程 23 代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
private void SystemTrayInitial() {
    if (!SystemTray.isSupported())                                  //判断当前系统是否支持系统栏
        return;
    try {
        String title = "EQ 通讯软件";                                //系统栏图标提示文本的标题
        String company = "吉林省×××科技有限公司";                    //系统栏图标提示文本
❶       SystemTray sysTray = SystemTray.getSystemTray();            //获取系统托盘对象
        Image image = Toolkit.getDefaultToolkit().getImage(
                EQ.class.getResource("/icons/sysTray.png"));        //系统栏图标
❷       trayicon = new TrayIcon(image, title + "\n" + company, createMenu());    //创建系统栏图标对象
❸       trayicon.setImageAutoSize(true);                            //设置自动大小
❹       trayicon.addActionListener(new SysTrayActionListener());    //添加事件监听器
❺       sysTray.add(trayicon);                                      //添加系统栏图标到系统托盘
❻       trayicon.displayMessage(title, company, MessageType.INFO);  //显示系统栏图标提示文本
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

代码贴士

❶ getSystemTray()：该方法用于获取系统托盘 SystemTray 类的实例对象，该实例对象无法通过构造方法创建。

❷ TrayIcon：该类是系统托盘的图标对象。

❸ setImageAutoSize()：该方法用于设置系统托盘图标大小的自动调整。

❹ addActionListener()：该方法用于为系统托盘图标添加事件监听器。

❺ add()：该方法用于添加系统栏图标到系统托盘中。

❻ displayMessage()：该方法用于显示系统托盘上的气泡提示文本。

（2）编写 createMenu()方法，该方法用于创建系统栏图标的弹出菜单，该菜单包括打开、访问服务器和退出 3 个菜单项和菜单项分隔符。当用户选择“打开”菜单项时，将显示企业内部通信系统的主窗体；选择“访问服务器”菜单项时将打开公共程序资源。另外主窗体没有提供退出功能，单击右上角的关闭按钮时执行的是最小化操作，所以只能使用“退出”菜单项完成程序的退出功能。关键代码如下：

例程 24　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
private PopupMenu createMenu() {                                    //创建系统栏菜单的方法
    PopupMenu menu = new PopupMenu();
    MenuItem exitItem = new MenuItem("退出");
    exitItem.addActionListener(new ActionListener() {               //系统栏退出事件
                public void actionPerformed(ActionEvent e) {
                    System.exit(0);
                }
            });
    MenuItem openItem = new MenuItem("打开");
    openItem.addActionListener(new ActionListener() {               //系统栏打开菜单项事件
                public void actionPerformed(ActionEvent e) {
                    if (!isVisible()) {                             //如果主窗体隐藏
                        setVisible(true);                           //显示主窗体
                        toFront();                                  //使主窗体显示在最上层
                    } else
                        toFront();
                }
            });
    MenuItem publicItem = new MenuItem("访问服务器");                 //系统栏的访问服务器菜单项事件
    publicItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String serverPaeh = preferences.get("pubPath", null);
            if (serverPaeh == null) {
                pushMessage("未设置公共程序路径");                     //添加提示信息
                return;
            }
            Resource.startFolder(serverPaeh);                       //调用 startFolder()方法
        }
    });
    menu.add(publicItem);                                           //添加访问服务器菜单项
    menu.add(openItem);                                             //添加打开菜单项
    menu.addSeparator();                                            //添加菜单项分隔符
    menu.add(exitItem);                                             //添加退出菜单项
    return menu;
}
```

（3）创建 SysTrayActionListener 内部类，它实现了 ActionListener 接口，是系统栏图标的双击事件监听器。当用户双击系统栏图标后，该监听器将实现主窗体的显示，这和系统栏图标的“打开”菜单项所实现的功能相同，但是双击系统栏图标会更加方便。关键代码如下：

例程 25　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
class SysTrayActionListener implements ActionListener {             //系统栏双击事件
    public void actionPerformed(ActionEvent e) {
        setVisible(true);
        toFront();
    }
}
```

2.7 系统工具模块设计

2.7.1 系统工具模块概述

企业内部通信系统的系统工具模块起到维护系统的作用，其功能包括用户搜索、更换程序外观、系统升级。在企业内部通信系统第一次运行时，使用用户搜索功能可以搜索内部网络中所有正在运行的计算机，并使用计算机的信息创建用户对象，然后将该用户对象保存到数据库中。在企业内部通信系统有新版本程序时，可以使用系统升级功能直接升级到最新版本，而不用重新安装。系统工具模块的运行效果如图 2.13 所示。

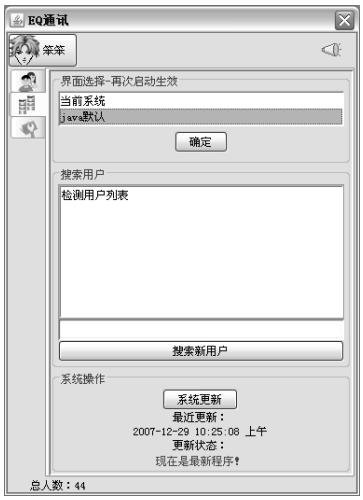


图 2.13 系统工具模块运行效果

2.7.2 系统工具模块技术分析

系统工具模块中使用了 Java 的 LookAndFeel 外观技术，每个 LookAndFeel 外观会包含不同控件的 UI 界面，不同的外观中控件的外观也会不同，例如 Windows 系统的“Windows 经典样式”外观和“Windows XP 样式”外观的按钮、列表、表格、菜单、工具栏甚至窗体的外观都不相同。

使用 UIManager 类的 setLookAndFeel()方法可以设置不同的 LookAndFeel 外观。企业内部通信系统提供了“当前系统”和“Java 默认值”两个外观选项，其中“Java 默认值”是 Swing 默认的外观，不需要特别设置，而“当前系统”外观需要使用 getSystemLookAndFeelClassName()方法获取当前系统的外观名称，然后调用 setLookAndFeel()方法将该外观名称设置为默认外观。

注意 必须在创建窗体和控件之前，使用 UIManager 类的 setLookAndFeel()方法设置外观，否则会造成外观样式显示不完整的后果。

2.7.3 系统工具模块的实现过程

系统工具使用的数据表：tb_users

系统工具模块的开发步骤如下：

（1）在程序主类中编写 createSysToolPanel()方法，用于创建系统工具选项卡，在该选项卡中包括界面选择、用户搜索和系统操作 3 部分，其中系统操作用于程序更新，它们都被添加到系统工具面板中，createSysToolPanel()方法必须设置好该面板的布局和初始化工作。关键代码如下：

例程 26 代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
private JScrollPane createSysToolPanel() {
    JPanel sysToolPanel = new JPanel();                    //系统工具面板
```

```
        sysToolPanel.setLayout(new BorderLayout());                          //设置面板布局
        JScrollPane sysToolScrollPanel = new JScrollPane();                  //创建滚动面板
        sysToolScrollPanel
                .setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
        sysToolScrollPanel.setBorder(new EmptyBorder(0, 0, 0, 0));           //设置滚动面板的边框
        sysToolScrollPanel.setViewportView(sysToolPanel);                    //使面板可以滚动
        sysToolPanel.setBorder(new BevelBorder(BevelBorder.LOWERED));        //设置系统工具面板边框
…//省略部分代码
        return sysToolScrollPanel;
}
```

（2）在 createSysToolPanel()方法中创建界面选择部分，该部分以列表控件显示了两种外观选择，当用户选择其中一种外观并单击“确定”按钮后，选择的外观会保存到首选项中，然后提示用户重新运行本软件。关键代码如下：

例程 27　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
JPanel interfacePanel = new JPanel();                                        //创建界面面板
sysToolPanel.add(interfacePanel, BorderLayout.NORTH);                        //设置面板的布局位置
interfacePanel.setLayout(new BorderLayout());                                //设置面板布局管理器
interfacePanel.setBorder(new TitledBorder("界面选择-再次启动生效"));          //设置面板的 Title 边框
faceList = new JList(new String[]{"当前系统", "java 默认"});                 //创建界面选择列表
interfacePanel.add(faceList);
faceList.setBorder(new BevelBorder(BevelBorder.LOWERED));                    //设置列表的边框
final JPanel interfaceSubPanel = new JPanel();
interfaceSubPanel.setLayout(new FlowLayout());
interfacePanel.add(interfaceSubPanel, BorderLayout.SOUTH);
selectInterfaceOKButton = new JButton("确定");                               //创建“确定”按钮
selectInterfaceOKButton.addActionListener(new ActionListener() {             //添加按钮事件监听器
    public void actionPerformed(ActionEvent e) {
        preferences.put("lookAndFeel", faceList.getSelectedValue()
                .toString());                                                //保存界面选择到首选项
        JOptionPane.showMessageDialog(EQ.this, "重新运行本软件后生效");       //提示重新运行软件
    }
});
interfaceSubPanel.add(selectInterfaceOKButton);                              //添加按钮到面板中
```

（3）在 createSysToolPanel()方法中创建用户搜索部分，包括搜索列表、搜索进度条和“搜索新用户”按钮 3 个控件。当单击“搜索新用户”按钮时，系统会根据用户在系统设置界面所设置的 IP 搜索范围搜索所有计算机信息，并创建对应的用户对象，然后保存到数据库中。关键代码如下：

例程 28　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
JPanel searchUserPanel = new JPanel();                                       //用户搜索面板
sysToolPanel.add(searchUserPanel);
searchUserPanel.setLayout(new BorderLayout());                               //设置面板布局管理器
final JPanel searchControlPanel = new JPanel();
searchControlPanel.setLayout(new GridLayout(0, 1));
searchUserPanel.add(searchControlPanel, BorderLayout.SOUTH);
final JList searchUserList = new JList(new String[]{"检测用户列表"});         //定义搜索用户列表
```

```
final JScrollPane scrollPane_2 = new JScrollPane(searchUserList);
scrollPane_2.setDoubleBuffered(true);
earchUserPanel.add(scrollPane_2);
searchUserList.setBorder(new BevelBorder(BevelBorder.LOWERED));                    //设置用户列表边框
searchUserButton = new JToggleButton("搜索新用户");                                 //创建“搜索新用户”按钮
searchUserButton.addActionListener(new SearchUserActionListener(searchUserList));//添加按钮的事件监听器
searchControlPanel.add(progressBar);
searchControlPanel.add(searchUserButton);
searchUserPanel.setBorder(new TitledBorder("搜索用户"));                            //设置面板的 Title 边框
```

（4）在 createSysToolPanel()方法中创建系统操作部分，该部分包括“系统更新”按钮和显示程序更新信息的标签控件。当用户单击“系统更新”按钮时，该按钮的事件监听器会下载当前最新版本的程序，然后更新相应的标签控件提示用户更新信息。关键代码如下：

例程 29 代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
final JPanel sysUpdatePanel = new JPanel();                                         //系统更新面板
sysUpdatePanel.setOpaque(false);
sysUpdatePanel.setLayout(new GridBagLayout());                                      //设置面板布局管理器
sysUpdatePanel.setBorder(new TitledBorder("系统操作"));                             //设置边框
sysToolPanel.add(sysUpdatePanel, BorderLayout.SOUTH);
final JButton sysUpdateButton = new JButton("系统更新");                            //定义“系统更新”按钮
final GridBagConstraints gridBagConstraints_1 = new GridBagConstraints();           //定义按钮布局
gridBagConstraints_1.gridx = 0;
gridBagConstraints_1.gridy = 0;
sysUpdatePanel.add(sysUpdateButton, gridBagConstraints_1);
sysUpdateButton.addActionListener(new SysUpdateListener());                         //添加系统更新事件
final JLabel updateLabel = new JLabel("最近更新：");
final GridBagConstraints updateLabelLayout = new GridBagConstraints();
updateLabelLayout.gridy = 1;
updateLabelLayout.gridx = 0;
sysUpdatePanel.add(updateLabel, updateLabelLayout);
final JLabel updateDateLabel = new JLabel();                                        //程序更新日期标签
Date date = new Date(localFile.lastModified());
String dateStr = String.format("%tF %<tr", date);
updateDateLabel.setText(dateStr);
final GridBagConstraints updateDateLayout = new GridBagConstraints();
updateDateLayout.gridy = 2;
updateDateLayout.gridx = 0;
sysUpdatePanel.add(updateDateLabel, updateDateLayout);
final JLabel updateStaticLabel = new JLabel("更新状态：");
final GridBagConstraints updateStaticLayout = new GridBagConstraints();
updateStaticLayout.gridy = 3;
updateStaticLayout.gridx = 0;
sysUpdatePanel.add(updateStaticLabel, updateStaticLayout);
final JLabel updateInfoLabel = new JLabel();                                        //版本信息标签
checkSysInfo(updateInfoLabel);                                                      //调用检测版本更新的方法
final GridBagConstraints gridBagConstraints_5 = new GridBagConstraints();
gridBagConstraints_5.gridy = 4;
gridBagConstraints_5.gridx = 0;
```

```
sysUpdatePanel.add(updateInfoLabel, gridBagConstraints_5);
JPanel statePanel = new JPanel();
add(statePanel, BorderLayout.SOUTH);
statePanel.setLayout(new BorderLayout());
statePanel.add(stateLabel);
stateLabel.setText("总人数：" + chatTree.getRowCount());          //设置状态栏标签内容
```

（5）创建“搜索新用户”按钮的事件监听器 SearchUserActionListener 类，在该监听器中调用 Resource 工具类的 searchUsers()方法搜索指定 IP 范围内的所有用户计算机信息。为避免等待搜索用户的业务逻辑过长而导致程序界面死锁，监听器创建了另一个线程来执行用户搜索的业务。关键代码如下：

例程 30　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
class SearchUserActionListener implements ActionListener {
    private final JList list;
    SearchUserActionListener(JList list) {
        this.list = list;
    }
    public void actionPerformed(ActionEvent e) {
        if (searchUserButton.isSelected()) {                      //如果单击“搜索新用户”按钮
            searchUserButton.setText("停止搜索");                  //将其变成“停止搜索”按钮
❶           new Thread(
❷               new Runnable() {                                  //创建内部线程
❸               public void run() {
                    Resource.searchUsers(chatTree, progressBar,
                            list, searchUserButton);              //调用 searchUsers()方法
                }
❹           }).start();
        } else
            searchUserButton.setText("搜索新用户");                //恢复按钮文本
    }
}
```

代码贴士

❶ Thread：线程类，用于开辟新的线程，执行其他任务。

❷ Runnable：线程接口，用于开辟新的线程，执行其他任务。

❸ run()：线程中的方法，用于定义线程的业务逻辑。

❹ start()：开始执行线程。

（6）创建“系统更新”按钮的事件监听器 SysUpdateListener 类，该监听器在用户单击“系统更新”按钮时，调用 Resource 工具类的 loginPublic()方法登录，升级服务器，然后调用 updateProject()方法更新系统程序文件。关键代码如下：

例程 31　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
private final class SysUpdateListener implements ActionListener {      //系统更新事件监听器
    public void actionPerformed(final ActionEvent e) {
        String username = preferences.get("username", null);            //获取用户名
        String password = preferences.get("password", null);            //获取密码
```

```
        if (username == null || password == null) {                    //如果用户名或密码为空
            pushMessage("未设置登录升级服务器的用户名或密码");          //提示设置登录信息
            return;
        }
        Resource.loginPublic(username, password);                      //调用 loginPublic()方法
        updateProject();                                               //调用 updateProject()方法
    }
}
```

（7）编写更新程序的 updateProject()方法，在该方法中分别创建本地程序和服务器最新程序文件对象，然后解析两个文件的最后修改日期，如果服务器程序的最后修改日期比本地程序更新，那么获取服务器文件的内容并更新到本地文件中。关键代码如下：

例程 32 代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
private void updateProject() {                                         //程序更新的方法
    netFilePath = preferences.get("updatePath", "EQ.jar");             //获取升级路径
    if (netFilePath.equals("EQ.jar")) {                                //如果首选项没有升级路径
        pushMessage("未设置升级路径");                                 //提示错误信息
        return;
    }
    netFile = new File(netFilePath);                                   //创建服务器程序文件对象
    localFile = new File(user_dir + File.separator + "EQ.jar");        //创建本地程序文件对象
    if (localFile != null && netFile != null && netFile.exists()
            && localFile.exists()) {
        Date netDate = new Date(netFile.lastModified());               //提取服务器文件修改时间
        Date localDate = new Date(localFile.lastModified());           //提取本地程序最近修改时间
        if (netDate.after(localDate)) {                                //如果服务器文件较新
            new Thread(new Runnable() {
                public void run() {
                    try {
                        Dialog frameUpdate = new UpdateFrame();        //创建更新窗口
                        frameUpdate.setVisible(true);                  //显示更新窗口
                        Thread.sleep(2000);                            //线程休眠 2 秒
                        FileInputStream fis = new FileInputStream(netFile);  //创建文件输入流
                        FileOutputStream fout = new FileOutputStrea(   //创建文件输出流
                                localFile);
                        int len = fis.available();                     //计算文件有效长度
                        if (len > 0) {
                            byte[] data = new byte[len];
                            if (fis.read(data) > 0) {                  //如果读取到网络文件数据
                                fout.write(data);                      //写入本地文件中
                            }
                        }
                        fis.close();                                   //关闭文件输入流
                        fout.close();                                  //关闭文件输出流
                        frameUpdate.setVisible(false);
                        frameUpdate = null;
                        showMessageDialog("更新完毕，请重新启动程序。");
```

```
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                    }
                }).start();
            } else {
                showMessageDialog("已经是最新的程序了。");
            }
        }
}
```

2.8　用户管理模块设计

2.8.1　用户管理模块概述

用户管理模块类似聊天软件的好友列表，其中包含所有用户信息，另外在用户名称上单击鼠标右键，会弹出管理菜单，菜单中包括更名、添加用户、删除用户、信使群发、访问主机资源和访问公共程序，其中访问主机资源是访问该用户的共享文件夹。用户管理模块的运行效果如图 2.14 所示。

图 2.14　用户管理模块界面效果

2.8.2　用户管理模块技术分析

用户管理模块主要用于显示用户列表，这个用户列表是使用 JTree 树控件实现的。JTree 控件的树节点默认的界面效果难以满足用户列表的外观，用户列表需要绘制当前选择用户的边框、头像、状态图标等信息。为提高用户列表的美观性，用户管理模块必须实现 TreeCellRenderer 接口，创建实现显示自定义图标的树单元格渲染器，这样就可以自定义树节点的样式了。

TreeCellRenderer 接口只定义了一个 getTreeCellRendererComponent()方法，该方法将关于绘制树节点的全部信息作为参数，在实现自己的树单元格渲染器时，可以忽略不需要的参数，也可以直接访问树节点 value 参数。getTreeCellRendererComponent()方法的语法格式如下：

```
Component getTreeCellRendererComponent(JTree tree,
                                       Object value,
                                       boolean selected,
                                       boolean expanded,
                                       boolean leaf,
                                       int row,
                                       boolean hasFocus)
```

语法中涉及的参数说明如表 2.5 所示。

表 2.5　参数说明

参 数 名 称	描　　述	参 数 名 称	描　　述
tree	JTree 树对象	leaf	是否是树的叶节点
value	当前树单元格的值	row	当前节点所在行号
selected	树单元格（也就是树节点）是否被选中	hasFocus	当前节点是否拥有焦点
expanded	节点是否展开		

2.8.3　用户管理模块的实现过程

用户管理模块的开发步骤如下：

（1）创建 UserTreeRanderer 类，该类继承 JPanel 类成为一个面板控件，同时该类也实现了 TreeCellRenderer 接口成为树节点的渲染器。该类的构造方法中接收 3 个图标参数，分别用于树节点的打开、关闭和叶节点的图标。关键代码如下：

例程 33　代码位置：光盘\TM\02\EQ\src\com\lzw\userList\UserTreeRanderer.java

```
public class UserTreeRanderer extends JPanel implements TreeCellRenderer {
    private Icon openIcon, closedIcon, leafIcon;                                    //节点图标
    private final JCheckBox label = new JCheckBox();                                //用户选择图标
    private final JLabel headImg= new JLabel();                                     //用户头像
    private static User user;                                                       //用户对象
    public UserTreeRanderer() {
        super();
        user = null;
    }
    public UserTreeRanderer(Icon open, Icon closed, Icon leaf) {
        openIcon = open;                                                            //节点展开图标
        closedIcon = closed;                                                        //节点折叠图标
        leafIcon = leaf;                                                            //叶节点图标
        setBackground(new Color(0xF5B9BF));                                         //设置背景色
        label.setFont(new Font("宋体", Font.BOLD, 14));                             //设置字体
        URL trueUrl = EQ.class.getResource("/image/chexkBoxImg/CheckBoxTruepng");   //选择用户的图标
        label.setSelectedIcon(new ImageIcon(trueUrl));
        URL falseUrl = EQ.class.getResource("/image/chexkBoxImg/CheckBoxFalse.png");//取消用户的图标
        label.setIcon(new ImageIcon(falseUrl));
        label.setForeground(new Color(0, 64, 128));
        final BorderLayout borderLayout = new BorderLayout();                       //创建布局管理器
        setLayout(borderLayout);                                                    //设置布局管理器
        user = null;
    }
…//省略部分代码
}
```

（2）在 UserTreeRanderer 类中重写父类的 getTreeCellRendererComponent()方法，它负责渲染树节点的界面样式。该方法将获取主窗体的宽度，并使用该宽度值设置节点的宽度，使节点与窗体同宽。

当用户选择某个节点时，该方法将使用指定颜色绘制节点的边框，以突出该节点被选择的效果。关键代码如下：

例程 34　代码位置：光盘\TM\02\EQ\src\com\lzw\userList\UserTreeRanderer.java

```
public Component getTreeCellRendererComponent(JTree tree, Object value,
        boolean selected, boolean expanded, boolean leaf, int row, boolean hasFocus) {
    if (value instanceof DefaultMutableTreeNode) {                          //判断 value 是否是节点
        DefaultMutableTreeNode node = (DefaultMutableTreeNode) value;
        Object uo = node.getUserObject();                                   //获取节点数据
        if (uo instanceof User)                                             //如果该数据是 User 实例
            user = (User) uo;                                               //初始化 User 对象
    } else if (value instanceof User)                                       //如果 value 是 User 实例
        user = (User) value;                                                //初始化 User 对象
    if (user != null && user.getIcon() != null) {
        int width = EQ.frame.getWidth();                                    //获取主窗体宽度
        if (width > 0)
           setPreferredSize(new Dimension(width, user.getIconImg().getIconHeight()));//使节点与窗体同宽
        headImg.setIcon(user.getIconImg());                                 //设置用户头像
        tipText = user.getName();
    } else {
        if (expanded)
            headImg.setIcon(openIcon);
        else if (leaf)
            headImg.setIcon(leafIcon);
        else
            headImg.setIcon(closedIcon);
    }
    add(headImg, BorderLayout.WEST);                                        //添加用户头像
    label.setText(value.toString());                                        //设置用户名称
    label.setOpaque(false);
    add(label, BorderLayout.CENTER);                                        //添加用户名称
    if (selected) {                                                         //如果该节点被选择
        label.setSelected(true);
        setBorder(new LineBorder(new Color(0xD46D73), 2, false));           //以指定颜色绘制边框
        setOpaque(true);
    } else {                                                                //否则
        setOpaque(false);
        label.setSelected(false);
        setBorder(new LineBorder(new Color(0xD46D73), 0, false));           //恢复原来的边框颜色
    }
    return this;
}
```

（3）创建 ChatTree 类，该类继承 JTree 类实现自定义的树控件，并且使用了之前定义的 UserTreeRanderer 树节点渲染器，它在构造方法中初始化类的属性，然后调用 sortUsers()方法添加并显示用户列表。关键代码如下：

例程 35 代码位置：光盘\TM\02\EQ\src\com\lzw\userList\ChatTree.java

```
public class ChatTree extends JTree {
    private DefaultMutableTreeNode root;
    private DefaultTreeModel treeModel;
    private List<User> userMap;                                    //用户集合
    private Dao dao;                                               //数据库操作类
    private EQ eq;
    public ChatTree(EQ eq) {
        super();
        root = new DefaultMutableTreeNode("root");                 //初始化根节点
        treeModel = new DefaultTreeModel(root);
        userMap = new ArrayList<User>();                           //初始化用户集合
        dao = Dao.getDao();                                        //初始化数据库操作类
        addMouseListener(new ThisMouseListener());
        setRowHeight(50);                                          //设置节点的高度
        setToggleClickCount(2);
        setRootVisible(false);                                     //隐藏根节点
        DefaultTreeCellRenderer defaultRanderer = new DefaultTreeCellRenderer();
        UserTreeRanderer treeRanderer = new UserTreeRanderer(defaultRanderer
                .getOpenIcon(), defaultRanderer.getClosedIcon(),
                defaultRanderer.getLeafIcon());                    //创建自定义节点渲染器
        setCellRenderer(treeRanderer);                             //设置该节点渲染器
        setModel(treeModel);
        sortUsers();                                               //添加并显示所有节点
        this.eq = eq;
    }
…//省略部分代码
}
```

（4）在 ChatTree 类中编写 sortUsers()方法，该方法的主体是一个内部线程，该线程首先获取本地 IP 地址，使用该地址从数据库中获取本地用户对象，并将本地用户显示在用户列表的首位。然后，从数据库中获取所有用户对象，将除自己以外的用户分别添加到用户列表中。最后，使第一个用户处于被选中的状态，并更新状态栏标签中显示的用户数量。关键代码如下：

例程 36 代码位置：光盘\TM\02\EQ\src\com\lzw\userList\ChatTree.java

```
private synchronized void sortUsers() {
    new Thread(new Runnable() {
        public void run() {
            try {
                Thread.sleep(100);                                 //线程休眠 100 毫秒
                root.removeAllChildren();
                String ip = InetAddress.getLocalHost().getHostAddress();  //获取本地 IP
                User localUser = dao.getUser(ip);                  //从数据库中获取自己
                if (localUser != null) {
                    DefaultMutableTreeNode node = new DefaultMutableTreeNode(
                            localUser);
                    root.add(node);                                //把自己显示在首位
                }
```

```
                    userMap = dao.getUsers();                                    //获取数据库中所有用户
                    Iterator<User> iterator = userMap.iterator();
                    while (iterator.hasNext()) {                                 //遍历用户集合
                        User user = iterator.next();
                        if(user.getIp().equals(localUser.getIp()))
                            continue;
                        root.add(new DefaultMutableTreeNode(user));              //添加用户到根节点
                    }
                    treeModel.reload();
                    ChatTree.this.setSelectionRow(0);                            //使第一个节点被选择
                    if (eq != null)
                        eq.setStatic("      总人数：" + getRowCount());           //更新状态栏标签
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }
```

（5）在 ChatTree 类中编写 delUser()方法，用于删除当前用户列表中选择的用户对象。该方法首先获取选择的树节点，从该节点中获取绑定的用户对象，然后以对话框提示用户是否确认删除，如果经过用户确认，将调用 delUser()方法从数据库中删除用户信息，最后调用根节点的 remove()方法删除该用户节点。关键代码如下：

例程 37　代码位置：光盘\TM\02\EQ\src\com\lzw\userList\ChatTree.java

```
public void delUser() {
    TreePath path = getSelectionPath();                                          //获取被选择的树节点
    if (path == null)
        return;
    User user = (User) ((DefaultMutableTreeNode) path
            .getLastPathComponent()).getUserObject();                            //获取节点中的用户对象
    int operation = JOptionPane.showConfirmDialog(this, "确定要删除用户：" + user
            + "?", "删除用户", JOptionPane.YES_NO_OPTION,
            JOptionPane.QUESTION_MESSAGE);                                       //以对话框提示确认删除
    if (operation == JOptionPane.YES_OPTION) {
        dao.delUser(user);                                                       //调用 delUser()方法
        root.remove((DefaultMutableTreeNode)path.getLastPathComponent());        //删除该节点
        treeModel.reload();
    }
}
```

（6）在 ChatTree 类中编写 delUser()方法，它可以向用户列表中添加新用户。该方法首先使用传递的 IP 参数到数据库中获取对应的用户对象，如果成功获取用户对象，说明数据库已存在该 IP 地址的用户，系统将使用对话框提示“用户已存在”，否则执行该 IP 地址的搜索任务。当确定该 IP 地址可以访问后，为该 IP 地址创建一个新的用户对象并添加到数据库中，然后调用 sortUsers()方法重新加载用户列表并提示用户添加成功。关键代码如下：

例程 38 代码位置：光盘\TM\02\EQ\src\com\lzw\userList\ChatTree.java

```
public boolean addUser(String ip, String opration) {
    try {
        if (ip == null)
            return false;
        User oldUser = dao.getUser(ip);                          //从数据库中获取 IP 相同的用户
        if (oldUser == null) {                                   //如果数据库中不存在该用户
            InetAddress addr = InetAddress.getByName(ip);
            if (addr.isReachable(1500)) {                        //如果该用户 IP 可以访问
                String host = addr.getHostName();                //获取它的主机名称
                User newUser = new User();                       //创建新用户对象
                newUser.setIp(ip);                               //初始化用户 IP
                newUser.setHost(host);                           //初始化用户主机名
                newUser.setName(host);                           //设置用户姓名
                newUser.setIcon("1.gif");                        //初始化用户头像
                dao.addUser(newUser);                            //添加该用户到数据库
                sortUsers();                                     //重新加载用户列表
                if (!opration.equals("search"))                  //如果不是系统自动搜索
                    JOptionPane.showMessageDialog(EQ.frame, "用户" + host
                            + "添加成功", "添加用户",
                            JOptionPane.INFORMATION_MESSAGE);    //弹出添加成功对话框
                return true;

            } else {                                             //如果该用户 IP 不可访问
                if (!opration.equals("search"))                  //并且不是系统自动搜索
                    JOptionPane.showMessageDialog(EQ.frame, "检测不到用户 IP："
                            + ip, "错误添加用户", JOptionPane.ERROR_MESSAGE);//对话框提示错误
                return false;
            }
        } else {                                                 //如果数据库存在该 IP 用户
            if (!opration.equals("search"))                      //并且不是系统自动搜索
                JOptionPane.showMessageDialog(EQ.frame, "已经存在用户 IP" + ip,
                        "不能添加用户", JOptionPane.WARNING_MESSAGE);//对话框提示用户已存在
            return false;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
```

2.8.4 单元测试

用户管理模块中定义了封装用户信息的 User 类，该类除了封装用户的 IP、主机名、姓名等属性外，还提供了获取用户头像的 getIconImg()方法，该方法返回的是 Icon 类的实例，可以直接添加到相应的控件中。如果 User 类出现故障，整个用户列表的界面将被破坏。为避免此类问题，需要编写 User 类的测试用例 UserTest 类。在程序调试和升级维护阶段，该用例将保证 User 类的正确性。UserTest 类的

关键代码如下：

```
❶  public class UserTest extends TestCase {
       private InetAddress local;                                    //定义本机地址对象
       private User user;                                            //定义用户对象
❷      public void setUp() throws Exception {
           local = InetAddress.getLocalHost();
           String ip = local.getHostAddress();                       //获取本机 IP
           String hostname = local.getHostName();                    //获取本机的主机名
           user = new User(hostname,ip);                             //使用本机信息初始化用户对象
           user.setName("新用户");                                    //设置用户姓名
           user.setTipText("我是本机用户");                           //设置用户提示文本
       }
       public void testGetIp() {                                     //测试封装的 IP
❸          assertNotNull("测试 IP==null",user.getIp());
❹          assertEquals("测试 IP", user.getIp(), local.getHostAddress());
       }
       public void testGetHost() {                                   //测试封装的主机名
           assertNotNull("测试主机名==null", user.getHost());
           assertEquals("测试主机名", user.getHost(), local.getHostName());
       }
       public void testGetIconImg() {                                //测试用户头像
           assertNotNull("测试头像==null", user.getIconImg());
       }
       public void testGetTipText() {                                //测试封装的提示文本
           assertNotNull("测试提示文本==null", user.getTipText());
           assertEquals("测试提示文本", "我是本机用户", user.getTipText());
       }
       public void testGetName() {                                   //测试封装的用户姓名
           assertNotNull("测试姓名==null", user.getName());
           assertEquals("测试姓名", "新用户", user.getName());
       }
   }
```

代码贴士

❶ TestCase：测试用例类，该类是 JUnit 的测试类，所有测试用例都需要继承该类。

❷ setUp()：测试之前调用的方法，该方法用于为所有测试提供数据。

❸ assertNotNull()：断言方法，该方法用于判断对象是否为 Null 值。

❹ assertEquals()：断言方法，该方法用于判断两个对象是否相等，即内容相等，而不是对象相同。

该单元测试的运行结果如图 2.15 所示，绿色的状态条说明 User 用户类可以正常工作，如果该状态条变成红色，说明 User 类存在缺陷，需要重新调试。

图 2.15　用户实例的单元测试

2.9 通信模块设计

2.9.1 通信模块概述

通信模块是企业内部通信系统的核心模块，它用于不同职工之间的通信，这种通信方式能够实现多个职工之间的通话，而不存在类似电话的占线问题，增加了任务分配的新方式，从而提高企业的工作效率。该模块可以使用 UDP 协议和系统信使两种方式发送通信信息。通信模块的界面运行效果如图 2.16 所示。

图 2.16 通信模块的界面运行效果

2.9.2 通信模块技术分析

通信模块使用基于 UDP 协议的数据报和套接字实现计算机之间的信息通信。UDP（User Datagram Protocol）协议就是用户数据报协议，它是一种无连接协议，在用该协议进行数据传输时，发送方只需要知道对方的 IP 地址和端口号就可以发送数据，并不需要进行连接，当连接的远程主机端口号处于监听状态时，则 UDP 必须处于连接状态。

Java 中对 UDP 数据报的发送和接收是通过 DatagramSocket 类实现的，DatagramPacket 类表示 UDP 数据包，它封装了数据报的属性和数据，这两个类的工作流程如图 2.17 所示。

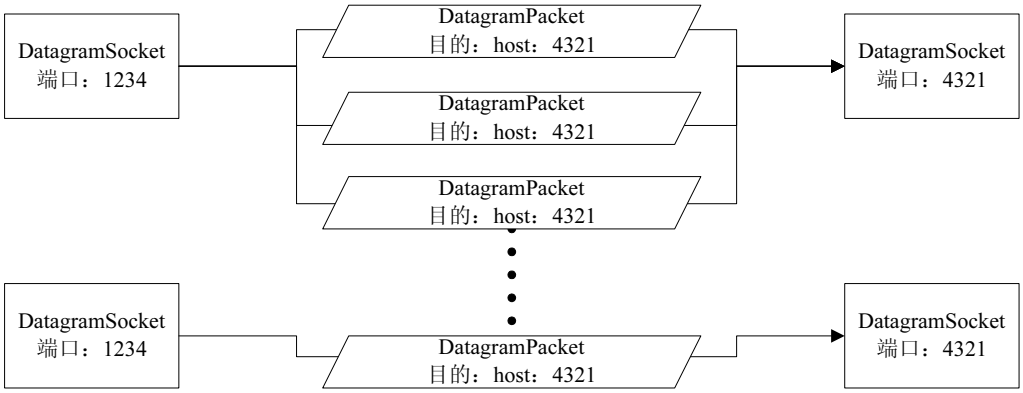


图 2.17 DatagramSocket 类和 DatagramPacket 类的工作流程

2.9.3 通信模块的实现过程

通信模块的开发步骤如下：

（1）创建 TelFrame 类，该类必须继承 JFrame 类实现 Swing 的窗体，在类的控件声明代码段中定

义窗体需要的所有控件，例如，接收信息的文本框、输入发送信息的文本框、“发送”按钮、信使按钮以及各种滚动面板等，另外还需要定义访问数据库的 Dao 实例，在通信窗口右侧显示的用户信息需要使用 Dao 实例从数据库中获取。关键代码如下：

例程 39　代码位置：光盘\TM\02\EQ\src\com\lzw\frame\TelFrame.java

```
public class TelFrame extends JFrame {
    private Dao dao;                                                    //数据库操作类
    private User user;                                                  //用户实例
    private JTextPane receiveText = new JTextPane();                    //信息接收文本框
    private JScrollPane scrollPane = new JScrollPane();                 //滚动面板
    private JTextPane sendText = new JTextPane();                       //信息编辑文本框
    private JScrollPane scrollPane_1 = new JScrollPane();               //滚动面板
    private JSplitPane splitPane = new JSplitPane();                    //分割面板
    private JButton sendButton = new JButton();                         //“发送”按钮
    private final JButton messageButton = new JButton();                //信使按钮
    private JPanel panel = new JPanel();
    private final static Map<String, TelFrame> instance = new HashMap<String, TelFrame>();//窗体实例集合
    private final JCheckBox messageMode = new JCheckBox();              //消息模式复选框
    private JToolBar toolBar = new JToolBar();                          //工具栏
    private JToggleButton toolFontButton = new JToggleButton();         //字体按钮
    private JButton toolFaceButton = new JButton();                     //表情按钮
    private final JScrollPane scrollPane_2 = new JScrollPane();         //滚动面板
    private final JLabel label_1 = new JLabel();
    private JPanel panel_3 = new JPanel();
    private byte[] buf;                                                 //数据缓冲
    private DatagramSocket ss;                                          //Socket
    private String ip;                                                  //IP 地址
    private DatagramPacket dp;                                          //数据报
    private TelFrame frame;                                             //窗体实例
    private ChatTree tree;                                              //用户列表实例
    private int rightPanelWidth = 148;                                  //右侧面板宽度
…//省略部分代码
}
```

（2）编写 getInstance()方法，用于获取唯一的窗体实例。该方法创建的所有窗体实例都会保存到 Map 集合类的实例中，除非退出企业内部通信系统，否则窗体的实例对象会一直保存在这个集合类中，并且用户再次打开已存在的窗体时，将直接从集合类中获取，不再创建新的窗体实例。关键代码如下：

例程 40　代码位置：光盘\TM\02\EQ\src\com\lzw\frame\TelFrame.java

```
public static synchronized TelFrame getInstance(DatagramSocket ssArg, DatagramPacket dp, ChatTree
treeArg) {
    String tmpIp = dp.getAddress().getHostAddress();                    //获取数据报的 IP 地址
❶   if (!instance.containsKey(tmpIp)) {                                 //如果集合中不存在该用户窗体
        TelFrame frame = new TelFrame(ssArg, dp, treeArg);              //创建窗体实例
❷       instance.put(tmpIp, frame);                                     //将窗体实例保存到集合中
        frame.receiveInfo(treeArg);                                     //接收信息
        if (!frame.isVisible()) {                                       //如果窗体处于隐藏状态
            frame.setVisible(true);                                     //显示窗体
```

```
            }
            frame.setState(JFrame.NORMAL);
            frame.toFront();                                    //将窗体放置在最前端
            return frame;
        } else {                                                //如果集合中包含该用户窗体
❸           TelFrame frame = instance.get(tmpIp);               //从集合中获取窗体实例
            frame.receiveInfo(treeArg);                         //接收信息
            if (!frame.isVisible()) {
                frame.setVisible(true);                         //显示窗体
            }
            frame.setState(JFrame.NORMAL);
            frame.toFront();
            return frame;
        }
    }
```

代码贴士

❶ containsKey()：该方法用于判断集合中是否包含指定名称的值。

❷ put()：该方法用于存储值对象到集合中，并指定一个名称。

❸ get()：该方法用于根据指定名称获取数据。

（3）在构造方法中初始化 TelFrame 类的所有控件属性，该构造方法接收 DatagramSocket、DatagramPacket 和 ChatTree 类的 3 个参数，它们分别是数据 Socket 服务、数据报和用户列表的实例对象，在通信窗体中需要使用它们。关键代码如下：

例程 41 代码位置：光盘\TM\02\EQ\src\com\lzw\frame\TelFrame.java

```
public TelFrame(DatagramSocket ssArg, DatagramPacket dpArg, final ChatTree treeArg) {
    this.tree = treeArg;                                        //获取用户列表对象
    ip = dpArg.getAddress().getHostAddress();                   //获取数据报中的 IP 地址
    user = dao.getUser(ip);                                     //获取该 IP 的用户对象
    dao = Dao.getDao();                                         //初始化 Dao 实例
    ss = ssArg;                                                 //获取数据服务对象
    dp = dpArg;                                                 //获取数据报
    buf = dp.getData();                                         //获取数据报中的数据
    setBounds(200, 100, 521, 424);                              //设置窗体位置和大小
    getContentPane().add(splitPane);                            //添加分割面板
    splitPane.setDividerSize(2);                                //设置面板分割线大小
    splitPane.setOrientation(JSplitPane.VERTICAL_SPLIT);        //定义面板竖向分割
    splitPane.setLeftComponent(scrollPane);                     //设置分割面板上半部的控件
    scrollPane.setViewportView(getReceiveText());               //添加接收文本框的滚动条
    receiveText.setMargin(new Insets(0, 0, 0, 0));              //设置接收文本框的边界
    receiveText.setEditable(false);                             //并使该文本框只读
    sendButton.addActionListener(new sendActionListener());     //添加“发送”按钮的事件监听器
    sendButton.setText("发送");                                 //设置按钮文本
    panel.add(messageButton);                                   //添加“信使”按钮
    messageButton.addActionListener(new MessageButtonActionListener());//添加“信使”按钮的事件监听器
    messageButton.setText("信使");                              //设置按钮文本
    sendText.addKeyListener(new SendTextKeyListener());         //添加“发送”文本框的事件监听器
```

```
        scrollPane_1.setViewportView(getSendText());                    //添加“发送”文本框的滚动条
        addWindowListener(new TelFrameClosing(tree));                   //添加窗体事件监听器
        scrollPane_2.setViewportView(label);
        String imgPath = EQ.class.getResource("/image/telFrameImage/telUserInfo.png")+ "";
        label.setText("<html><body background='" + imgPath
                + "'><table width='" + rightPanelWidth
                + "'><tr><td>用户名：<br>  " + user.getName()
                + "</td></tr><tr><td>主机名：<br>  " + user.getHost()
                + "</td></tr>" + "<tr><td>IP 地址：<br>  " + user.getIp()
                + "</td></tr><tr><td colspan='2' height="
                + this.getHeight() * 2
                + "></td></tr></table></body></html>");                 //设置窗体右侧的用户信息
        label_1.setIcon(new ImageIcon(EQ.class.getResource("/image/telFrameImage/telUserImage.png")));
        setTitle("与『" + user + "』通讯中");                            //设置窗体标题
        setVisible(true);                                               //显示窗体
    }
```

（4）编写 receiveInfo()方法，用于接收其他用户发送的通信信息。该方法首先从数据报中获取发送信息的用户 IP 地址，然后从数据库中获取该 IP 用户的姓名，同时将这些用户 IP、用户姓名和发送时间以蓝色字体显示到信息文本框中，最后将数据报中的通信信息文本追加到信息文本框中。关键代码如下：

例程 42　代码位置：光盘\TM\02\EQ\src\com\lzw\frame\TelFrame.java

```
private void receiveInfo(final ChatTree tree) {                         //接收信息
    if (buf.length > 0) {
        String rText = new String(buf).replace("" + (char) 0, "");
        String hostAddress = dp.getAddress().getHostAddress();
        String info = dao.getUser(hostAddress).getName();
        info = info + "    (" + new Date().toLocaleString() + ")";
        appendReceiveText(info, Color.BLUE);
        appendReceiveText(rText + "\n", null);
    }
}
```

（5）创建“发送”按钮的事件监听器 sendActionListener 类，它是一个内部类，并且该类实现了 ActionListener 接口，拥有处理按钮事件的能力。它在 actionPerformed()方法中获取数据报中的通信信息。关键代码如下：

例程 43　代码位置：光盘\TM\02\EQ\src\com\lzw\frame\TelFrame.java

```
class sendActionListener implements ActionListener {
    public void actionPerformed(final ActionEvent e) {
        String sendInfo = getSendInfo()                                 //获取发送信息
        if (sendInfo == null)
            return;
        insertUserInfoToReceiveText(tree);
        appendReceiveText(sendInfo + "\n", null);                       //添加到信息文本框
        byte[] tmpBuf = sendInfo.getBytes();
        DatagramPacket tdp = null;                                      //创建数据报
```

```
            try {
                tdp = new DatagramPacket(tmpBuf, tmpBuf.length,
                        new InetSocketAddress(ip, 1111));          //初始化数据报
                ss.send(tdp);                                      //发送数据报
            } catch (SocketException e2) {
                e2.printStackTrace();
            } catch (IOException e1) {
                e1.printStackTrace();
                JOptionPane.showMessageDialog(TelFrame.this, e1
                        .getMessage());
            }
            sendText.setText(null);                                //清空发送文本框
            sendText.requestFocus();                               //使发送文本框获得焦点
            if (messageMode.isSelected())                          //如果选择了“消息模式”复选框
                setState(ICONIFIED);                               //则窗体最小化
        }
    }
```

（6）创建 MessageButtonActionListener 内部类，该类是“信使”按钮的事件监听器。当用户输入通信信息，并单击“信使”按钮时，该事件监听器的 actionPerformed()方法会调用 Resource 工具类的 sendMessenger()方法将通信信息以系统信使方式发送到对方计算机上。关键代码如下：

例程 44 代码位置：光盘\TM\02\EQ\src\com\lzw\frame\TelFrame.java

```
private class MessageButtonActionListener implements ActionListener {
    public void actionPerformed(final ActionEvent e) {
        try {
            Document doc = sendText.getDocument();                 //获取文档对象
            String sendInfo = doc.getText(0, doc.getLength());     //获取发送的通信信息
            if (sendInfo.equals("") || sendInfo == null) {         //限制空信息的发送
                JOptionPane.showMessageDialog(TelFrame.this, "不能发送空信息。");
                return;
            }
            insertUserInfoToReceiveText(tree);
            appendReceiveText(sendInfo, null);
            Resource.sendMessenger(user, sendInfo, frame);         //调用 sendMessenger() 方法
            sendText.setText(null);                                //清空发送信息文本框
            sendText.requestFocus();                               //使文本框获得焦点
        } catch (BadLocationException e1) {
            e1.printStackTrace();
        }
    }
}
```

2.10 开发技巧与难点分析

Java 应用程序的资源（如图片、声音等）是应用程序不可缺少的组成部分，当程序需要设计窗体

背景、按钮图片、提示音等功能时需要获取这些资源，但是这些资源如果和程序的 JAR 打包文件分开存放，很容易因资源丢失而导致程序无法运行，或者无法显示界面。

如果将资源与程序存放在一起并打包到一个 JAR 文件中，则可以保证程序和资源的同步。这可以通过 Class 类的 getResource()方法实现获取资源的 URL 路径，然后使用 ImageIcon 类创建程序界面需要的背景图片或者按钮的图标文件，也可以通过该 URL 路径创建其他资源，如文本文件、声音文件等。获取资源路径的关键代码如下：

```
URL path = EQ.class.getResource("/NEWFACE/" + faceNum + ".png");          //获取资源路径
ImageIcon img = new ImageIcon(path);                                      //创建图片资源
```

资源文件的名称应该尽量使用英文，因为中文名称在打包成 JAR 文件后会产生乱码，那样即使路径正确，也无法获取文件名混乱的资源。

2.11　使用系统托盘

在 JDK 6.0 中，AWT 新增加了 SystemTray 类和 TrayIcon 类，它们可以在系统托盘区创建一个托盘程序。本节将介绍这两个类的具体使用方法。

SystemTray 类表示桌面右下角的系统托盘，系统托盘由运行在桌面上的所有应用程序共享，可以使用 getSystemTray()方法获取系统托盘对象，该方法总是对每个应用程序返回相同的实例。其语法格式如下：

```
SystemTray.getSystemTray()
```

并不是所有的操作系统都能支持系统托盘，当操作系统不存在或不支持系统托盘时，getSystemTray()方法将抛出 UnsupportedOperationException 异常。为避免程序出现错误，在获取系统托盘实例前，需要检查系统托盘是否被支持，这可以通过调用 isSupported()方法来实现。其语法格式如下：

```
SystemTray. isSupported()
```

SystemTray 类可以包含一个或多个 TrayIcon 类的实例，即系统栏图标，可以使用 add()方法将它们添加到托盘中，当不再需要托盘时，使用 remove()方法将其移除。TrayIcon 对象由图像、弹出菜单和一组相关监听器组成。创建系统托盘的关键代码如下：

例程 45　代码位置：光盘\TM\02\EQ\src\com\lzw\EQ.java

```
try {
    String title = "EQ 通讯软件";                                          //系统栏图标提示文本的标题
    String company = "吉林省×××科技有限公司";                              //系统栏图标提示文本
    SystemTray sysTray = SystemTray.getSystemTray();                      //获取系统托盘对象
    Image image = Toolkit.getDefaultToolkit().getImage(
            EQ.class.getResource("/icons/sysTray.png"));                  //系统栏图标
    trayicon = new TrayIcon(image, title + "\n" + company, createMenu()); //创建系统栏图标对象
    trayicon.setImageAutoSize(true);                                      //设置自动大小
```

```
        trayicon.addActionListener(new SysTrayActionListener());          //添加事件监听器
        sysTray.add(trayicon);                                            //添加系统栏图标到系统托盘
        trayicon.displayMessage(title, company, MessageType.INFO);        //显示系统栏图标提示文本
    } catch (Exception e) {
        e.printStackTrace();
    }
```

2.12 本 章 小 结

本章主要以 UDP 协议的通信方式，制作了一个企业内部通信系统，该系统可以实现在线通信、信使发送功能。通过对本章的学习，读者可以掌握 UDP 协议的通信方式，并学会如何在 Java 语言中执行外部命令、如何获取资源；另外，还可以学到 JavaDB 数据库和系统托盘两个新特性，希望读者能熟练掌握这些新技术，为以后的程序开发奠定基础。

第 3 章

企业人事管理系统

（Swing+Hibernate+Oracle 实现）

企业的发展不仅需要技术的竞争、市场的竞争、服务的竞争，还需要人才的竞争，并且成为市场竞争中一个重要的环节。优秀人才的引入将给企业的发展注入新鲜的血液，带给企业巨大的发展空间。所以，吸引人才，留住人才就成为了企业人事管理的一个重要课题。要想留住人才不仅需要企业具有良好的发展前景，更重要的是企业要有一个健全的管理体制，这不仅能节省企业大量的人力和物力，还可以提高企业的经济效益，从而带动企业快速发展。

通过阅读本章，可以学习到：

- 企业人事管理系统的软件结构和业务流程
- Oracle 数据库的使用方法
- 如何利用 Hibernate 建立持久层
- Swing 中表格行选取事件的使用方法
- Swing 中树状结构的使用和维护，如维护公司结构树
- Swing 中选取并显示图片的方法
- 通过 Java 反射验证数据是否为空
- 页面中组件联动的实现方法
- 如何开发一些简单、实用的功能模块，如支持树状结构的下拉菜单
- 如何在程序中调用其他工具软件，如计算器、Excel 等

3.1 开 发 背 景

飞速发展的技术变革和创新，以及迅速变化的差异化顾客需求等新竞争环境的出现，使得越来越多的组织通过构筑自身的人事竞争力来维持生存并促进持续发展。在“以人为本”观念的熏陶下，企业人事管理在组织中的作用日益突出。但是，人员的复杂性和组织的特有性使得企业人事管理成为难题。基于这个时代背景，企业人事管理便成为企业管理的重要内容。企业人事管理系统的作用之一是为企业的员工建立人事档案，它的出现使得人事档案查询、调用的速度加快，也使得精确分析大量员工的知识、经验、技术、能力和职业抱负成为可能，从而实现企业人事管理的标准化、科学化、数字化。

3.2 系 统 分 析

伴随着企业人事管理系统化的日益完善，企业人事管理系统在企业管理中越来越受到企业管理者的青睐。企业人事管理系统的功能全面、操作简单，可以快速地为员工建立电子档案，并且便于修改、保存和查看，实现了无纸化存档，为企业节省了大量的资金和空间。通过企业人事管理系统，还可以实现对企业员工的考勤管理、奖惩管理、培训管理、待遇管理和快速生成待遇报表。

3.3 系 统 设 计

3.3.1 系统目标

根据企业对人事管理的要求，本系统需要实现以下目标：

- ☑ 操作简单方便，界面简洁大方。
- ☑ 方便、快捷的档案管理。
- ☑ 简单、实用的考勤和奖惩管理。
- ☑ 简单、实用的培训管理。
- ☑ 针对企业中不同的待遇标准，实现待遇账套管理。
- ☑ 简单明了的账套维护功能。
- ☑ 方便、快捷的账套人员设置。
- ☑ 功能强大的待遇报表功能。
- ☑ 系统运行稳定、安全可靠。

3.3.2 系统功能结构

企业人事管理系统主要包括人事管理和待遇管理两大功能模块，用来提供对企业员工的人事和待遇管理；以及系统的辅助功能模块，包括系统维护和用户管理，用来提供对系统的维护和系统安全；

还包含一个系统工具模块，用来快速运行系统中的常用工具，如系统计算器和 Excel 表格等。

人事管理模块包含的子模块有档案管理、考勤管理、奖惩管理和培训管理。其中，档案管理模块用来维护员工的基本信息，包括档案信息、职务信息和个人信息。其中，档案信息包括员工的照片。档案信息只可以添加和修改，不可以删除，因为员工档案将作为企业的永久资源和历史记录进行保存。在维护员工档案时，可以通过企业结构树快速查找员工。考勤管理模块用来记录员工的考勤信息，如迟到、请假、加班等。奖惩管理模块用来记录员工的奖惩信息，如因为某事情奖励或惩罚员工。培训管理模块用来记录对员工的培训信息。

待遇管理模块包含的子模块有账套管理、 人员设置和统计报表。其中，账套管理模块用来建立和维护账套。所谓账套，就是对不同员工采用不同的待遇标准。例如，对已经签订劳动合同的员工和处于试用期的员工的基本工资是不同的，针对这种情况可以分别建立一个试用期账套和合同工账套。这里假设处于试用期的员工的基本工资为 2000，而已经签订劳动合同的员工的基本工资为 3000，则可以分别将试用期账套和合同工账套中的基本工资项设为 2000 和 3000。账套中的部分项目可以用于考勤管理模块的考勤项目。人员设置模块用来设置对员工采用前面建立的哪个账套，即采用哪个待遇标准，如果没有适合的账套，则可以继续建立新的账套。统计报表模块将以表格的形式统计员工的待遇情况，这里将用到在考勤管理和奖惩管理模块填写的数据，可以按月、季度、半年和年统计。

系统维护模块包含的子模块有企业架构、基本资料和初始化系统。其中，企业架构模块用来维护企业的组织结构，企业架构将以树状结构显示；基本资料模块用来维护职务种类、用工形式、账套项目、考勤项目、民族和籍贯信息；初始化系统模块用来对系统进行初始化，在正式使用前需要对系统进行初始化。

用户管理模块包含的子模块有新增用户和修改密码。其中，新增用户模块用来添加和维护系统的管理员，包括冻结和删除管理员，该模块只有超级管理员有权使用；修改密码模块用来为当前登录用户修改登录密码。

系统工具模块包含的功能有打开计算器、打开 Word 和打开 Excel，以方便用户快速地打开这 3 个常用的系统工具。

企业人事管理系统的功能结构如图 3.1 所示。

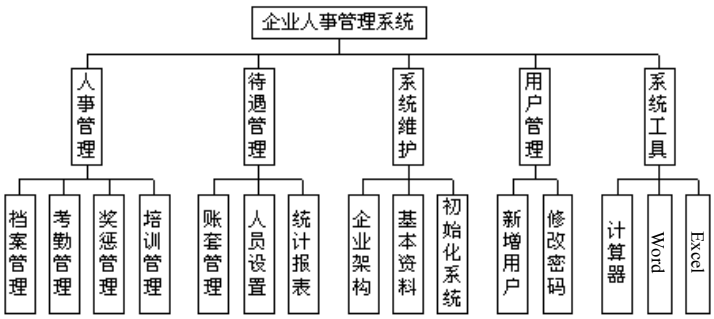


图 3.1　企业人事管理系统功能结构

3.3.3　系统预览

企业人事管理系统由多个界面组成，下面仅列出几个典型界面，其他界面效果可参见光盘中的源程序。

企业人事管理系统的主窗体效果如图 3.2 所示，窗体的左侧为系统的功能结构导航，窗体的上方为系统常用功能的快捷按钮。

单击图 3.2 中左侧导航栏中的“档案管理”节点，将打开如图 3.3 所示的档案列表界面。单击界面上方的“新建员工档案”按钮，可以建立新的员工档案；单击左侧的企业架构树中的部门节点，在右侧将显示相应部门的员工列表，首先选中其中的一行，然后单击界面上方的“修改员工档案”按钮，

可以修改选中员工的档案。

图 3.2　企业人事管理系统主窗体效果
（光盘\…\IndexFrame.java）

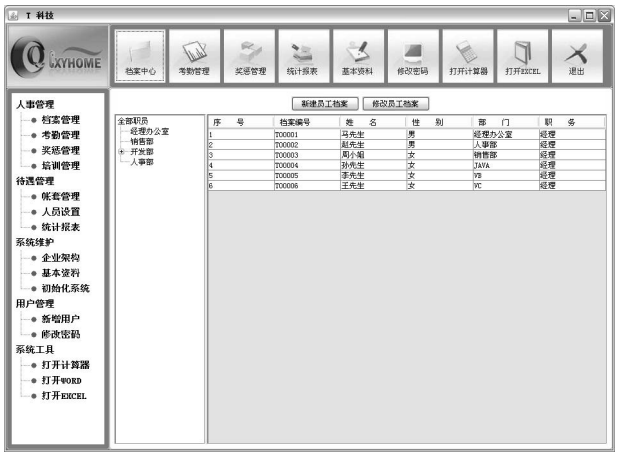


图 3.3　档案列表界面（光盘\…\
personnel\RecordSelectedPanel.java）

单击图 3.2 中左侧导航栏中的“培训管理”节点，将打开如图 3.4 所示的培训管理界面。在该界面可以建立培训信息，以及设置参训人员列表。

单击图 3.2 中左侧导航栏中的“账套管理”节点，将打开如图 3.5 所示的账套管理界面。在该界面可以维护账套信息，主要包括建立账套、添加或删除账套项目，以及修改项目金额。

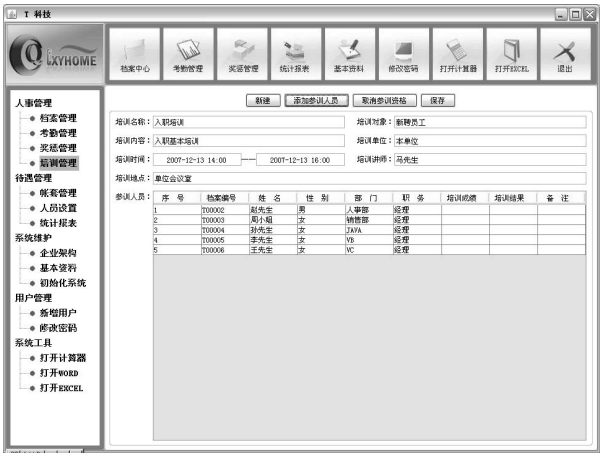


图 3.4　培训管理界面（光盘\…\personnel\
BringUpOperatePanel.java）

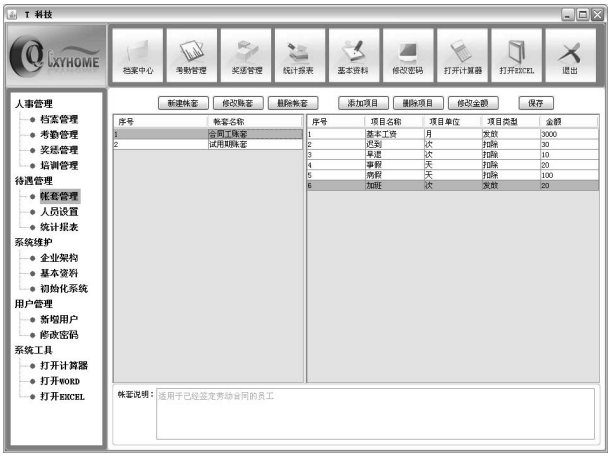


图 3.5　账套管理界面（光盘\…\treatement\
CriterionSetPanel.java）

单击图 3.2 中左侧导航栏中的“统计报表”节点，将打开如图 3.6 所示的统计报表界面。在该界面可以生成统计报表，可以生成的报表种类有月报表、季度报表、半年报表和年度报表。

说明　由于路径太长，因此省略了部分路径，省略的路径是“TM\03\PersonnalManage\src\com\mwq\frame”。

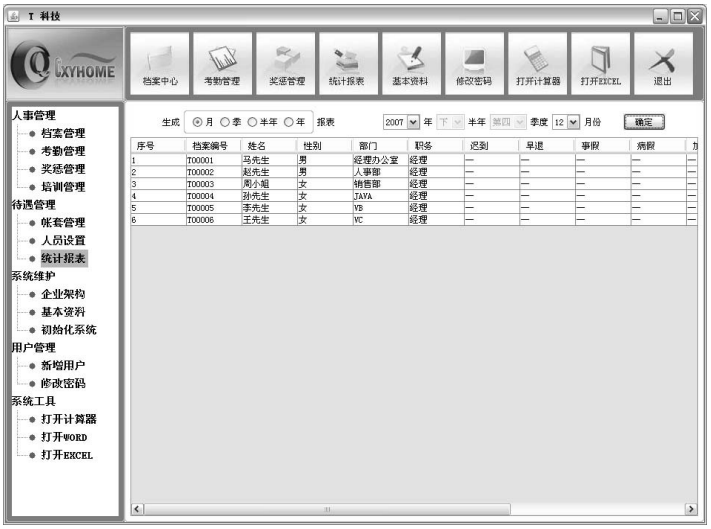


图 3.6　统计报表界面（光盘\…\treatement\ReportFormsPanel.java）

3.3.4　业务流程图

企业人事管理系统的业务流程如图 3.7 所示。

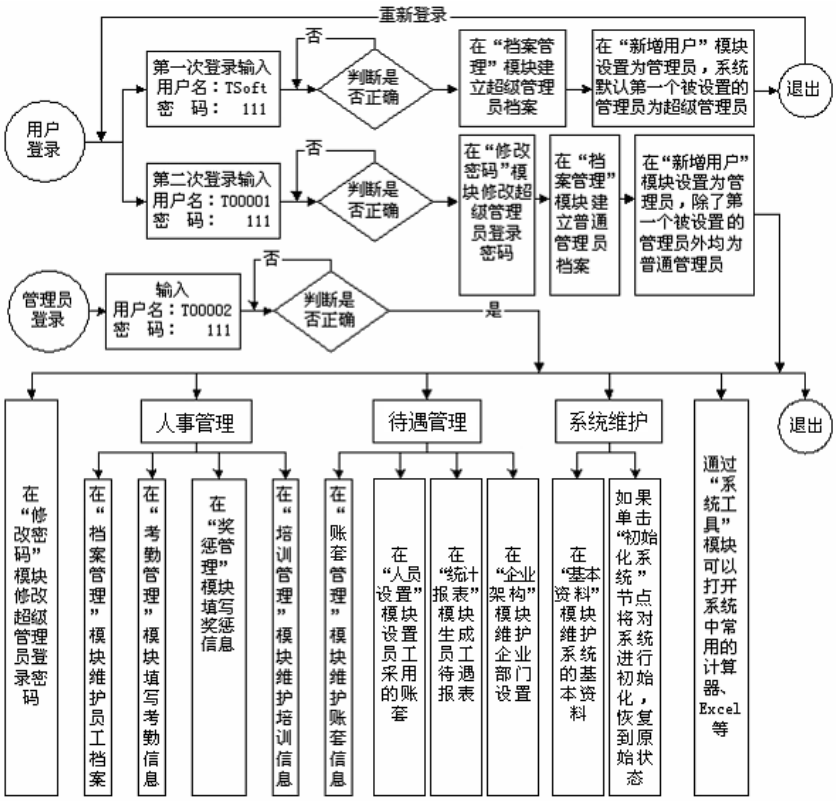


图 3.7　企业人事管理系统的业务流程图

3.3.5 文件夹结构设计

每个项目都会有相应的文件夹组织结构。当项目中的窗体过多时，为了便于查找和使用，可以将窗体进行分类，放入不同的文件夹中，这样既便于前期开发，又便于后期维护。企业人事管理系统文件夹组织结构如图 3.8 所示。

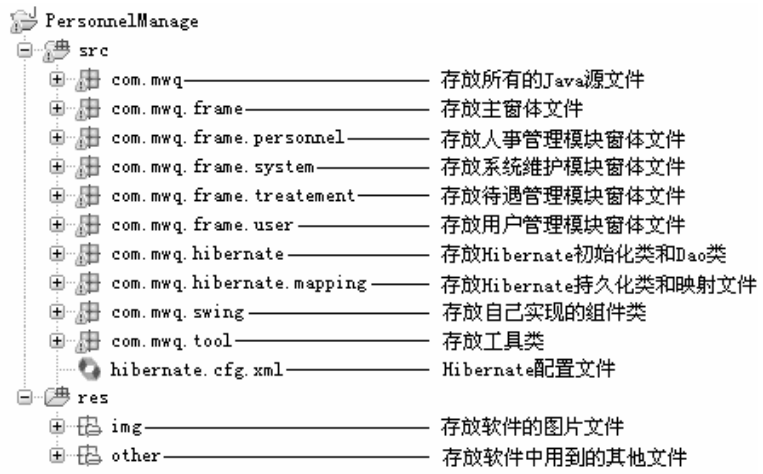


图 3.8 企业人事管理系统文件夹结构图

3.4 数据库设计

在开发应用程序时，对数据库的操作是必不可少的，而一个数据库的设计优秀与否，将直接影响到软件的开发进度和性能，所以对数据库的设计就显得尤为重要。数据库的设计要根据程序的需求及其功能制定，如果在开发软件之前不能很好地设计数据库，在开发过程中将反复修改数据库，这将严重影响开发进度。

3.4.1 数据库分析

企业人事管理系统的需求主要包括对人事档案的管理，其中包括档案信息、职务信息和个人信息；人事考勤、奖惩、培训的管理，并且考勤和奖惩信息将体现到待遇统计当中；待遇管理，还要针对企业的现实需求，要求企业人事管理系统支持多账套功能。

3.4.2 数据库概念设计

数据库设计是系统设计过程中的重要组成部分，它是通过管理系统的整体需求而制定的，数据库设计的好坏直接影响到系统的后期开发。下面对本系统中具有代表性的数据库设计进行详细说明。

在开发企业人事管理系统时，最重要的是人事档案信息。本系统将档案信息又分为档案信息、职

务信息和个人信息，由于信息多而复杂，这里只给出关键的信息。档案信息表的 E-R 图如图 3.9 所示。

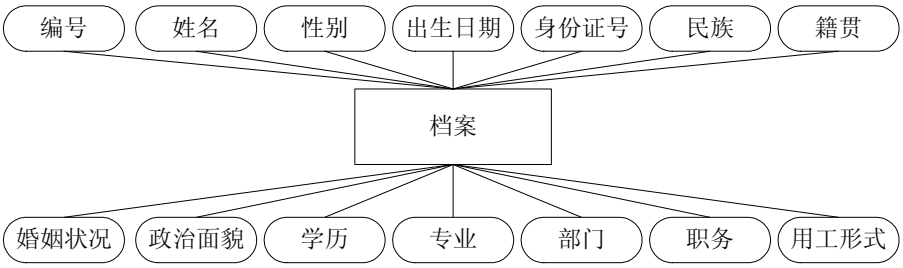


图 3.9　档案信息表 E-R 图

本系统提供了人事考勤记录和人事奖惩记录功能，这里只给出人事考勤信息表的 E-R 图，如图 3.10 所示。

根据企业人事管理中的现实需求，本系统提供了多账套管理功能，通过这一功能，可以很方便地对各种类型的员工实施不同的待遇标准。账套信息表的 E-R 图如图 3.11 所示。

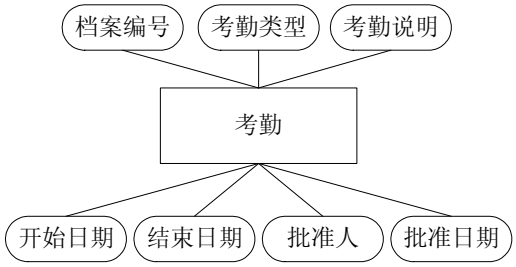


图 3.10　考勤信息表 E-R 图

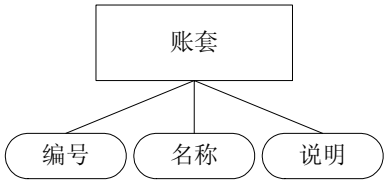


图 3.11　账套信息表 E-R 图

每个账套都要包含多个账套项目，这些账套中的项目可以有零个或多个是不同的，区别是每个账套项目的金额是不同的。账套项目信息表的 E-R 图如图 3.12 所示。

建立多账套是为了实现对员工按照不同的待遇标准进行管理，所以要将员工设置到不同的账套中，即表示对该员工实施相应的待遇标准。账套设置信息表的 E-R 图如图 3.13 所示。

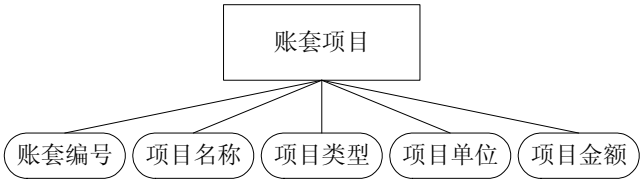


图 3.12　账套项目信息表 E-R 图

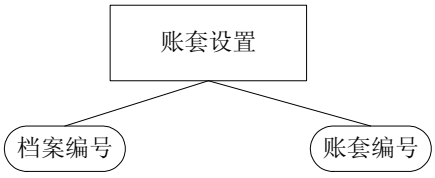


图 3.13　账套设置信息表 E-R 图

3.4.3　数据库逻辑结构设计

数据库概念设计中已经分析了档案、考勤和账套等主要的数据实体对象，这些实体对象是数据表结构的基本模型，最终的数据模型都要实施到数据库中，形成整体的数据结构。可以使用 PowerDesigner 工具完成这个数据库的建模，其模型结构如图 3.14 所示。

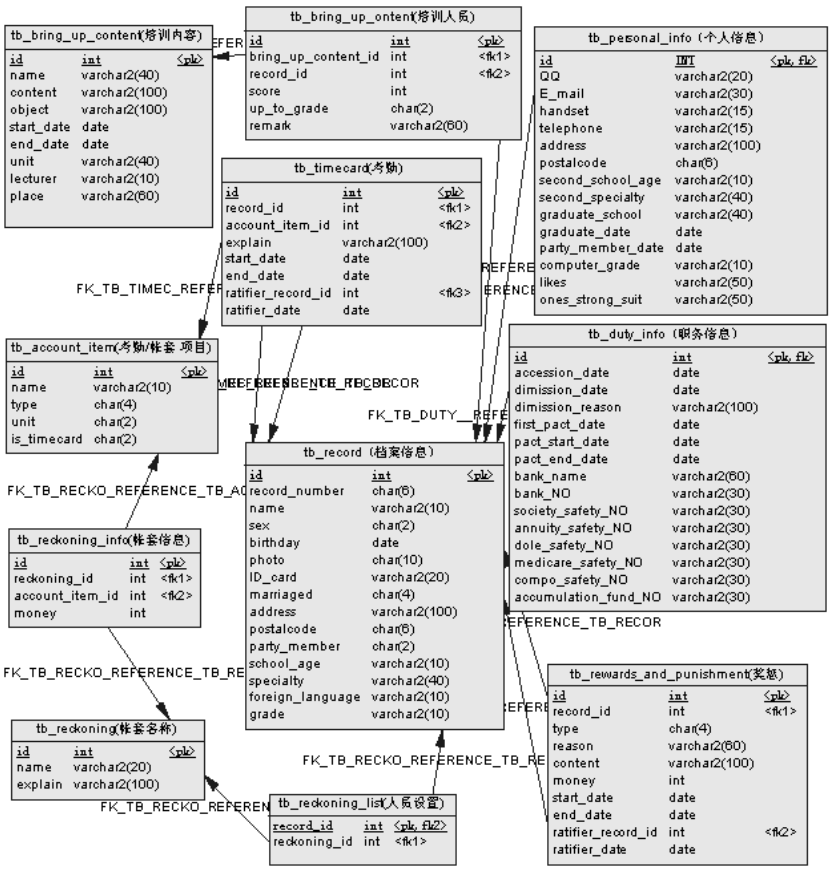


图 3.14　企业人事管理系统数据库模型

3.5　主窗体设计

主窗体是软件系统的一个重要组成部分，是提供人机交互的一个必不可少的操作平台。通过主窗体，用户可以打开与系统相关的各个子操作模块，完成对软件的操作和使用，另外通过主窗体，用户还可以快速掌握本系统的基本功能。

3.5.1　导航栏的设计

通过本系统的导航栏，可以打开本系统的所有子模块。导航栏的效果如图 3.15 所示。

本系统的导航栏是通过树组件实现的，在这里不显示树的根节点，并且打开软件时树结构是展开的，还设置在叶子节点折叠和展开时均不采用图标。

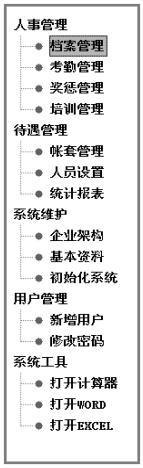


图 3.15　导航栏效果

下面的代码将通过树节点对象创建一个树结构，最后创建一个树模型对象。

例程 01　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\ IndexFrame.java

```
DefaultMutableTreeNode root = new DefaultMutableTreeNode("root");                    //创建树的根节点
DefaultMutableTreeNode personnelNode = new DefaultMutableTreeNode("人事管理");    //创建树的一级子节点
personnelNode.add(new DefaultMutableTreeNode("档案管理"));      //创建树的叶子节点并添加到一级子节点
personnelNode.add(new DefaultMutableTreeNode("考勤管理"));
personnelNode.add(new DefaultMutableTreeNode("奖惩管理"));
personnelNode.add(new DefaultMutableTreeNode("培训管理"));
root.add(personnelNode);                                                             //向根节点添加一级子节点
DefaultMutableTreeNode treatmentNode = new DefaultMutableTreeNode("待遇管理");
treatmentNode.add(new DefaultMutableTreeNode("账套管理"));
treatmentNode.add(new DefaultMutableTreeNode("人员设置"));
treatmentNode.add(new DefaultMutableTreeNode("统计报表"));
root.add(treatmentNode);
DefaultMutableTreeNode systemNode = new DefaultMutableTreeNode("系统维护");
systemNode.add(new DefaultMutableTreeNode("企业架构"));
systemNode.add(new DefaultMutableTreeNode("基本资料"));
systemNode.add(new DefaultMutableTreeNode("初始化系统"));
root.add(systemNode);
DefaultMutableTreeNode userNode = new DefaultMutableTreeNode("用户管理");
if (record == null) {          //当 record 为 null 时，说明是通过默认用户登录的，此时只能新增用户，不能修改密码
    userNode.add(new DefaultMutableTreeNode("新增用户"));
} else {                                                                             //否则为通过管理员登录
    String purview = record.getTbManager().getPurview();
    if (purview.equals("超级管理员")) {              //只有当管理员的权限为“超级管理员”时，才有权新增用户
        userNode.add(new DefaultMutableTreeNode("新增用户"));
    }
    userNode.add(new DefaultMutableTreeNode("修改密码"));      //只有通过管理员登录时才有权修改密码
}
root.add(userNode);
DefaultMutableTreeNode toolNode = new DefaultMutableTreeNode("系统工具");
toolNode.add(new DefaultMutableTreeNode("打开计算器"));
toolNode.add(new DefaultMutableTreeNode("打开 WORD"));
toolNode.add(new DefaultMutableTreeNode("打开 EXCEL"));
root.add(toolNode);
DefaultTreeModel treeModel = new DefaultTreeModel(root);      //通过树节点对象创建树模型对象
```

下面的代码将利用在例程01中创建的树模型对象创建一个树对象，并设置树对象的相关绘制属性。

例程 02　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\IndexFrame.java

```
    tree = new JTree(treeModel);                                                     //通过树模型对象创建树对象
    tree.setBackground(Color.WHITE);                                                 //设置树的背景色
❶   tree.setRootVisible(false);                                                     //设置不显示树的根节点
    tree.setRowHeight(28);                                                           //设置各节点的高度为 28 像素
    Font font = new Font("宋体", Font.BOLD, 16);
    tree.setFont(font);                                                              //设置节点的字体样式
    DefaultTreeCellRenderer renderer = new DefaultTreeCellRenderer();                //创建一个树的绘制对象
    renderer.setClosedIcon(null);                                                    //设置节点折叠时不采用图标
```

```
        renderer.setOpenIcon(null);                                          //设置节点展开时不采用图标
        tree.setCellRenderer(renderer);                                      //将树的绘制对象设置到树中
        int count = root.getChildCount();                                    //获得一级节点的数量
        for (int i = 0; i < count; i++) {                                    //遍历树的一级节点
            DefaultMutableTreeNode node = (DefaultMutableTreeNode) root.getChildAt(i); //获得指定索引的一级节点
对象
            TreePath path = new TreePath(node.getPath());                    //获得节点对象的路径
❷           tree.expandPath(path);                                           //展开该节点
        }
❸           tree.addTreeSelectionListener(new TreeSelectionListener() {      //捕获树的选取事件
            public void valueChanged(TreeSelectionEvent e) {
                …//由于篇幅有限，此处省略了处理捕获事件的具体代码，详见光盘源代码
            }
        });
        leftPanel.add(tree);                                                 //将树添加到面板组件中
```

代码贴士

❶ setRootVisible(boolean b)：该方法用于设置是否显示树的根节点，默认为显示根节点，即默认为 true；如果设置为 false，则不显示树的根节点。

❷ expandPath(TreePath path)：该方法用于展开指定路径的节点。

❸ addTreeSelectionListener(TreeSelectionListener listener)：该方法用于为树添加捕获树节点被选中和取消的事件。

3.5.2 工具栏的设计

为了方便用户使用系统，在工具栏上为常用的系统子模块提供了快捷按钮，通过这些按钮，用户可以快速地进入系统中常用的子模块。工具栏的效果如图 3.16 所示。

图 3.16　工具栏效果

下面的代码将创建一个用来添加快捷按钮的面板，并且为面板设置了边框，面板的布局管理器为水平箱式布局。

例程 03　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\IndexFrame.java

```
final JPanel buttonPanel = new JPanel();                                 //创建工具栏面板
final GridLayout gridLayout = new GridLayout(1, 0);                      //创建一个水平箱式布局管理器对象
gridLayout.setVgap(6);                                                   //箱的垂直间隔为 6 像素
gridLayout.setHgap(6);                                                   //箱的水平间隔为 6 像素
buttonPanel.setLayout(gridLayout);                                       //设置工具栏面板采用的布局管理器为箱式布局
buttonPanel.setBackground(Color.WHITE);                                  //设置工具栏面板的背景色
buttonPanel.setBorder(new TitledBorder(null, "",TitledBorder.DEFAULT_JUSTIFICATION,
        TitledBorder.DEFAULT_POSITION, null, null));                     //设置工具栏面板采用的边框样式
topPanel.add(buttonPanel, BorderLayout.CENTER);                          //将工具栏面板添加到上级面板中
```

在工具栏上提供了用来快速打开“档案管理”、“考勤管理”、“奖惩管理”、“统计报表”、

“基本资料”和“修改密码”子模块的按钮，以及“打开计算器”和“打开 EXCEL”两个打开常用系统工具的按钮，还有一个用来快速退出系统的“退出”按钮。这些快捷按钮的实现代码基本相同，所以这里只给出“档案管理”快捷按钮的实现代码。关键代码如下：

例程 04　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\IndexFrame.java

```
final JButton recordShortcutKeyButton = new JButton();          //创建进入“档案管理”的快捷按钮
//为按钮添加事件监听器，用来捕获按钮被单击的事件
recordShortcutKeyButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        rightPanel.removeAll();                                  //移除内容面板中的所有内容
        rightPanel.add(new RecordSelectedPanel(rightPanel),
                BorderLayout.CENTER);                            //将“档案管理”面板添加到内容面板中
        SwingUtilities.updateComponentTreeUI(rightPanel); //刷新内容面板中的内容
    }
});
recordShortcutKeyButton.setText("档案管理");
buttonPanel.add(recordShortcutKeyButton);
```

在实现修改密码时，需要判断当前的登录用户，如果用户是通过系统的默认用户登录的，则不允许修改密码，需要把“修改密码”按钮设置为不可用。关键代码如下：

例程 05　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\IndexFrame.java

```
final JButton updatePasswordShortcutKeyButton = new JButton();
if (record == null)                    //当 record 为 null 时，说明是通过默认用户登录的，此时不能修改密码
    updatePasswordShortcutKeyButton.setEnabled(false);          //在这种情况下设置按钮为不可用
updatePasswordShortcutKeyButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        rightPanel.removeAll();
        SwingUtilities.updateComponentTreeUI(rightPanel);
        UpdatePasswordDialog dialog = new UpdatePasswordDialog();      //创建用来修改密码的对话框
        dialog.setRecord(record);                                      //将当前登录管理员的档案对象传入对话框
        dialog.setVisible(true);                                       //设置对话框为可见的，即显示对话框
    }
});
updatePasswordShortcutKeyButton.setText("修改密码");
buttonPanel.add(updatePasswordShortcutKeyButton);
```

通过 java.awt.Desktop 类的 open(File file)方法，可以运行系统中的其他软件，例如运行系统计算器。为了方便用户使用系统计算器和 Excel，本系统提供了“打开计算器”和“打开 EXCEL”两个按钮。这两个按钮的实现代码基本相同，下面将以打开系统计算器为例，讲解如何在 Java 程序中打开其他软件。关键代码如下：

例程 06　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\IndexFrame.java

```
final JButton counterShortcutKeyButton = new JButton();
counterShortcutKeyButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Desktop desktop = Desktop.getDesktop();                    //获得当前系统对象
```

```
            File file = new File("C:/WINDOWS/system32/calc.exe");          //创建一个系统计算器对象
            try {
                desktop.open(file);                                        //打开系统计算器
            } catch (Exception e1) {                                       //当打开失败时，弹出提示信息
                JOptionPane.showMessageDialog(null, "很抱歉，未能打开系统自带的计算器！",
                        "友情提示", JOptionPane.INFORMATION_MESSAGE);
                return;
            }
        }
});
counterShortcutKeyButton.setText("打开计算器");
buttonPanel.add(counterShortcutKeyButton);
```

最后创建一个用来快速退出系统的“退出”按钮。关键代码如下：

例程 07 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\IndexFrame.java

```
final JButton exitShortcutKeyButton = new JButton();
exitShortcutKeyButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.exit(0);                                                    //退出系统
    }
});
exitShortcutKeyButton.setText("退出");
buttonPanel.add(exitShortcutKeyButton);
```

3.6 公共模块设计

公共模块的设计是软件开发的一个重要组成部分，它既起到了代码重用的作用，又起到了规范代码结构的作用，尤其在团队开发的情况下，是解决重复编码的最好方法，这样对软件的后期维护也起到了积极的作用。

3.6.1 编写 Hibernate 配置文件

在 Hibernate 配置文件中包含了两方面的内容，一方面是连接数据库的基本信息，例如连接数据库的驱动程序、URL、用户名、密码等；另一方面是 Hibernate 的配置信息，例如配置数据库使用的方言、持久化类映射文件等，还可以配置是否在控制台输出 SQL 语句，以及是否对输出的 SQL 语句进行格式化和添加提示信息等。本系统使用的 Hibernate 配置文件的关键代码如下：

例程 08 代码位置：光盘\mr\03\PersonnelManage\src\hibernate.cfg.xml

```
<property name="connection.driver_class">                    <!-- 配置数据库的驱动类 -->
    com.microsoft.jdbc.sqlserver.SQLServerDriver
</property>
<property name="connection.url">                             <!-- 配置数据库的连接路径 -->
    jdbc:sqlserver://localhost:1433;databasename=db_PersonnelManage
```

```
</property>
<property name="connection.username">sa</property>          <!-- 配置数据库的连接用户名 -->
❶    <property name="connection.password"></property>    <!-- 配置数据库的连接密码-->
<property name="dialect">                                  <!-- 配置数据库使用的方言 -->
      org.hibernate.dialect.SQLServerDialect
</property>
❷    <property name="show_sql">true</property>             <!-- 配置在控制台显示 SQL 语句 -->
<property name="format_sql">true</property>                <!-- 配置对输出的 SQL 语句进行格式化 -->
<property name="use_sql_comments">true</property>  <!-- 配置在输出的 SQL 语句前面添加提示信息 -->
❸    <mapping resource="com/mwq/hibernate/mapping/TbDept.hbm.xml" />  <!-- 配置持久化类映射文件 -->
```

代码贴士

❶ connection.password：该属性用来配置连接数据库的密码，这里密码为空，在这种情况下也可以省略该行配置代码，这里加上是为了讲解之用。

❷ show_sql：该属性用来配置是否在控制台输出 SQL 语句，默认为 false，即不输出。建议在调试程序时将该属性以及 format_sql 和 use_sql_comments 属性同时设置为 true，这样可以帮助快速找出错误原因，但是在发布程序之前，一定要将这 3 个属性再设置为 false（也可以删除这 3 行配置代码，因为它们的默认值均为 false，笔者推荐删除），这样做的好处是节省了格式化、注释和输出 SQL 语句的时间，从而提高了软件的性能。

❸ resource：该属性用来配置持久化类映射文件，每一个持久化类都要做这样的映射，由于篇幅有限，这里只给出了一行，详见光盘源代码。

3.6.2　编写 Hibernate 持久化类和映射文件

持久化类是数据实体对象的表现形式，通常情况下持久化类与数据表是相互对应的，它们通过持久化类映射文件建立映射关系。持久化类不需要实现任何类和接口，只需要提供一些属性及其对应的 set/get()方法。需要注意的是，每一个持久化类都需要提供一个没有入口参数的构造方法。

下面是持久化类 TbRecord 的部分代码，为了节省篇幅，这里只给出了两个具有代表性的属性，其中属性 id 为主键。

例程 09　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\mapping\TbRecord.java

```
public class TbRecord {
    public TbRecord () {
    }
    private int id;
    private String name;
    public void setId(int id) {
        this.id = id;
    }
    public int getId() {
        return id;
    }
    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
```

```
    }
}
```

下面是与持久化类 TbRecord 对应的映射文件 TbRecord.hbm.xml 的相应代码，持久化类映射文件负责建立持久化类与对应数据表之间的映射关系。

例程 10 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\mapping\TbRecord.hbm.xml

```
❶   <class name="com.mwq.hibernate.mapping.TbRecord" table="tb_record"
        schema="dbo" catalog="db_PersonnelManage">
❷       <id name="id" type="java.lang.Integer">
            <column name="id" />
❸           <generator class="increment" />
        </id>
❹       <property name="name" type="java.lang.String">
❺           <column name="name" length="10" not-null="true" />
        </property>
    </class>
```

代码贴士

❶ <class>：该元素用来配置持久化类与数据表之间的映射关系，其中 name 属性为持久化类名称，table 属性为数据表名称，schema 属性为数据表所属用户的权限，catalog 属性为数据表所在的数据库名称。

❷ <id>：该元素用来配置主键的映射关系，其中 name 属性为持久化类中属性的名称，type 属性为持久化类中属性的类型。

❸ <generator>：该元素用来配置主键的生成方式，当将 class 属性设置为 increment 时，表示采用 Hibernate 自增。

❹ <property>：该元素用来配置属性的映射关系，其中 name 属性为持久化类中属性的名称，type 属性为持久化类中属性的类型。

❺ <column>：该元素用来配置与持久化类中的属性对应的列，其中 name 属性为列的名称，length 属性为值的最大长度，not-null 属性为是否允许为空，当设为 true 时表示不允许为空，默认为 false。

3.6.3 编写通过 Hibernate 操作持久化对象的常用方法

对数据库的操作，离不开增、删、改、查，所以在这里也离不开实现这些功能的方法。不过在这里还针对 Hibernate 的特点，实现了两个具有特殊功能的方法，一个是用来过滤关联对象集合的方法，另一个是用来批量删除记录的方法。下面只介绍这两个方法和一个删除单个对象的方法，其他参见光盘源代码。

下面的方法是用来过滤一对多关联中 Set 集合中对象的方法，这是 Hibernate 提供的一个非常实用的集合过滤功能，通过该功能可以从关联集合中检索出符合指定条件的对象，检索条件可以是所有合法的 HQL 语句。关键代码如下：

例程 11 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\BaseDao.java

```
public List filterSet(Set set, String hql) {
    Session session = HibernateSessionFactory.getSession();              //获得 Session 对象
    Query query = session.createFilter(set, hql); //通过 session 对象的 createFilter()方法按照 hql 条件过滤 set
集合
    List list = query.list();                                            //执行过滤，返回值为 List 型结果集
```

```
    return list;                                                    //返回过滤结果
}
```

下面的方法用来删除指定的持久化对象。关键代码如下：

例程 12　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\BaseDao.java

```
public boolean deleteObject(Object obj) {
    boolean isDelete = true;                                        //默认删除成功
    Session session = HibernateSessionFactory.getSession();         //获得 Session 对象
    Transaction tr = session.beginTransaction();                    //开启事务
    try {
        session.delete(obj);                                        //删除指定的持久化对象
        tr.commit();                                                //提交事务
    } catch (HibernateException e) {
        isDelete = false;                                           //删除失败
        tr.rollback();                                              //回退事务
        e.printStackTrace();
    }
    return isDelete;
}
```

下面的方法用来批量删除对象，通过这种方法删除对象，每次只需要执行一条 SQL 语句。关键代码如下：

例程 13　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\BaseDao.java

```
public boolean deleteOfBatch(String hql) {
    boolean isDelete = true;                                        //默认删除成功
    Session session = HibernateSessionFactory.getSession();         //获得 Session 对象
    Transaction tr = session.beginTransaction();                    //开启事务
    try {
        Query query = session.createQuery(hql);                     //预处理 HQL 语句，获得 Query 对象
        query.executeUpdate();                                      //执行批量删除
        tr.commit();                                                //提交事务
    } catch (HibernateException e) {
        isDelete = false;                                           //删除失败
        tr.rollback();                                              //回退事务
        e.printStackTrace();
    }
    return isDelete;
}
```

3.6.4　创建用于特殊效果的部门树对话框

在系统中有多处需要填写部门，如果通过 JComboBox 组件提供部门列表，则不能够体现出企业的组织架构，用户在使用过程中也不是很直观和方便，因此开发了一个用于特殊效果的部门树对话框，

例如在新建档案时需要填写部门，利用该对话框实现的效果如图 3.17 所示。

图 3.17　用于特殊效果的部门树对话框

用户在填写部门时，只需要单击文本框后的按钮，就会弹出一个用来选取部门的部门树对话框，并且这个对话框显示在文本框和按钮的正下方，通过这种方法实现对部门的选取，对于用户将更加直观和方便。

从图 3.17 可以看出，这里用来选取部门的部门树对话框不需要提供标题栏，并且建议令这个对话框阻止当前线程，这样做的好处是可以强制用户选取部门，并且可以及时地销毁对话框，释放其占用的资源。实现这两点的关键代码如下：

例程 14　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\common\DeptTreeDialog.java

```
setModal(true);                                    //设置对话框阻止当前线程
setUndecorated(true);                              //设置对话框不提供标题栏
```

下面开始创建部门树。首先创建树节点对象，包括根节点及其子节点，并将子节点添加到上级节点中，然后利用根节点对象创建树模型对象，最后利用树模型对象创建树对象。当树节点超过一定数量时，树结构的高度可能大于对话框的高度，所以要将部门树放在滚动面板当中。关键代码如下：

例程 15　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\common\DeptTreeDialog.java

```
final JScrollPane scrollPane = new JScrollPane();                    //创建滚动面板
getContentPane().add(scrollPane, BorderLayout.CENTER);
TbDept company = (TbDept) dao.queryDeptById(1);
DefaultMutableTreeNode root = new DefaultMutableTreeNode(company.getName());      //创建部门树的根节点
Set depts = company.getTbDepts();
for (Iterator deptIt = depts.iterator(); deptIt.hasNext();) {
    TbDept dept = (TbDept) deptIt.next();
    DefaultMutableTreeNode deptNode = new DefaultMutableTreeNode(dept.getName());//创建部门树的二级子节点
    root.add(deptNode);
    Set sonDepts = dept.getTbDepts();
    for (Iterator sonDeptIt = sonDepts.iterator(); sonDeptIt.hasNext();) {
        TbDept sonDept = (TbDept) sonDeptIt.next();
        deptNode.add(new DefaultMutableTreeNode(sonDept.getName())); //创建部门树的叶子节点
    }
}
DefaultTreeModel treeModel = new DefaultTreeModel(root);             //利用根节点对象创建树模型对象
tree = new JTree(treeModel);                                         //利用树模型对象创建树对象
scrollPane.setViewportView(tree);                                    //将部门树放到滚动面板中
```

在通过构造方法创建部门树对话框时，需要传入要填写部门的文本框对象，这样在捕获树节点被选中的事件后会自动填写部门名称。用来捕获树节点被选中事件的关键代码如下：

例程 16　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\common\DeptTreeDialog.java

```
tree.addTreeSelectionListener(new TreeSelectionListener() {          //捕获树节点被选中的事件
    public void valueChanged(TreeSelectionEvent e) {
```

```
            TreePath treePath = e.getPath();                          //获得被选中树节点的路径
            DefaultMutableTreeNode node = (DefaultMutableTreeNode) treePath
                    .getLastPathComponent();                          //获得被选中树节点的对象
            if (node.getChildCount() == 0) {                          //被选中的节点为叶子节点
                textField.setText(node.toString());                   //将选中节点的名称显示到传入的文本框中
            } else {                                                  //被选中的节点不是叶子节点
                JOptionPane.showMessageDialog(null, "请选择所在的具体部门",
                        "错误提示", JOptionPane.ERROR_MESSAGE);
                return;
            }
            dispose();                                                //销毁部门树对话框
        }
    });
```

3.6.5　创建通过部门树选取员工的面板和对话框

在系统中有多处需要通过部门树选取员工，其中一处是在主窗体中，其他的均在对话框中，所以这里需要单独实现一个通过部门树选取员工的面板，然后将面板添加到主窗体或对话框中，从而实现代码的最大重用。最终实现的对话框效果如图 3.18 所示，当选中左侧部门树中的相应部门时，在右侧表格中将列出该部门及其子部门的所有员工。

图 3.18　“按部门查找员工”对话框

下面实现面板类 DeptAndPersonnelPanel，首先创建表格，在创建表格时，可以通过向量初始化表格，也可以通过数组初始化表格。关键代码如下：

例程 17　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\common\DeptAndPersonnelPanel.java

```
tableColumnV = new Vector<String>();                                  //创建表格列名向量
String tableColumns[] = new String[] { "序号", "档案编号", "姓名", "性别", "部门", "职务" };
for (int i = 0; i < tableColumns.length; i++) {                       //添加表格列名
    tableColumnV.add(tableColumns[i]);
}
tableValueV = new Vector<Vector<String>>();                           //创建表格值向量
showAllRecord();                                                      //默认显示所有档案
tableModel = new DefaultTableModel(tableValueV, tableColumnV);        //创建表格模型对象
table = new JTable(tableModel);                                       //创建表格对象
personnalScrollPane.setViewportView(table);                           //将表格添加到滚动面板中
```

然后为部门树添加节点选取事件处理代码，当选取的为根节点时，将显示所有档案，当选取的为子节点时，将显示该部门的档案，否则显示选中部门包含子部门的所有档案。关键代码如下：

例程 18 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\common\DeptAndPersonnelPanel.java

```
tree.addTreeSelectionListener(new TreeSelectionListener() {
    public void valueChanged(TreeSelectionEvent e) {
        TreePath path = e.getPath();                                    //获得被选中树节点的路径
        tableValueV.removeAllElements();                                //移除表格中的所有行
        if (path.getPathCount() == 1) {                                 //选中树的根节点
            showAllRecord();                                            //显示所有档案
        } else {                                                        //选中树的子节点
            String deptName = path.getLastPathComponent().toString();   //获得选中部门的名称
            TbDept selectDept = (TbDept) dao.queryDeptByName(deptName); //检索指定部门对象
            Iterator sonDeptIt = selectDept.getTbDepts().iterator();
            if (sonDeptIt.hasNext()) {                                  //选中树的二级节点
                while (sonDeptIt.hasNext()) {
                    showRecordInDept((TbDept) sonDeptIt.next());        //显示选中部门所有子部门的档案
                }
            } else {                                                    //选中树的叶子节点
                showRecordInDept(selectDept);                           //显示选中部门的档案
            }
        }
        tableModel.setDataVector(tableValueV, tableColumnV);
    }
});
```

下面实现对话框类 DeptAndPersonnelDialog，在对话框中提供 3 个按钮，用户可以通过“全选”按钮选择表格中的所有档案，也可以用鼠标点选指定档案，然后单击“添加”按钮，将选中的档案记录添加到指定向量中，添加结束后单击“退出”按钮。需要注意的是，在单击“退出”按钮时并没有销毁对话框，只是将其变为不可见，在调用对话框的位置获得选中档案信息后才销毁对话框。

负责捕获“添加”按钮事件的关键代码如下：

例程 19 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\common\DeptAndPersonnelDialog.java

```
final JButton addButton = new JButton();
addButton.addActionListener(new ActionListener() {                  //捕获按钮被按下的事件
    public void actionPerformed(ActionEvent e) {
        int[] rows = table.getSelectedRows();                       //获得选中行的索引
        int columnCount = table.getColumnCount();                   //获得表格的列数
        for (int row = 0; row < rows.length; row++) {
            Vector<String> recordV = new Vector<String>();          //创建一个向量对象，代表表格的一行
            for (int column = 0; column < columnCount; column++) {
                recordV.add(table.getValueAt(rows[row], column).toString());//将表格中的值添加到向量中
            }
            selectedRecordV.add(recordV);                           //将代表选中行的向量添加到另一个向量中
        }
    }
});
addButton.setText("添加");                                          //设置按钮的名称
```

3.7　人事管理模块设计

人事管理模块是企业人事管理系统的灵魂，是其他模块的基础，所以能否合理设计人事管理模块，对系统的整体设计和系统功能的开发将起到十分重要的作用。

3.7.1　人事管理模块功能概述

人事管理模块包含档案管理、考勤管理、奖惩管理和培训管理 4 个子模块。

档案管理模块用来建立和修改员工档案，当进入档案管理模块时，将看到如图 3.19 所示的界面。

图 3.19　员工档案列表界面

单击“新建员工档案”按钮或在表格中选中要修改的员工档案后单击“修改员工档案”按钮，将打开如图 3.20 所示的界面，在该界面中可以建立或修改员工档案，并且可以设置员工照片，填写完成后单击“保存”按钮保存员工档案。

考勤管理和奖惩管理用来填写相关记录，这些记录信息将体现在统计报表模块，例如在这里给某员工填写一次迟到考勤，在做统计报表时将根据其采用的账套在其待遇中扣除相应的金额。这两个模块的实现思路基本相同，在这里只给出考勤管理界面，如图 3.21 所示。

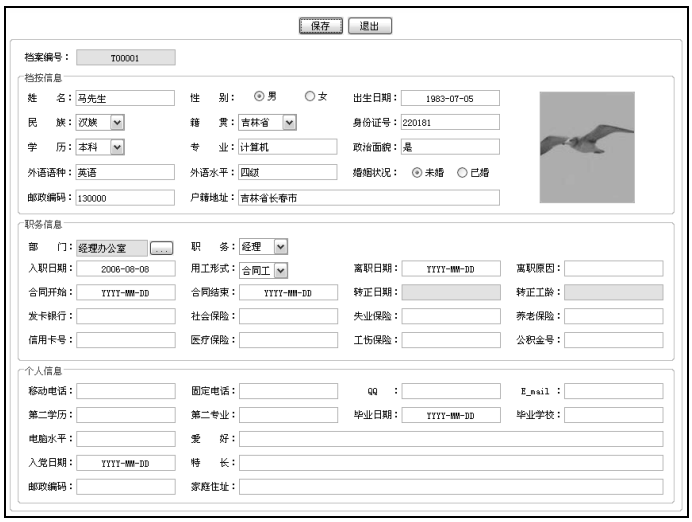


图 3.20　填写档案信息界面

图 3.21　考勤管理界面

培训管理用来记录对员工的培训信息，如图 3.22 所示，选中培训记录后单击“查看”按钮可以查看具体的培训人员。

图 3.22　培训列表界面

3.7.2 人事管理模块技术分析

在开发该模块时，需要处理大量用户输入的信息。处理用户输入信息的第一步是检查用户输入信息的合法性。如果是利用常规方法去验证每个组件接收到的数据，将耗费大量的时间和代码。对于这种情况，可以利用 Java 的反射机制先进行简单的验证，例如不允许为空的验证，然后再针对特殊的数据进行具体的验证，例如日期型数据。

在建立员工档案时需要支持上传员工照片的功能。如果想支持这一功能，必须了解两项关键技术，一是如何弹出用来选取照片的对话框，二是如何将照片文件上传到指定的位置。用来选取照片的对话框可以通过 javax.swing.JFileChooler 类实现，还可以通过实现 javax.swing.filechooser.FileFilter 接口，对指定路径中的文件进行过滤，令照片选取对话框中只显示照片文件。实现上传照片功能需要通过 java.io.File、java.io.FileInputStream 和 java.io.FileOutputStream 类联合实现。

在考勤管理和奖惩管理模块，既可以直接在员工下拉列表框中选取考勤或奖惩的员工，也可以先选取员工所在的部门，对员工下拉列表框中的可选项进行筛选，然后再选取具体的员工。要实现这一功能，需要实现组件之间的联动，即当选取部门时，将间接控制员工下拉列表框的变化；同样在选取员工下拉列表框时，也要间接控制部门组件的变化，即在部门组件中要显示员工所在的部门。可以通过捕获各个组件的事件完成这一功能，例如通过 java.awt.event.ItemListener 监听器捕获下拉列表框中被选中的事件，通过 javax.swing.event.TreeSelectionListener 监听器捕获树节点被选中的事件。

3.7.3 人事管理模块的实现过程

人事管理使用的主要数据表：tb_record、tb_duty_info、tb_personal_info、tb_timecard、tb_rewards_and_ punishment、tb_bring_up_content、tb_bring_up_ontent

在开发人事管理模块时，主要是突破技术分析中的几个技术点，掌握这几个技术点后，就可以顺利地实现人事管理模块了。

1．实现上传员工照片功能

在开发上传员工照片功能时，首先是确定显示照片的载体。在 Swing 中可以通过 JLable 组件显示照片，在该组件中也可以显示文字。在本系统中如果已上传照片则显示照片，否则显示提示文字。关键代码如下：

例程 20 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\personnel\RecordOperatePanel.java

```
photoLabel = new JLabel();                                          //创建用来显示照片的对象
photoLabel.setHorizontalAlignment(SwingConstants.CENTER);           //设置照片或文字居中显示
photoLabel.setBorder(new TitledBorder(null, "", TitledBorder.DEFAULT_JUSTIFICATION,
        TitledBorder.DEFAULT_POSITION, null, null));                //设置边框
photoLabel.setPreferredSize(new Dimension(120, 140));               //设置显示照片的大小
if (UPDATE_RECORD == null || UPDATE_RECORD.getPhoto() == null) {    //新建档案或未上传照片
    photoLabel.setText("双击添加照片");                              //显示文字提示
} else {                                                            //修改档案并且已上传照片
    URL url = this.getClass().getResource("/personnel_photo/");     //获得指定路径的绝对路径
    String photo = url.toString().substring(5) + UPDATE_RECORD.getPhoto();   //组织员工照片的存放路径
```

```
    photoLabel.setIcon(new ImageIcon(photo));                    //创建照片对象并显示
}
```

然后是确定如何弹出供用户选取照片的对话框。可以通过按钮捕获用户上传照片的请求，也可以通过 JLable 组件自己捕获该请求，即为 JLable 组件添加鼠标监听器，当用户双击该组件时，弹出供用户选取照片的对话框，本系统采用的是后者。Swing 提供了一个用来选取文件的对话框类 JFileChooser，当执行 JFileChooser 类的 showOpenDialog()方法时将弹出文件选取对话框，如图 3.23 所示。该方法返回 int 型值，用来区别用户执行的操作。当返回值为静态常量 APPROVE_OPTION 时，表示用户选取了照片，在这种情况下将选中的照片显示到 JLable 组件中。关键代码如下：

图 3.23　选取照片对话框

例程 21　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\personnel\RecordOperatePanel.java

```
photoLabel.addMouseListener(new MouseAdapter() {                 //添加鼠标监听器
    public void mouseClicked(MouseEvent e) {
        if (e.getClickCount() == 2) {                            //判断是否为双击
            JFileChooser fileChooser = new JFileChooser();       //创建文件选取对话框
            fileChooser.setFileFilter(new FileFilter() {         //为对话框添加文件过滤器
                    public String getDescription() {             //设置提示信息
                        return "图像文件（.jpg;.gif）";
                    }
                    public boolean accept(File file) {           //设置接受文件类型
                        if (file.isDirectory())
                            return true;                         //为文件夹则返回 true
                        String fileName = file.getName().toLowerCase();
                        if (fileName.endsWith(".jpg") || fileName.endsWith(".gif"))
                            return true;                         //为 JPG 或 JIF 格式文件则返回 true
                        return false;                        //否则返回 false，即不显示在文件选取对话框中
                    }
                });
            int i = fileChooser.showOpenDialog(getParent()); //弹出文件选取对话框并接收用户的处理信息
            if (i == fileChooser.APPROVE_OPTION) {               //用户选取了照片
                File file = fileChooser.getSelectedFile();       //获得用户选取的文件对象
                if (file != null) {
                    ImageIcon icon = new ImageIcon(file
                            .getAbsolutePath());                 //创建照片对象
                    photoLabel.setText(null);                    //取消提示文字
                    photoLabel.setIcon(icon);                    //显示照片
                }
            }
        }
    }
});
```

最后就是在保存档案信息时将照片上传到指定路径下，上传到指定路径下的照片名称将修改为档案编号，但是因为可以上传两种格式的图片，为了记录图片格式，还是要将照片名称保存到数据库中。关键代码如下：

例程 22 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\personnel\RecordOperatePanel.java

```
if (photoLabel.getIcon() != null) {                                    //查看是否上传照片
    File selectPhoto = new File(photoLabel.getIcon().toString());      //通过选中图片的路径创建文件对象
    URL url = this.getClass().getResource("/personnel_photo/");        //获得指定路径的绝对路径
    StringBuffer uriBuffer = new StringBuffer(url.toString());         //组织文件路径
    String selectPhotoName = selectPhoto.getName();
    int i = selectPhotoName.lastIndexOf(".");
    uriBuffer.append(recordNoTextField.getText());
    uriBuffer.append(selectPhotoName.substring(i));
    try {
        File photo = new File(new URL(uriBuffer.toString()).toURI());  //创建上传文件对象
        record.setPhoto(photo.getName());                              //将图片名称保存到数据库
        if (!photo.exists()) {                                         //如果文件不存在则创建文件
            photo.createNewFile();
        }
        InputStream inStream = new FileInputStream(selectPhoto);       //创建输入流对象
        OutputStream outStream = new FileOutputStream(photo);          //创建输出流对象
        int readBytes = 0;                                             //读取字节数
        byte[] buffer = new byte[10240];                               //定义缓存数组
        while ((readBytes = inStream.read(buffer, 0, 10240)) != -1) {  //从输入流读取数据到缓存数组中
            outStream.write(buffer, 0, readBytes);                     //将缓存数组中的数据输出到输出流
        }
        outStream.close();                                             //关闭输出流对象
        inStream.close();                                              //关闭输入流对象
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

2．实现组件联动功能

在开发考勤功能时，实现了部门和员工组件之间的联动功能，如果用户直接单击“考勤员工”下拉列表框，在下拉列表框中将显示所有员工，如图 3.24 所示。这是因为在初始化“考勤员工”下拉列表框时，添加的是所有员工。关键代码如下：

图 3.24 直接单击“考勤员工”下拉列表

例程 23 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\personnel\TimecardPanel.java

```
personnalComboBox = new JComboBox();                    //创建下拉列表框对象
personnalComboBox.addItem("请选择");                    //添加提示项
Iterator recordIt = dao.queryRecord().iterator();       //检索所有员工
while (recordIt.hasNext()) {                            //通过循环添加到下拉列表框中
    TbRecord record = (TbRecord) recordIt.next();
```

```
        personnalComboBox.addItem(record.getRecordNumber() + "      " + record.getName());
    }
```

当用户选中考勤员工后，在“所在部门”文本框将填入选中员工所在的部门，实现这一功能是通过捕获下拉列表框选项状态发生改变的事件。关键代码如下：

例程 24　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\personnel\TimecardPanel.java

```
personnalComboBox.addItemListener(new ItemListener() {      //捕获下拉列表框选项状态发生改变的事件
❶       public void itemStateChanged(ItemEvent e) {
❷           if (e.getStateChange() == ItemEvent.SELECTED) {//查看是否是由选中当前项触发的
❸               String selectedItem = (String) e.getItem();      //获得选中项的内容
                if (selectedItem.equals("请选择")) {             //当选中项为“请选择”时，设置部门文本框为空
                    inDeptTextField.setText(null);
                } else {                                         //否则设置部门文本框为选中员工所在的部门
                    TbRecord record = (TbRecord) dao.queryRecordByNum(selectedItem.substring(0, 6));
                    inDeptTextField.setText(record.getTbDutyInfo().getTbDept().getName());
                }
            }
        }
});
```

代码贴士

❶ itemStateChanged()：当下拉列表框的选中项发生改变时将触发该方法。

❷ getStateChange()：该方法返回一个 int 型值。当返回值等于静态常量 ItemEvent.DESELECTED 时，表示此次事件是由取消原选中项触发的；当返回值等于静态常量 ItemEvent.SELECTED 时，表示此次事件是由选中当前项触发的。

❸ getItem()：通过该方法可以获得触发此次事件的选项的内容。

如果用户先选中考勤员工所在的部门，例如选中“经理办公室”，再单击“考勤员工”下拉列表框，在下拉列表框中将显示选中部门的所有员工，如图 3.25 所示。这是因为在捕获按钮事件弹出部门选取对话框时，根据用户选取的部门对“考勤员工”下拉列表框进行了处理。关键代码如下：

图 3.25　选中部门后再单击“考勤员工”下拉列表框

例程 25　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\personnel\TimecardPanel.java

```
final JButton inDeptTreeButton = new JButton();                 //创建按钮对象
inDeptTreeButton.addActionListener(new ActionListener() {       //捕获按钮事件
    public void actionPerformed(ActionEvent e) {
        DeptTreeDialog deptTree = new DeptTreeDialog(inDeptTextField);          //创建部门选取对话框
        deptTree.setBounds(375, 317, 101, 175);                 //设置部门选取对话框的显示位置
        deptTree.setVisible(true);                              //弹出部门选取对话框
        TbDept dept = (TbDept) dao.queryDeptByName(inDeptTextField.getText());  //检索选中的部门对象
        personnalComboBox.removeAllItems();                     //清空“考勤员工”下拉列表框中的所有选项
        personnalComboBox.addItem("请选择");                    //添加提示项
        //通过 Hibernate 的一对多关联获得与该部门关联的职务信息对象
        Iterator dutyInfoIt = dept.getTbDutyInfos().iterator();
        while (dutyInfoIt.hasNext()) {                                  //遍历职务信息对象
```

```
                TbDutyInfo dutyInfo = (TbDutyInfo) dutyInfoIt.next();                    //获得职务信息对象
                //通过 Hibernate 的一对一关联获得与职务信息对象关联的档案信息对象
                TbRecord tbRecord = dutyInfo.getTbRecord();
                personnalComboBox.addItem(tbRecord.getRecordNumber() + "      " + tbRecord.getName());
            }
        }
});
inDeptTreeButton.setText("...");
```

3．通过 Java 反射验证数据是否为空

在添加培训记录时，所有的培训信息均不允许为空，并且都是通过文本框接收用户输入信息的，在这种情况下可以通过 Java 反射验证数据是否为空，当为空时弹出提示信息，并令为空的文本框获得焦点。关键代码如下：

例程 26 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\personnel\ BringUpOperatePanel.java

```
❶   Field[] fields = BringUpOperatePanel.class.getDeclaredFields(); //通过 Java 反射机制获得类中的所有属性
    for (int i = 0; i < fields.length; i++) {                        //遍历属性数组
        Field field = fields[i];                                     //获得属性
❷       if (field.getType().equals(JTextField.class)) {              //只验证 JtextField 类型的属性
            field.setAccessible(true);          //默认情况下不允许访问私有属性，如果设为 true 则允许访问
            JTextField textField = null;
            try {
❸               textField = (JTextField) field.get(BringUpOperatePanel.this);  //获得本类中的对应属性
            } catch (Exception e) {
                e.printStackTrace();
            }
            if (textField.getText().equals("")) {                    //查看该属性是否为空
                String infos[] = { "请将培训信息填写完整！", "所有信息均不允许为空！" };
                JOptionPane.showMessageDialog(null, infos, "友情提示", JOptionPane.INFORMATION_
MESSAGE);
                textField.requestFocus();                            //令为空的文本框获得焦点
                return;
            }
        }
    }
```

代码贴士

❶ getDeclaredFields()：该方法返回一个 Field 型数组，在数组中包含了调用类的所有属性，包括公共、保护、默认（包）访问和私有字段，但不包括继承的字段。

❷ getType()：该方法返回一个 Class 对象，它标识了此属性的声明类型。

❸ BringUpOperatePanel.this：代表本类。

3.7.4 单元测试

在测试添加培训记录时，在不填写任何信息的情况下直接单击“保存”按钮，并没有弹出如图 3.26 所示的提示培训信息不允许为空的对话框。

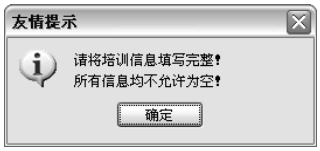


图 3.26　提示培训信息不允许为空的对话框

查看控制台，发现抛出了如图 3.27 所示的异常信息，异常信息中提示不能访问 BringUpOperatePanel 类中 private 类型的属性，并且提示是由 BringUpOperatePanel 类的第 170 行抛出的，第 170 行的相关代码如图 3.28 所示。

```
java.lang.IllegalAccessException: Class com.mwq.frame.personnel.BringUpOperatePanel$4
        can not access a member of class com.mwq.frame.personnel.BringUpOperatePanel with modifiers "private"
        at sun.reflect.Reflection.ensureMemberAccess(Unknown Source)
        at java.lang.reflect.Field.doSecurityCheck(Unknown Source)
        at java.lang.reflect.Field.getFieldAccessor(Unknown Source)
        at java.lang.reflect.Field.get(Unknown Source)
        at com.mwq.frame.personnel.BringUpOperatePanel$4.actionPerformed(BringUpOperatePanel.java:170)
```

图 3.27　在控制台抛出的异常信息

```
167         JTextField textField = null;
168         try {
169             textField = (JTextField) field
170                     .get(BringUpOperatePanel.this);// 获得本类中的对应属性
171         } catch (Exception e) {
172             e.printStackTrace();
173         }
```

图 3.28　提示发生错误的代码

对照提示信息分析代码，发现这是因为所有 JTextField 类型的属性均声明为私有（private），而 Java 反射机制默认不允许访问私有属性。在这种情况下可以通过 java.lang.reflect.Field 类的 setAccessible (boolean accessible)方法设置私有属性可以访问，即设置为 true，默认为 false。关键代码如下：

例程 27　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\personnel\BringUpOperatePanel.java

```
field.setAccessible(true);                    //默认情况下不允许访问私有属性，如果设置为 true 则允许访问
JTextField textField = null;
try {
        textField = (JTextField) field.get(BringUpOperatePanel.this);          //获得本类中的对应属性
} catch (Exception e) {
    e.printStackTrace();
}
```

3.8　待遇管理模块设计

待遇管理功能是企业人事管理系统的主要功能之一，该功能需要建立在人事管理功能的基础上，例如人事管理中的考勤管理和奖惩管理在待遇管理中将用到。

3.8.1　待遇管理模块功能概述

待遇管理模块包含账套管理、人员设置和统计报表 3 个子模块。

账套管理模块用来建立和维护账套信息，包括建立、修改和删除账套，以及为账套添加项目和修

改金额，或者从账套中删除项目。首先需要建立一个账套，“新建账套”对话框如图 3.29 所示，其中账套说明用来详细介绍该账套的适用范围。然后为新建的账套添加项目，“添加项目”对话框如图 3.30 所示，选中要添加的项目后单击“添加”按钮。最后修改新添加项目的金额，“修改金额”对话框如图 3.31 所示，输入项目金额后单击“确定”按钮。

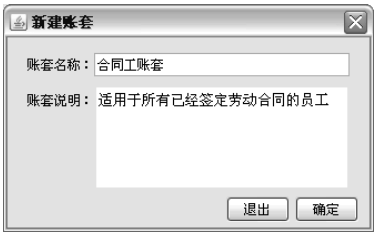


图 3.29 “新建账套”对话框

图 3.30 “添加项目”对话框

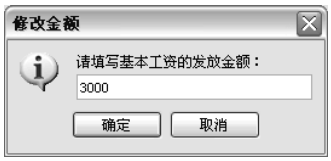


图 3.31 “修改金额”对话框

人员设置模块用来设置每个账套具体适合的人员，在这里将用到 3.6.5 节实现的通过部门树选取员工的对话框。

统计报表模块用来生成员工待遇统计报表，可以生成月、季、半年和年报表，如图 3.32 所示。当生成月报表时，可以选择统计的年和月份；当生成季报表时，可以选择统计的年和季度；当生成半年报表时，可以选择统计的年以及上半年或下半年；当生成年报表时，则只可以选择统计的年。

图 3.32 生成统计报表的种类

3.8.2 待遇管理模块技术分析

在实现修改账套项目金额的功能时，通常情况下是通过 JDialog 对话框实现，但是因为只需要接收一条修改金额的信息，所以在这里也可以通过 JOptionPane 提示框实现，这样在实现功能的前提下可以少创建一个类，提高了代码的可读性。

在开发应用程序时，充分使用提示对话框也是一个不错的选择，既可以帮助用户使用系统，又可以保证系统的安全运行。

3.8.3 待遇管理模块的实现过程

待遇管理使用的主要数据表：tb_reckoning、tb_reckoning_info、tb_reckoning_list

在开发账套管理模块时，首先需要建立一个账套，通过弹出对话框获得账套名称和账套说明，将新建的账套添加到左侧的账套表格中，并设置为选中行，还要同步刷新右侧的账套项目表格。关键代码如下：

例程 28 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\treatement\CriterionSetPanel.java

```
final JButton addSetButton = new JButton();
addSetButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (needSaveRow == -1) {                    //没有需要保存的账套
```

```
                CreateCriterionSetDialog createCriterionSet = new CreateCriterionSetDialog();
                createCriterionSet.setBounds((width - 350) / 2, (height - 250) / 2, 350, 250);
                createCriterionSet.setVisible(true);          //弹出“新建账套”对话框，接收账套名称和账套说明
                if (createCriterionSet.isSubmit()) {          //单击“确定”按钮
                    String name = createCriterionSet.getNameTextField().getText();          //获得账套名称
                    String explain = createCriterionSet.getExplainTextArea().getText();     //获得账套说明
                    needSaveRow = leftTableValueV.size();     //将新建账套设置为需要保存的账套
                    Vector<String> newCriterionSetV = new Vector<String>();//创建代表账套表格行的向量对象
                    newCriterionSetV.add(needSaveRow + 1 + ""); //添加账套序号
                    newCriterionSetV.add(name);               //添加账套名称
                    leftTableModel.addRow(newCriterionSetV);  //将向量对象添加到左侧的账套表格中
                    leftTable.setRowSelectionInterval(needSaveRow, needSaveRow);//设置新建账套为选中行
                    textArea.setText(explain);                //设置账套说明
                    TbReckoning reckoning = new TbReckoning(); //创建账套对象
                    reckoning.setName(name);                  //设置账套名称
                    reckoning.setExplain(explain);            //设置账套说明
                    reckoningV.add(reckoning);                //将账套对象添加到向量中
                    refreshItemAllRowValueV(needSaveRow);     //同步刷新右侧的账套项目表格
                }
            } else {                                          //有需要保存的账套，弹出提示保存对话框
                JOptionPane.showMessageDialog(null, "请先保存账套：" + leftTable.getValueAt(needSaveRow, 1),
                        "友情提示", JOptionPane.INFORMATION_MESSAGE);
            }
        }
    });
    addSetButton.setText("新建账套");
```

然后为新建的账套添加项目，通过弹出对话框获得用户添加的项目，因为需要通过弹出对话框中的表格对象获得选中项目的信息，所以在完成添加项目之前不能销毁添加项目对话框对象，而是将其设置为不可见的，添加完成后才能销毁，并且需要判断新添加项目在账套中是否已经存在。关键代码如下：

例程 29　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\treatement\CriterionSetPanel.java

```
public void addItem(int leftSelectedRow) {
    AddAccountItemDialog addAccountItemDialog = new AddAccountItemDialog();
    addAccountItemDialog.setBounds((width - 500) / 2, (height - 375) / 2, 500, 375);
    addAccountItemDialog.setVisible(true);                    //弹出“添加项目”对话框
    JTable itemTable = addAccountItemDialog.getTable();       //获得对话框中的表格对象
    int[] selectedRows = itemTable.getSelectedRows();         //获得选中行的索引
    if (selectedRows.length > 0) {                            //有新添加的项目
        needSaveRow = leftSelectedRow;                        //设置当前账套为需要保存的账套
        int defaultSelectedRow = rightTable.getRowCount(); //将选中行设置为新添加项目的第一行
        TbReckoning reckoning = reckoningV.get(leftSelectedRow);          //获得选中账套的对象
        for (int i = 0; i < selectedRows.length; i++) {                   //通过循环向账套中添加项目
            String name = itemTable.getValueAt(selectedRows[i], 1).toString();  //获得项目名称
            String unit = itemTable.getValueAt(selectedRows[i], 2).toString();  //获得项目单位
            Iterator<TbReckoningInfo> reckoningInfoIt = reckoning
                    .getTbReckoningInfos().iterator();    //遍历账套中的现有项目
```

```
                boolean had = false;                                    //默认在现有项目中不包含新添加的项目
                while (reckoningInfoIt.hasNext()) {                     //通过循环查找是否存在
                    TbAccountItem accountItem = reckoningInfoIt.next().getTbAccountItem();
                                                                        //获得已有的项目对象
                    if (accountItem.getName().equals(name) && accountItem.getUnit().equals(unit)) {
                        had = true;                                     //存在
                        break;                                          //跳出循环
                    }
                }
                if (!had) {                                             //如果没有则添加
                    TbReckoningInfo reckoningInfo = new TbReckoningInfo();//创建账套信息对象
                    TbAccountItem accountItem = (TbAccountItem) dao
                            .queryAccountItemByNameUnit(name, unit);    //获得账套项目对象
                    accountItem.getTbReckoningInfos().add(reckoningInfo);//建立从账套项目到账套信息的关联
                    reckoningInfo.setTbAccountItem(accountItem);        //建立从账套信息到账套项目的关联
                    reckoningInfo.setMoney(0);                          //设置项目金额为 0
                    reckoningInfo.setTbReckoning(reckoning);            //建立从账套信息到账套的关联
                    reckoning.getTbReckoningInfos().add(reckoningInfo); //建立从账套到账套信息的关联
                }
            }
            refreshItemAllRowValueV(leftSelectedRow);                   //同步刷新右侧的账套项目表格
            rightTable.setRowSelectionInterval(defaultSelectedRow, defaultSelectedRow); //设置选中行
            addAccountItemDialog.dispose();                             //销毁“添加项目”对话框
        }
    }
```

下面为添加的项目修改金额，否则是不允许保存的，因为默认项目金额为 0，这是没有意义的。这里是通过 JOptionPane 提示框获得修改后的金额，之后还要判断用户输入的金额是否符合要求，首要条件是数字，这里还要求必须为 1～999999 之间的整数。关键代码如下：

例程 30 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\treatement\CriterionSetPanel.java

```
public void updateItemMoney(int leftSelectedRow, int rightSelectedRow) {
    String money = null;
    done: while (true) {
❶       money = JOptionPane.showInputDialog(null, "请填写"
                + rightTable.getValueAt(rightSelectedRow, 1) + "的"
                + rightTable.getValueAt(rightSelectedRow, 3) + "金额：",
                "修改金额", JOptionPane.INFORMATION_MESSAGE);
        if (money == null) {                                            //用户单击“取消”按钮
            break done;                                                 //取消修改
        } else {                                                        //用户单击“确定”按钮
            if (money.equals("")) {                                     //未输入金额，弹出提示对话框
❷               JOptionPane.showMessageDialog(null, "请输入金额！", "友情提示",
                        JOptionPane.INFORMATION_MESSAGE);
            } else {                                                    //输入了金额
                Pattern pattern = Pattern.compile("[1-9][0-9]{0,5}");   //金额必须在 1～999999 之间
                Matcher matcher = pattern.matcher(money);               //通过正则表达式判断是否符合要求
                if (matcher.matches()) {                                //符合要求
                    needSaveRow = leftSelectedRow;                      //设置当前账套为需要保存的账套
```

```
                        rightTable.setValueAt(money, rightSelectedRow, 4);          //修改项目金额
                        int nextSelectedRow = rightSelectedRow + 1;                 //默认存在下一行
                        if (nextSelectedRow < rightTable.getRowCount()) {           //存在下一行
                            rightTable.setRowSelectionInterval(nextSelectedRow, nextSelectedRow);
                        }
                        String name = rightTable.getValueAt(rightSelectedRow, 1).toString();//获得项目名称
                        String unit = rightTable.getValueAt(rightSelectedRow, 2).toString();  //获得项目单位
                        TbReckoning reckoning = reckoningV.get(leftSelectedRow);    //获得选中账套的对象
                        Iterator reckoningInfoIt = reckoning.getTbReckoningInfos().iterator();//遍历项目
                        while (reckoningInfoIt.hasNext()) {                         //通过循环查找选中项目
                            TbReckoningInfo reckoningInfo = (TbReckoningInfo) reckoningInfoIt.next();
                            TbAccountItem accountItem = reckoningInfo.getTbAccountItem();
                            if (accountItem.getName().equals(name) && accountItem.getUnit().equals(unit)) {
                                reckoningInfo.setMoney(new Integer(money));         //修改金额
                                break;                                              //跳出循环
                            }
                        }
                        break done;                                                 //修改完成
                    } else {                                                        //不符合要求，弹出提示对话框
❸                       String infos[] = { "金额输入错误，请重新输入！", "金额必须为 0~999999 之间的整数！" };
                        JOptionPane.showMessageDialog(null, infos, "友情提示",
                                JOptionPane.INFORMATION_MESSAGE);
                    }
                }
            }
        }
    }
```

代码贴士

❶ showInputDialog()：该静态方法用来弹出提示某些输入的提示框，可以接受用户的输入信息，也可以提供几个备选项供用户选择。

❷ showMessageDialog()：该静态方法用来弹出提示某些消息的提示框，消息的类型可以为错误（ERROR_MESSAGE）、消息（INFORMATION_MESSAGE）、警告（WARNING_MESSAGE）、问题（QUESTION_MESSAGE）或普通（PLAIN_MESSAGE）。

❸ infos[]：如果需要将消息分多行显示，可以将消息放到数组中，每一个数组索引代表一行。

最后开发统计报表，首先判断报表类型，然后根据报表类型组织报表的起止时间。下面是生成季度报表的代码。

例程 31　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\treatement\ReportFormsPanel.java

```
String quarter = quarterComboBox.getSelectedItem().toString();      //获得报表季度
if (quarter.equals("第一")) {
    reportForms(year + "-1-1", year + "-3-31");                     //生成报表
} else if (quarter.equals("第二")) {
    reportForms(year + "-4-1", year + "-6-30");                     //生成报表
} else if (quarter.equals("第三")) {
    reportForms(year + "-7-1", year + "-9-30");                     //生成报表
```

```
} else {                                                    //第四季度
    reportForms(year + "-10-1", year + "-12-31");           //生成报表
}
```

下面的代码负责在生成报表时向表格中添加员工的关键信息，初始实发金额为 0 元。

例程 32 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\treatement\ReportFormsPanel.java

```
TbRecord record = (TbRecord) recordIt.next();               //获得档案对象
Vector recordV = new Vector();                              //创建与档案对象对应的向量
recordV.add(num++);                                         //添加序号
recordV.add(record.getRecordNumber());                      //添加档案编号
recordV.add(record.getName());                              //添加姓名
recordV.add(record.getSex());                               //添加性别
TbDutyInfo dutyInfo = record.getTbDutyInfo();
recordV.add(dutyInfo.getTbDept().getName());                //添加部门
recordV.add(dutyInfo.getTbDuty().getName());                //添加职务
int salary = 0;                                             //初始实发金额为 0
```

下面的代码负责在生成报表时计算员工的奖惩金额，通过 Hibernate 的关联得到的是员工的所有奖惩，需要通过集合过滤功能进行过滤，检索符合条件的奖惩信息。关键代码如下：

例程 33 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\treatement\ReportFormsPanel.java

```
Set rewAndPuns = record.getTbRewardsAndPunishmentsForRecordId();
String types[] = new String[] { "奖励", "惩罚" };
for (int i = 0; i < types.length; i++) {
    String filterHql = "where this.type='" + types[i] + "' and ( ( startDate between '" + reportStartDateStr
            + "' and '" + reprotEndDateStr + "' or endDate between '" + reportStartDateStr
            + "' and '" + reprotEndDateStr + "' ) or ( '" + reportStartDateStr
            + "' between startDate and endDate and '" + reprotEndDateStr
            + "' between startDate and endDate ) )";        //组织用来过滤集合的 HQL 语句
    List list = dao.filterSet(rewAndPuns, filterHql);       //过滤奖惩记录
    if (list.size() > 0) {                                  //存在奖惩
        column += 1;                                        //列索引加 1
        int money = 0;                                      //初始奖惩金额为 0
        for (Iterator it = list.iterator(); it.hasNext();) {
            TbRewardsAndPunishment rewAndPun = (TbRewardsAndPunishment) it.next();
            money += rewAndPun.getMoney();                  //累加奖惩金额
        }
        recordV.add(money);                                 //添加奖惩金额
        if (i == 0)                                         //奖励
            salary += money;                                //计算实发金额
        else                                                //惩罚
            salary -= money;                                //计算实发金额
    } else {
        recordV.add("—");                                   //没有奖励或惩罚
    }
}
```

3.9　系统维护模块设计

系统维护模块用来维护系统的基本信息，例如企业架构信息和常用的职务种类、用工形式等信息，另外该模块还提供了对系统进行初始化的功能。

3.9.1　系统维护模块功能概述

系统维护模块包含企业架构、基本资料和初始化系统 3 个子模块。

企业架构模块用来维护企业的组织结构信息，包括修改公司及部门的名称，添加或删除部门。企业架构主界面效果如图 3.33 所示。

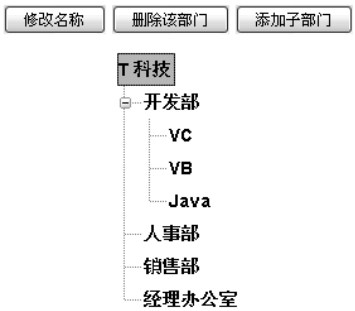


图 3.33　企业架构主界面

可以选中公司或部门后单击“修改名称”按钮，修改公司或部门的名称。如果修改的是公司名称，将弹出如图 3.34 所示的界面；如果修改的是部门名称，将弹出如图 3.35 所示的界面。

也可以为公司或部门添加子部门。如果是在公司下添加子部门，则选中公司节点；如果是在公司所属部门下添加子部门，则选中所属部门，然后单击“添加子部门”按钮，将弹出如图 3.36 所示的界面，在二级部门（例如图 3.33 中的 Java）下不能再包含子部门，如果试图在该级部门下建立子部门，将弹出如图 3.37 所示的提示框。

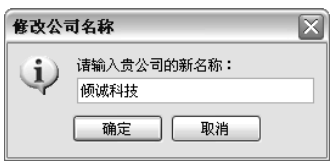


图 3.34　修改公司名称

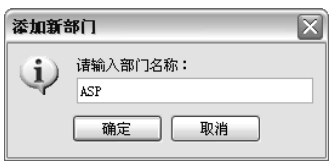


图 3.35　修改部门名称

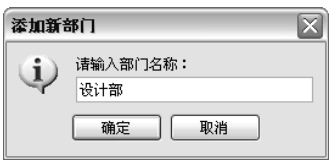


图 3.36　为公司或一级部门添加子部门

还可以选中部门节点后单击“删除该部门”按钮删除选中的部门，在删除之前将弹出如图 3.38 所示的提示框。需要注意的是，公司节点不允许删除，如果试图删除公司，将弹出如图 3.39 所示的提示框。

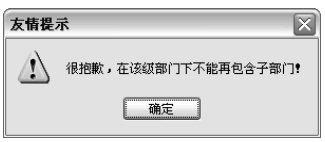


图 3.37　不允许在二级部门下添加子部门

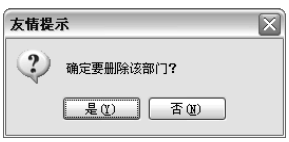


图 3.38　询问是否删除部门

图 3.39　提示公司不允许删除

基本资料模块用来维护系统中的基本信息，例如职务种类、账套项目等，如图 3.40 所示。

初始化系统模块用来对系统进行初始化，用户在使用系统之前，或者是想清空系统中的数据，可以通过该功能实现。不过在对系统进行初始化之前，一定要弹出一个如图 3.41 所示的提示框询问是否初始化，这样会增加系统的安全性，因为可能是用户不小心点到的。初始化完成后，将弹出如图 3.42 所示的提示框，该提示框的内容为本系统的使用步骤。

图 3.40　维护基本资料界面

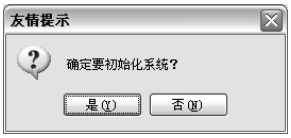


图 3.41　询问是否初始化系统

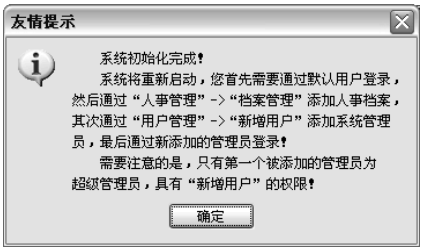


图 3.42　初始化完成后弹出的提示框

3.9.2　系统维护模块技术分析

维护企业架构是该模块的技术难点。因为企业架构是一个树状结构，所以需要通过 Swing 中的 JTree 组件完成。对企业架构的维护主要包括修改公司或部门的名称、设立新部门或取消现有部门，这对于 JTree 组件，就是修改节点名称、添加新节点或删除现有节点。为了实现上述对树节点的操作，还需要一些其他的操作 JTree 组件的知识，例如如何获得选中树节点的路径、如何获得选中树节点的对象、如何展开指定的树节点等。

3.9.3　系统维护模块的实现过程

系统维护使用的主要数据表：tb_dept

维护企业架构是该模块的技术难点，所以在这里只介绍企业架构模块的实现过程。

（1）实现修改名称的功能，分为修改公司名称和修改部门名称。在修改公司名称之前弹出提示询问是否真的修改，在修改部门名称时则不询问，直接弹出消息框供用户输入新名称。修改时需要分两步实现，第一步是修改系统界面的企业架构树，第二步是修改持久化对象，并持久化到数据库中。关键代码如下：

例程 34　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\system\FrameworkPanel.java

```
final JButton updateButton = new JButton();
updateButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
❶       TreePath selectionPath = tree.getSelectionPath();
        TbDept selected = null;
        String newName = null;
❷       if (selectionPath.getPathCount() == 1) {              //修改公司名称
            int i = JOptionPane.showConfirmDialog(null, "确定要修改贵公司的名称？",
                    "友情提示", JOptionPane.YES_NO_OPTION);   //弹出提示框
            if (i == 0) {                                     //修改（单击“是”按钮）
                String infos[] = { "请输入贵公司的新名称：", "修改公司名称", "请输入贵公司的新名称！" };
                newName = getName(infos);                     //获得修改后的名称
                if (newName != null)
                    selected = company;                       //修改的为公司名称
            }
```

```
        } else {//修改部门名称
            String infos[] = { "请输入部门的新名称：", "修改部门名称", "请输入部门的新名称！" };
            newName = getName(infos);                    //获得修改后的名称
            if (newName != null) {
                selected = company;                      //选中部门的所属部门
❸               Object[] paths = selectionPath.getPath(); //选中部门节点的路径对象
                for (int i = 1; i < paths.length; i++) {  //遍历选中节点路径
                    Iterator deptIt = selected.getTbDepts().iterator();
                    finded: while (deptIt.hasNext()) {    //通过循环查找选中节点路径对应的部门
                        TbDept dept = (TbDept) deptIt.next();
                        if (dept.getName().equals(paths[i].toString())) {
                            selected = dept;             //找到选中节点路径对应的部门
                            break finded;                //跳出到指定位置
                        }
                    }
                }
            }
        }
        if (selected != null) {
❹           DefaultMutableTreeNode treeNode = (DefaultMutableTreeNode) selectionPath
                    .getLastPathComponent();             //获得选中节点对象
❺           treeNode.setUserObject(newName);             //修改节点名称
❻           treeModel.reload();                          //刷新树结构
❼           tree.setSelectionPath(selectionPath);        //设置节点为选中状态
            selected.setName(newName);                   //修改部门对象
            dao.updateObject(company);                   //将修改持久化到数据库
            HibernateSessionFactory.closeSession();      //关闭数据库连接
        }
    }
});
updateButton.setText("修改名称");
```

代码贴士

❶ getSelectionPath()：通过该方法可以获得选中节点的路径对象，通过该对象可以获得选中节点的相关信息，例如选中节点对象、选中节点级别等。

❷ getPathCount()：通过该方法可以获得选中节点的级别，当返回值为 1 时，代表选中的为树的根节点；为 2 时，则代表选中的是树的直属子节点。

❸ getPath()：通过该方法可以获得选中节点路径包含的节点对象，返回值为 Object 型的数组。例如 A 为树的根节点，a1 和 a2 为 A 的子节点，如果选中 a2，则返回的为 Object nodes[]={A, a2}。

❹ getLastPathComponent()：通过该方法可以获得选中节点的对象。例如 A 为树的根节点，a1 和 a2 为 A 的子节点，如果选中 a2，则返回的为 a2。

❺ setUserObject(Object title)：该方法用来设置节点的标题。

❻ reload()：该方法用来刷新已经被修改树模型对象。

❼ setSelectionPath(TreePath path)：该方法用来设置选中指定路径的节点。

（2）实现添加部门的功能。只允许在公司及其直属部门下添加部门，在获得用户输入的要添加部门的名称后，需要判断在其所属部门中是否存在该部门，如果存在则弹出提示，否则添加到系统界面的企业架构树中，并持久化到数据库。关键代码如下：

例程 35 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\system\FrameworkPanel.java

```
final JButton addButton = new JButton();
addButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        TreePath selectionPath = tree.getSelectionPath();
        int pathCount = selectionPath.getPathCount();          //获得选中节点的级别
        had: if (pathCount == 3) {                             //选中的为 3 级节点
            JOptionPane.showMessageDialog(null, "很抱歉，在该级部门下不能再包含子部门！",
                    "友情提示", JOptionPane.WARNING_MESSAGE);
        } else {                                               //选中的为 1 级或 2 级节点
            String infos[] = { "请输入部门名称：", "添加新部门", "请输入部门名称！" };
            String newName = getName(infos);                   //获得新部门的名称
            if (newName != null) {                             //创建新部门
                DefaultMutableTreeNode parentNode = (DefaultMutableTreeNode) selectionPath
                        .getLastPathComponent();               //获得选中部门节点对象
❶               int childCount = parentNode.getChildCount();//获得该部门包含子部门的个数
                for (int i = 0; i < childCount; i++) {         //查看新创建的部门是否已经存在
❷                   TreeNode childNode = parentNode.getChildAt(i);
                    if (childNode.toString().equals(newName)) {
                        JOptionPane.showMessageDialog(null, "该部门已经存在！",
                                "友情提示", JOptionPane.WARNING_MESSAGE);
                        break had;                             //已经存在，跳出到指定位置
                    }
                }
                DefaultMutableTreeNode childNode = new DefaultMutableTreeNode(newName);
                                                               //创建部门节点
❸               treeModel.insertNodeInto(childNode, parentNode, childCount);
                                                               //插入新部门到选中部门的最后
❹               tree.expandPath(selectionPath);                //展开指定路径中的尾节点
                TbDept selected = company;                     //默认选中的为 1 级节点
                if (pathCount == 2) {                          //选中的为 2 级节点
                    String selectedName = selectionPath.getPath()[1].toString();  //获得选中节点的名称
                    Iterator deptIt = company.getTbDepts().iterator();//创建公司直属部门的迭代器对象
                    finded: while (deptIt.hasNext()) {         //遍历公司的直属部门
                        TbDept dept = (TbDept) deptIt.next();
                        if (dept.getName().equals(selectedName)) {  //查找与选中节点对应的部门
                            selected = dept;                   //设置为选中部门
                            break finded;                      //跳出循环
                        }
                    }
                }
                TbDept sonDept = new TbDept();                 //创建新部门对象
                sonDept.setName(newName);                      //设置部门名称
                sonDept.setTbDept(selected);                   //建立从新部门到所属部门的关联
                selected.getTbDepts().add(sonDept);            //建立从所属部门到新部门的关联
                dao.updateObject(company);                     //将新部门持久化到数据库
                HibernateSessionFactory.closeSession(); //关闭数据库连接
            }
        }
    }
```

```
});
addButton.setText("添加子部门");
```

代码贴士

❶ getChildCount()：通过该方法可以获得包含子节点的个数。

❷ getChildAt(int childIndex)：该方法用来获得指定索引位置的子节点，索引从 0 开始。

❸ insertNodeInto(MutableTreeNode newChild, MutableTreeNode parent, int index)：该方法用来将新创建的子节点 newChild 插入到父节点 parent 中索引为 index 的位置。

❹ expandPath(TreePath path): 该方法用来展开指定路径中的尾节点，前提条件是指定路径的尾节点不能是叶子节点。

（3）实现删除现有部门的功能。公司节点不允许删除，如果试图删除公司节点，将弹出不允许删除的提示，在删除部门之前也将弹出提示框询问是否确定删除，如果确定删除，则从系统界面的企业架构树中删除选中部门，并持久化到数据库中。关键代码如下：

例程 36　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\system\FrameworkPanel.java

```
final JButton delButton = new JButton();
delButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        TreePath selectionPath = tree.getSelectionPath();
        int pathCount = selectionPath.getPathCount();          //获得选中节点的级别
        if (pathCount == 1) {                                  //选中的为 1 级节点，即公司节点
            JOptionPane.showMessageDialog(null, "公司节点不能删除！", "友情提示",
                    JOptionPane.WARNING_MESSAGE);
        } else {                                               //选中的为 2 级或 3 级节点，即部门节点
            DefaultMutableTreeNode treeNode = (DefaultMutableTreeNode) selectionPath
                    .getLastPathComponent();                   //获得选中部门节点对象
            int i = JOptionPane.showConfirmDialog(null, "确定要删除该部门："
                    + treeNode, "友情提示", JOptionPane.YES_NO_OPTION);
            if (i == 0) {                                      //删除
❶               treeModel.removeNodeFromParent(treeNode);   //删除选中节点
❷               tree.setSelectionRow(0);                    //选中根（公司）节点
                TbDept selected = company;                  //选中部门的所属部门
                Object[] paths = selectionPath.getPath();   //选中部门节点的路径对象
                int lastIndex = paths.length - 1;           //获得最大索引
                for (int j = 1; j <= lastIndex; j++) {      //遍历选中节点路径
                    Iterator deptIt = selected.getTbDepts().iterator();
                    finded: while (deptIt.hasNext()) {      //通过循环查找选中节点路径对应的部门
                        TbDept dept = (TbDept) deptIt.next();
                        if (dept.getName().equals(paths[j].toString())) {
                            if (j == lastIndex)             //为选中节点
                                selected.getTbDepts().remove(dept);      //删除选中部门
                            else                            //为所属节点
                                selected = dept;
                            break finded;                   //跳出到指定位置
                        }
                    }
                }
                dao.updateObject(company);                  //同步删除数据库
                HibernateSessionFactory.closeSession(); //关闭数据库连接
```

```
                }
            }
        }
});
delButton.setText("删除该部门");
```

代码贴士

❶ removeNodeFromParent(MutableTreeNode node)：该方法用来从树模型中移除指定节点。

❷ setSelectionRow(int row)：该方法用来设置选中的行。树的根节点为第 0 行。例如，A 为树的根节点，a1 和 a2 为 A 的子节点，s 为 a1 的子节点，如果 a1 节点处于合并状态，则 a2 为第 2 行；如果 a1 节点处于展开状态，则 a2 为第 3 行。

3.9.4 单元测试

开发完成企业架构模块后，需要对该模块进行整体测试，这是软件开发过程中的最后一步，也是最关键的一步。

首先测试添加新部门，假设公司需要设立一个后勤部，单击“添加子部门”按钮，在弹出的对话框中输入“后勤部”后单击“确定”按钮，“后勤部”节点即成功添加到树中。

下面测试修改部门名称，将“后勤部”修改为“后勤服务部”。单击“修改名称”按钮，在弹出的对话框中输入“后勤服务部”后单击“确定”按钮，发现在树中仍为“后勤部”，查看控制台，并没有抛出任何异常信息，重新运行程序，发现“后勤部”已经变为了“后勤服务部”，这说明已经修改成功，可是为什么在修改结束后树中并没有改变呢？继续测试删除现有部门，选中“后勤服务部”后单击“删除该部门”按钮，在弹出的提示框中单击“是”按钮，发现删除功能也实现了，在树中已经没有“后勤服务部”节点了。

修改名称后在树结构中并没有显示新名称，但是重新运行系统后却显示了修改后的名称，说明已经修改成功，只是未刷新页面的缘故。

在添加和删除节点时，都是通过 DefaultTreeModel 类的对象 treeModel 实现的。关键代码如下：

例程 37 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\system\FrameworkPanel.java

```
treeModel.removeNodeFromParent(treeNode);
```

例程 38 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\system\FrameworkPanel.java

```
treeModel.insertNodeInto(childNode, parentNode, childCount);
```

修改节点名称则是通过 DefaultMutableTreeNode 类的对象 treeNode 实现的。关键代码如下：

例程 39 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\system\FrameworkPanel.java

```
treeNode.setUserObject(newName);
```

在通过 DefaultTreeModel 类的对象 treeModel 修改树节点时，会同步刷新系统界面，而通过 DefaultMutableTreeNode 类的对象 treeNode 修改节点时，则不会同步刷新系统界面，必须通过 DefaultTreeModel 类的对象 treeModel 进行手动刷新，这样修改后在系统界面就会显示为修改后的内容了。关键代码如下：

例程 40 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\frame\system\FrameworkPanel.java

```
treeModel.reload();
```

3.10　开发技巧与难点分析

由 JTable 类实现的表格是可编辑的，并且表格列允许重新排列。而本系统使用的所有表格都不需要表格的可编辑功能，并且多数表格拥有的列数都很少，也不需要表格列的重新排列功能。基于这种情况，本系统通过继承 JTable 类并重写部分方法，实现了自己的表格组件类 MTable，通过 MTable 类创建的表格是不可编辑的，并且表格列不允许重新排列。

JTable 类的 isCellEditable()方法返回一个 boolean 型值，当返回 true 时，说明表格是可编辑的，所以只需要重写该方法，直接返回 false，这样表格就不可编辑了；通过 JTable 类的 getTableHeader()方法可以获得 JTableHeader 类的对象，通过该对象可以设置表头的相关信息，包括设置表格列是否允许重新排列，设置的方法是 setReorderingAllowed(boolean isAllowed)，当该方法的入口参数为 true 时，表格列允许重新排列，为 false 时则不允许重新排列。MTable 类的完整代码如下：

例程 41　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\mwing\MTable.java

```
public class MTable extends JTable {
    public MTable(DefaultTableModel tableModel) {
        super(tableModel);                                    //继承超类的构造方法
    }
    public boolean isCellEditable(int row, int column) {      //重写该方法，直接返回 false，令表格不可编辑
        return false;
    }
    public JTableHeader getTableHeader() {
        JTableHeader tableHeader = super.getTableHeader();//通过超类方法获得 JtableHeader 类的对象
        tableHeader.setReorderingAllowed(false);              //设置表格列不允许重新排列
        return tableHeader;
    }
}
```

3.11　Hibernate 关联关系的建立方法

在本系统中有多处用到了 Hibernate 的一对一和一对多关联，通过对 Hibernate 关联关系的使用，可以快速地通过一个对象获得与之关联的对象。下面将介绍这两种关联方式的配置方法。

3.11.1　建立一对一关联

本系统将档案信息和职务信息分别保存到了两个表中，如图 3.43 所示，与这两个表对应的持久化类为 TbRecord 和 TbDutyInfo，在这两个持久化类之间就用到了 Hibernate 的一对一关联，因为本系统只允许一个员工担任一个职务。

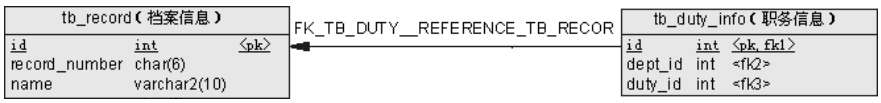


图 3.43　一对一关联模型

（1）在拥有主键的持久化类 TbRecord 中创建一个关联类 TbDutyInfo 的对象 tbDutyInfo 及其对应的 set/get()方法。关键代码如下：

例程 42 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\mapping\TbRecord.java

```
private TbDutyInfo tbDutyInfo;
public TbDutyInfo getTbDutyInfo() {
    return tbDutyInfo;
}
public void setTbDutyInfo(TbDutyInfo tbDutyInfo) {
    this.tbDutyInfo = tbDutyInfo;
}
```

（2）在持久化类 TbRecord 的映射文件 TbRecord.hbm.xml 中添加如下代码：

例程 43 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\mapping\TbRecord.hbm.xml

```
<one-to-one name="tbPersonalInfo" class="com.mwq.hibernate.mapping.TbPersonalInfo" cascade="all" />
```

<one-to-one/>元素用来映射持久化类之间的一对一关联关系，其中 name 属性为持久化类中关联类的对象，class 属性为关联类的类型，cascade 属性用来设置对关联对象的操作级别，当设为 all 时，表示当保存、修改或删除当前对象时，将级联保存、修改或删除关联对象。

（3）在关联类 TbDutyInfo 中创建一个 TbRecord 类的对象 tbRecord 及其对应的 set/get()方法，并且在相应的映射文件中添加如下代码，当将<one-to-one/>元素的 constrained 属性设置为 true 时，说明该类的主键将同时作为外键，参照关联类的主键。关键代码如下：

例程 44 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\mapping\TbDutyInfo.hbm.xml

```
<one-to-one name="tbRecord" class="com.mwq.hibernate.mapping.TbRecord" constrained="true" />
```

既然关联类的主键将同时作为外键，就要修改关联类的主键的映射代码。修改后的代码如下：

例程 45 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\mapping\TbDutyInfo.hbm.xml

```
<id name="id" type="java.lang.Integer">
    <column name="id" />
❶   <generator class="foreign">
❷       <param name="property">tbRecord</param>
    </generator>
</id>
```

代码贴士

❶ foreign：将 class 属性设置为 foreign 时，表示该主键将同时作为外键，所以主键的生成方式将参考外键。

❷ <param>：这里用来设置外键的参考信息，参考的为关联对象 tbRecord 的主键的值。

3.11.2 建立一对多关联

虽然本系统只允许一个员工担任一个职务，即一个员工只能属于一个部门，但是一个部门却可以拥有多个员工，所以部门和员工之间是一对多的关系，即在持久化类 TbDept 和 TbDutyInfo 之间存在

着 Hibernate 的一对多关联关系，如图 3.44 所示。下面将探讨 Hibernate 一对多关联关系的建立方法。

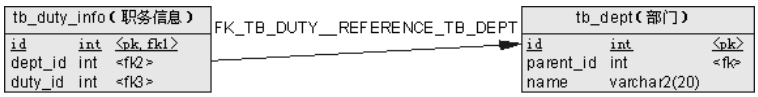


图 3.44　一对多关联模型

（1）在持久化类 TbDutyInfo 中创建一个关联类 TbDept 的对象 tbDept 及其对应的 set/get()方法，并且在相应的映射文件中添加如下代码：

例程 46　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\mapping\ TbDutyInfo.hbm.xml

```
❶  <many-to-one name="tbDept" class="com.mwq.hibernate.mapping.TbDept" fetch="select" lazy="false">
❷      <column name="dept_id" not-null="true" />
   </many-to-one>
```

代码贴士

❶ <many-to-one>：该元素用来映射一对多关联关系中的一方。

❷ <column>：这里用来设置本类对应表中参考关联类对应表中主键的外键列的名称。

（2）在关联类 TbDept 中创建一个集合类 java.util.Set 的对象 tbDutyInfos 及其对应的 set/get()方法，并且在相应的映射文件中添加如下代码：

例程 47　代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\mapping\TbDept.hbm.xml

```
❶  <set name="tbDutyInfos" lazy="false">
❷      <key column="dept_id" />
❸      <one-to-many class="com.mwq.hibernate.mapping.TbDutyInfo" />
   </set>
```

代码贴士

❶ <set>：该元素用来映射一对多关联关系中的多方，同时表明该元素的 name 属性值的类型为 java.util.Set；lazy 属性用来设置对关联对象的检索策略，当设置为 false 时表示立即检索关联对象，默认为 true，即只有当访问关联对象时才检索。

❷ <key>：这里用来设置关联类对应表中参考本类对应表主键的外键的名称。

❸ <one-to-many>：该元素的 class 属性的值为关联类的名称，同时也表明 Set 集合中存放对象的类型。

注意　例程 46 中的<column>元素的 name 属性的值和例程 47 中的<key>元素的 column 属性的值必须相同，均为外键表中的外键列的名称。

至此，一个一对多关联就建立完成了。上面建立的是普通的一对多关联，还有一种特殊的一对多关联，就是一对多自关联。所谓自关联，就是外键参考的为本表中的主键。建立方法与普通的一对多关联完全相同，只是对初学者来说有些难于理解。

通常情况下一个企业的组织架构是呈树状的，即在一个部门当中还可能包含几个下级部门，本系统就支持这种情况，所以本系统中用来保存部门信息的表结构如图 3.45 所示。

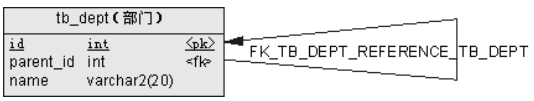


图 3.45　特殊一对多关联模型

针对图 3.45 这种情况建立的一对多关联，就叫做一对多自关联。在这种情况下，在与表 tb_dept

对应的持久化类 TbDept 中既要包含 TbDept 类的对象 tbDept，用来存放该部门对象所属的上级部门对象；又要包含集合类 java.util.Set 的对象 tbDepts，用来存放该部门对象包含的下级部门对象。关键代码如下：

例程 48 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\mapping\TbDept.java

```
private TbDept tbDept;
private Set tbDepts = new HashSet(0);
public TbDept getTbDept() {
    return this.tbDept;
}
public void setTbDept(TbDept tbDept) {
    this.tbDept = tbDept;
}
public Set getTbDepts() {
    return this.tbDepts;
}
public void setTbDepts(Set tbDepts) {
    this.tbDepts = tbDepts;
}
```

同样，在持久化类 TbDept 的映射文件中既要包含<many-to-one>元素，用来映射对象 tbDept，又要包含<set>元素，用来映射对象 tbDepts。关键代码如下：

例程 49 代码位置：光盘\mr\03\PersonnelManage\src\com\mwq\hibernate\mapping\TbDept.hbm.xml

```
<many-to-one name="tbDept" class="com.mwq.hibernate.mapping.TbDept" fetch="select">
    <column name="parent_id" not-null="false" />
</many-to-one>
<set name="tbDepts" lazy="false" cascade="all-delete-orphan">
    <key column="parent_id" />
    <one-to-many class="com.mwq.hibernate.mapping.TbDept" />
</set>
```

3.12 本 章 小 结

本章严格按照软件工程的实施流程，通过一个典型的企业人事管理系统，为读者详细讲解了软件的开发流程。通过本章的学习，读者可以了解到 Java 应用程序的开发流程；企业人事管理系统的软件结构、业务流程和开发过程。在开发过程中要重点掌握 Swing 中表格行选取事件的使用方法、选取并显示图片的方法、页面中组件联动功能的实现方法、对树结构的使用和维护、在程序中调用其他工具软件的方法，以及如何利用现有组件开发一些简单实用的功能模块。另外，还应掌握 Hibernate 持久层技术的使用方法。

第 4 章

酒店管理系统

（Swing+SQL Server 2005 实现）

由于软件技术的不断发展，应用软件已经遍及到社会的各行各业，大到厂矿校企，小到餐饮洗浴，并且正在以其独特的优势，服务于社会的各行各业。将应用软件应用于现代的餐饮业，解决了传统记账、统计、核算方式既费时费力，又容易出错的问题，通过使用酒店管理系统，可以快速完成营业记账工作，并且可以轻松地对营业额进行统计、核算，原来既费时又费力的工作，现在只需要轻点几下鼠标和键盘，就可以轻松完成，既提高了工作效率，又节省了人力资源，为餐饮企业的快速发展创造了巨大空间。

通过阅读本章，可以学习到：

- 酒店管理系统的软件结构和业务流程
- 根据用户输入的部分内容快速获取已点菜品的方法
- 人性化控制点菜数量的方法
- 系统自动结账的实现方法
- Swing 中鼠标或键盘事件的使用方法
- 年、月、日下拉列表框联动保证日期合法性的方法
- 通过正则表达式验证用户输入数据合法性的方法
- 系统时钟的实现方法

4.1 概　　述

“民以食为天”，随着人民生活水平的不断提高，餐饮业在服务行业中的地位也越来越重要，如何从激烈的竞争中脱颖而出，已经成为每位餐饮经营者在思考的问题。

经过多年的发展，对餐饮企业的管理已经逐渐由简单的人工管理，逐步进入到规范、科学管理的阶段。众所周知，在科学管理的具体实现方法中，最有效的工具就是应用管理软件进行管理。

以往的人工操作管理中存在着许多问题，例如：

☑ 人工计算账单容易出现错误。
☑ 收银工作中容易发生账单丢失。
☑ 客人具体消费信息难以查询。
☑ 无法对以往营业数据进行查询。

4.2 系 统 分 析

随着餐饮行业的迅速发展，现有的人工管理方式已不能满足管理者的需求，广大餐饮业经营者已经意识到使用计算机应用软件的重要性，决定在餐饮企业的经营管理上引入计算机应用软件管理系统。

根据餐饮行业的特点和实际情况，酒店管理系统应以餐饮业务为基础，突出前台管理，重视营业数据分析等功能，从专业角度出发，努力为餐饮管理者提供科学、有效的管理模式和数据分析功能。

开台点菜功能是酒店管理系统的最常用功能，也是酒店管理系统的主要功能之一，所以需要将该功能设计得更加人性化和智能化，例如在确定添加菜品时，既可以通过菜品编号确定，又可以通过菜品助记码确定，并且默认添加菜品的数量为一个。

自动结账功能也是酒店管理系统的最常用功能，同样是酒店管理系统的主要功能之一，用户只需要选中结账的台号，系统就会自动为选中的台号计算消费金额，并且用户输入实收金额后，系统还会自动计算出需要找零的金额，这样既节省了系统操作员的精力，又避免了由于计算失误造成的损失。

报表功能是酒店管理系统不可缺少的一部分，因为酒店管理系统是一个记账式软件，对于一个餐饮企业，日报表是不可缺少的。

再就是系统安全和系统维护功能，这是应用软件必不可少的一部分，用来保障软件的安全运行。

4.3 系 统 设 计

4.3.1 系统目标

依据餐饮行业的特点，本系统需要实现以下目标：

☑ 操作简单方便，界面简洁大方。

- ☑ 方便、快捷的开台点菜功能。
- ☑ 智能化定位菜品的功能。
- ☑ 快速查看开台点菜信息的功能。
- ☑ 自动结账功能。
- ☑ 按开台和商品实现的日结账功能。
- ☑ 按日消费额汇总统计实现的月结账功能。
- ☑ 按日营业额实现的年结账功能。
- ☑ 系统运行稳定、安全可靠。

4.3.2　系统功能结构

酒店管理系统的功能结构如图 4.1 所示。

4.3.3　系统预览

酒店管理系统由多个程序界面组成，下面仅列出几个典型界面，其他界面效果请参见光盘中的源程序。

酒店管理系统的主窗体效果如图 4.2 所示，窗体的中间部分用来显示当前的开台及点菜信息，窗体的下方用来操作该系统，例如开台点菜，自动结账，台号、菜系和菜品的维护，营业额报表等。

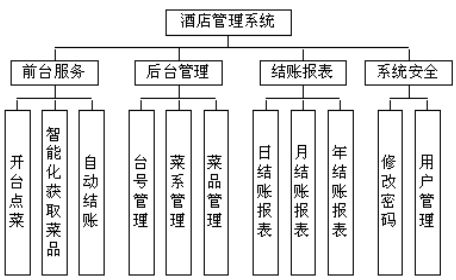


图 4.1　酒店管理系统功能结构

图 4.2　酒店管理系统主窗体效果（光盘\…\TipWizardFrame.java）

单击图 4.2 中右下方的“台号管理”按钮，将打开如图 4.3 所示的“台号管理”对话框，该对话框用来维护台号信息，包括台号及座位数。

单击图 4.2 中右下方的“菜品管理”按钮，将打开如图 4.4 所示的“菜品管理”对话框，该对话框用来维护菜品信息，包括名称、助记码、菜系、单位和单价，其中助记码用来在点菜时快速获取菜品信息（建议设置为菜品名称的首字母，例如将菜品“雪蔓火焰山”的助记码设置为“xmhys”）。

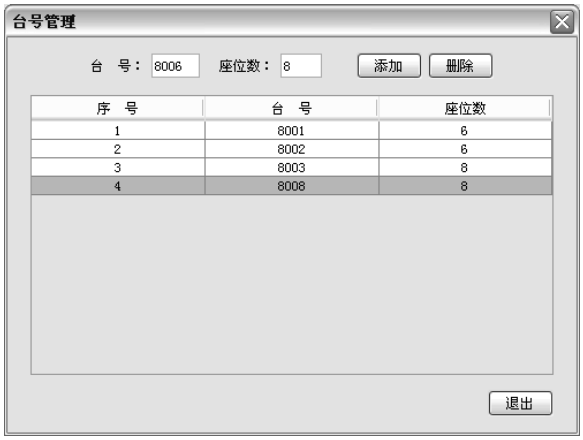


图 4.3　台号管理（光盘\…\manage\DeskNumDialog.java）

图 4.4　菜品管理（光盘\…\manage\MenuDialog.java）

单击图 4.2 中右下方的“日结账”按钮，将打开如图 4.5 所示的“日结账”对话框，该对话框用来统计指定日期的销售情况，包括日营业额和各个商品的日营业额。

说明　由于路径太长，因此省略了部分路径，省略的路径是“TM\04\DrinkeryManage\src\com\mwq\frame”。

4.3.4　业务流程图

酒店管理系统的业务流程如图 4.6 所示。

图 4.5　日结账窗体效果（光盘\…\check_out\DayDialog.java）

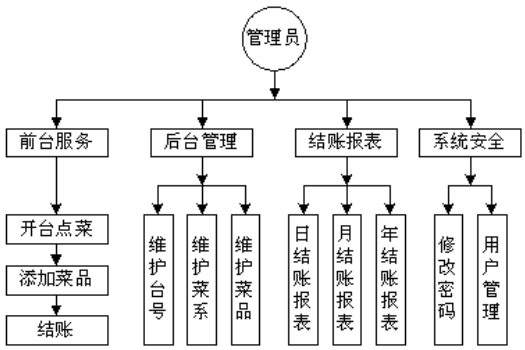


图 4.6　酒店管理系统的业务流程图

4.3.5　文件夹结构设计

每个项目都会有相应的文件夹组织结构。当项目中的窗体过多时，为了便于查找和使用，可以将窗体进行分类，放入不同的文件夹中，这样既便于前期开发，又便于后期维护。酒店管理系统文件夹组织结构如图 4.7 所示。

图 4.7　酒店管理系统文件夹结构

4.4　数据库设计

在开发应用程序时，对数据库的操作是必不可少的，而一个数据库的设计优秀与否，将直接影响到软件的开发进度和性能，所以对数据库的设计就显得尤为重要。数据库的设计要根据程序的需求及其功能制定，如果在开发软件之前不能很好地设计数据库，在开发过程中将反复修改数据库，必将严重影响开发进度。

4.4.1　数据库分析

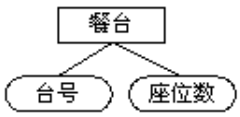
酒店管理系统的需求包括开台点菜功能、智能化获取菜品功能、自动结账功能、营业额报表功能等。在这些功能中主要涉及的数据表包括台号表、菜品表、消费单表；为了使系统更完善，还需要为菜品分类，即需要用到菜系表；为了实现菜品的日销售情况统计，还要建立一个消费项目表，用来记录消费单消费的菜品。

4.4.2　数据库概念设计

数据库设计是系统设计过程中的重要组成部分，它是通过管理系统的整体需求而制定的，数据库设计的好坏直接影响到系统的后期开发。下面对本系统中具有代表性的数据库设计进行详细说明。

餐台和菜系在本系统中是最简单的实体，在本系统中用来描述餐台信息的只有台号和座位数，而描述菜系的主要是名称。餐台信息表的 E-R 图如图 4.8 所示，菜系信息表的 E-R 图如图 4.9 所示。

在描述菜品实体时，加入了助记码，目的是为了实现智能化获取菜品功能，通过这一功能系统操作员可以快速地获取顾客所点的菜品信息。菜品信息表的 E-R 图如图 4.10 所示。



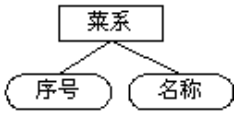


图 4.8　餐台信息表 E-R 图　图 4.9　菜系信息表 E-R 图

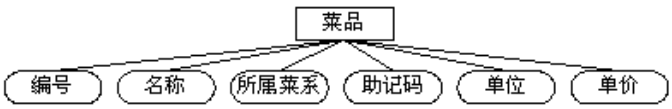


图 4.10　菜品信息表 E-R 图

消费单用来记录每次消费的相关信息，例如消费时使用的餐台、消费时间、消费金额等。消费单信息表的 E-R 图如图 4.11 所示。

消费项目用来记录每个消费单消费的菜品，记录的主要信息有所属消费单、消费菜品的名称、消费数量、消费额。消费项目信息表的 E-R 图如图 4.12 所示。

图 4.11　消费单信息表 E-R 图

图 4.12　消费项目信息表 E-R 图

4.4.3　数据库逻辑结构设计

在数据库概念设计中已经分析了菜品、消费单、消费项目等主要的数据库实体对象，这些实体对象是数据表结构的基本模型，最终的数据模型都要实施到数据库中，形成整体的数据结构，可以使用 PowerDesigner 工具完成这个数据库的建模。具体的模型结构如图 4.13 所示。

4.4.4　视图设计

完成数据库建模后，还可以根据实际需要建立一些视图，通过对视图的应用，可以减少在程序中编写复杂的 SQL 语句。在开发酒店管理系统的日结账功能时，需要查询指定日期的所有消费单，然后根据消费单查询消费项目并关联查询项目名称，所以为表 tb_menu 和表 tb_order_item 建立了一个视图 v_order_item_and_menu，如图 4.14 所示。

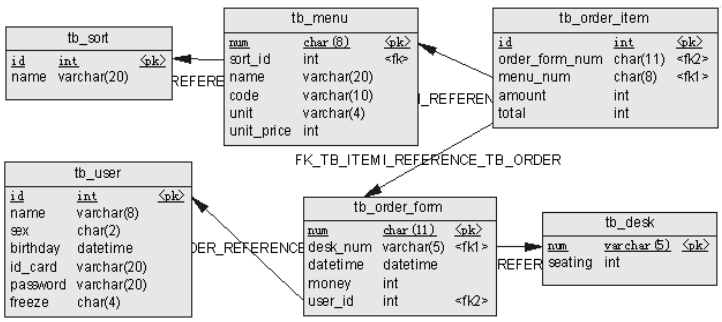


图 4.13　酒店管理系统数据库模型

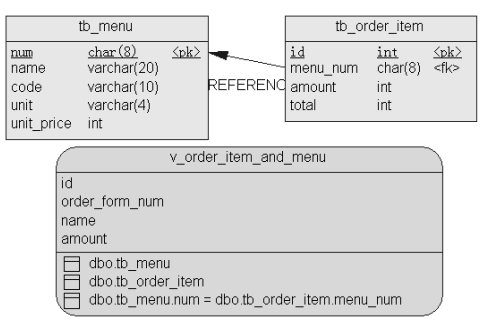


图 4.14　酒店管理系统用到的视图

4.5　公共模块设计

4.5.1　编写数据库连接类

数据库连接类负责加载数据库驱动程序，以及创建和关闭数据库连接，为了最大程度地应用每个已经创建的数据库连接，这里将其保存到了 ThreadLocal 类的对象中。

（1）在数据库连接类中定义一些常量，包括连接数据库使用的驱动程序、连接数据库的路径、连接数据库使用的用户名和密码，并且定义一个 ThreadLocal 类的对象，用来保存已经创建的数据库连接。关键代码如下：

例程 01　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\dao\JDBC.java

```
private static final String DRIVERCLASS = "com.microsoft.jdbc.sqlserver.SQLServerDriver";   //数据库驱动
private static final String URL = "jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_DrinkeryManage";
                                                                                            //路径
private static final String USERNAME = "sa";                                                //连接数据库的用户名
private static final String PASSWORD = "";                                                  //连接数据库的密码
private static final ThreadLocal<Connection> threadLocal = new ThreadLocal<Connection>();//创建保存连接的对象
```

（2）编写用来加载数据库驱动程序的代码，通常情况下将其放到静态代码块中，这样做的好处是只在该类第一次被加载（即第一次被调用）时执行加载数据库驱动程序的动作，避免了反复加载数据库驱动程序，从而提高软件的性能。关键代码如下：

例程 02　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\dao\JDBC.java

```
static {                                                                    //通过静态方法加载数据库驱动
    try {
        Class.forName(DRIVERCLASS).newInstance();                           //加载数据库驱动
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

（3）编写用来创建和关闭数据库连接的方法，这里将这两个方法均定义为静态的，这样通过类名就可以调用方法，方便使用。在这两个方法中首先从 ThreadLocal 类的对象中获得数据库连接，然后判断是否存在可用的数据库连接，如果存在则直接返回或关闭，否则重新创建。关键代码如下：

例程 03　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\dao\JDBC.java

```
public static Connection getConnection() {                                  //创建数据库连接的方法
    Connection conn = threadLocal.get();                                    //从线程中获得数据库连接
    if (conn == null) {                                                     //没有可用的数据库连接
        try {
            conn = DriverManager.getConnection(URL, USERNAME, PASSWORD);//创建新的数据库连接
            threadLocal.set(conn);                                          //将数据库连接保存到线程中
        } catch (SQLException e) {
```

```
                e.printStackTrace();
            }
        }
        return conn;
    }
    public static boolean closeConnection() {                    //关闭数据库连接的方法
        boolean isClosed = true;                                 //默认关闭成功
        Connection conn = threadLocal.get();                     //从线程中获得数据库连接
        threadLocal.set(null);                                   //清空线程中的数据库连接
        if (conn != null) {                                      //数据库连接可用
            try {
                conn.close();                                    //关闭数据库连接
            } catch (SQLException e) {
                isClosed = false;                                //关闭失败
                e.printStackTrace();
            }
        }
        return isClosed;
    }
```

4.5.2 封装常用的操作数据库的方法

对数据库的操作包括查询、添加、修改和删除，其中查询是通过 executeQuery(String sql)方法执行 SQL 语句实现的，添加、修改和删除是通过 executeUpdate(String sql)方法执行 SQL 语句实现的。在本系统中共提供了 4 个用来执行查询的方法，分别用来查询多个记录、查询指定记录、查询多个记录的指定值和查询指定记录的指定值；一个用来添加、修改和删除记录的方法。由于篇幅有限，在这里只介绍用来查询多个记录的方法，以及用来添加、修改和删除记录的方法。

下面的方法用来查询多个记录。为了将检索结果直接应用于表格组件，这里全部用 Vector 向量对象封装查询结果，并且为代表每一行的向量添加了行序号。关键代码如下：

例程 04 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\dao\BaseDao.java

```
protected Vector selectSomeNote(String sql) {
    Vector<Vector<Object>> vector = new Vector<Vector<Object>>();   //创建结果集向量
    Connection conn = JDBC.getConnection();                         //获得数据库连接
    try {
        Statement stmt = conn.createStatement();                    //创建连接状态对象
        ResultSet rs = stmt.executeQuery(sql);                      //执行 SQL 语句获得查询结果
        int columnCount = rs.getMetaData().getColumnCount();        //获得查询数据表的列数
        int row = 1;                                                //定义行序号
        while (rs.next()) {                                         //遍历结果集
            Vector<Object> rowV = new Vector<Object>();             //创建行向量
            rowV.add(new Integer(row++));                           //添加行序号
            for (int column = 1; column <= columnCount; column++) {
                rowV.add(rs.getObject(column));                     //添加列值
            }
            vector.add(rowV);                                       //将行向量添加到结果集向量中
```

```
            }
            rs.close();                                          //关闭结果集对象
            stmt.close();                                        //关闭连接状态对象
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return vector;                                           //返回结果集向量
}
```

下面的方法用来添加、修改和删除记录。这里采用手动提交，目的是捕获持久化异常，并回退数据库，以保证数据的合法性。关键代码如下：

例程 05　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\dao\BaseDao.java

```
public boolean longHaul(String sql) {
    boolean isLongHaul = true;                                   //默认持久化成功
    Connection conn = JDBC.getConnection();                      //获得数据库连接
    try {
        conn.setAutoCommit(false);                               //设置为手动提交
        Statement stmt = conn.createStatement();                 //创建连接状态对象
        stmt.executeUpdate(sql);                                 //执行 SQL 语句
        stmt.close();                                            //关闭连接状态对象
        conn.commit();                                           //提交持久化
    } catch (SQLException e) {
        isLongHaul = false;                                      //持久化失败
        try {
            conn.rollback();                                     //回退
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
        e.printStackTrace();
    }
    return isLongHaul;                                           //返回持久化结果
}
```

4.5.3　自定义表格组件

在使用表格时，通常情况下将表格列的值设置为居中显示，这样看起来更美观。由 JTable 组件实现的表格将一个表格分为了两个部分，一部分是表格头，即用来显示表格列名的部分；另一部分是除表格头之外的部分。表格的每一部分对应一个 DefaultTableCellRenderer 类的对象，即单元格对象，通过该对象可以设置所属表格部分包含单元格的相关属性，例如设置单元格内容的显示位置。

通过 JTable 类的 getDefaultRenderer()方法可以得到单元格对象，这个单元格对象代表的是除表格头之外的部分包含的单元格，在 MTable 类中对该方法进行了简单的重构，即通过单元格对象设置单元格内容水平为居中显示。关键代码如下：

例程 06　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\mwing\MTable.java

```
public TableCellRenderer getDefaultRenderer(Class<?> columnClass) {
    //获得除表格头之外部分的单元格对象
```

```
❶  DefaultTableCellRenderer tableRenderer = (DefaultTableCellRenderer) super.getDefaultRenderer
(columnClass);                                          //设置单元格内容居中显示
❷  tableRenderer.setHorizontalAlignment(DefaultTableCellRenderer.CENTER);
    return tableRenderer;
}
```

代码贴士

❶ getDefaultRenderer(columnClass)：该方法用来获得表格的单元格对象。

❷ setHorizontalAlignment(int alignment)：该方法用来设置单元格内容的显示位置，可选的静态常量有 LEFT（靠左侧显示）、CENTER（居中显示）和 RIGHT（靠右侧显示）。

如果想设置表格头的相关绘制属性，首先要获得表格头对象，即 JTableHeader 类的对象，通过 JTableHeader 类的 setReorderingAllowed(boolean reorderingAllowed)方法可以设置表格列是否可以重排；通过 JTableHeader 类的 getDefaultRenderer()方法也可以得到单元格对象，这个单元格对象代表的是表格头包含的单元格，在 MTable 类中对该方法也进行了简单的重构，即通过单元格对象设置单元格内容（即列名）水平为居中显示。关键代码如下：

例程 07 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\mwing\MTable.java

```
public JTableHeader getTableHeader() {
    //获得表格头对象
    JTableHeader tableHeader = super.getTableHeader();
❶   tableHeader.setReorderingAllowed(false);          //设置表格列不可重排
    //获得表格头的单元格对象
❷   DefaultTableCellRenderer headerRenderer = (DefaultTableCellRenderer) tableHeader.getDefaultRenderer();
    //设置单元格内容（即列名）居中显示
    headerRenderer.setHorizontalAlignment(DefaultTableCellRenderer.CENTER);
    return tableHeader;
}
```

代码贴士

❶ setReorderingAllowed(boolean reorderingAllowed)：该方法用来设置表格头是否可以重新排列，即表格的列是否可以移动，默认为 true，即默认可以移动。

❷ getDefaultRenderer()：该方法用来获得表格头（即显示列名的位置）的单元格对象。

JTable 类的 setRowSelectionInterval(int fromRow, int toRow)方法用来设置表格中选中的行，第一个参数为开始选中行的索引，第二个参数为结束选中行的索引。在本系统中表格的选择模式为单选，如果通过该方法设置，需要设置两个参数，所以在 MTable 类中实现了一个重载方法 setRowSelectionInterval(int row)，用来设置唯一被选中的行，该方法仍然需要调用前面的方法。关键代码如下：

例程 08 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\mwing\MTable.java

```
public void setRowSelectionInterval(int row) {  //重载方法
    setRowSelectionInterval(row, row);
}
```

4.5.4　编写利用正则表达式验证数据合法性的方法

在获得用户的输入内容时，经常需要验证用户输入数据的合法性，例如对用户输入日期的验证。验证这一类数据时，较好的办法是利用正则表达式，所以本系统提供了一个可重用的利用正则表达式验证数据合法性的方法，每次使用时只需要传入验证规则和验证内容，返回值为 boolean 型，当返回 true 时表示验证通过，数据合法；当返回 false 时表示验证未通过，数据不合法。关键代码如下：

例程 09　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\tool\Validate.java

```
public static boolean execute(String rule, String content) {
    Pattern pattern = Pattern.compile(rule);          //利用验证规则创建 Pattern 对象
    Matcher matcher = pattern.matcher(content);       //利用验证内容获得 Matcher 对象
    return matcher.matches();                         //返回验证结果
}
```

4.6　主窗体设计

本系统将主窗体划分为 6 个工作区，分别是开台签单工作区、自动结账工作区、后台管理工作区、结账报表工作区、系统安全工作区和系统提示区。酒店管理系统主窗体效果如图 4.15 所示。

图 4.15　酒店管理系统主窗体效果

在开台签单工作区中使用了分割面板，这样系统操作员可以根据实际需要，调整开台列表和签单列表的大小，并设置分割面板支持快速展开/折叠的分割条，效果如图 4.16 所示，既可以将光标移动到分割条的上方随意调整分割条的位置，又可以通过单击分割条上的◀或▶按钮将分割条移动到分割面板的最左侧或最右侧，单击另一个则使分割条恢复到原位置；不支持快速展开/折叠的分割条效果如图 4.17 所示。

实现图 4.16 所示分割面板的关键代码如下：

例程 10　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
    final JSplitPane splitPane = new JSplitPane();                    //创建分割面板对象
❶   splitPane.setOrientation(JSplitPane.HORIZONTAL_SPLIT);            //设置为水平分割
❷   splitPane.setDividerLocation(755);                                //设置面板默认的分割位置
❸   splitPane.setDividerSize(10);                                     //设置分割条的宽度
❹   splitPane.setOneTouchExpandable(true);                           //设置为支持快速展开/折叠分割条
    splitPane.setBorder(new TitledBorder(null, "",
            TitledBorder.DEFAULT_JUSTIFICATION,
```

```
            TitledBorder.DEFAULT_POSITION, null, null));          //设置面板的边框
    getContentPane().add(splitPane, BorderLayout.CENTER);         //将分割面板添加到上级容器中
    final JPanel leftPanel = new JPanel();                        //创建放于分割面板左侧的普通面板对象
❺   splitPane.setLeftComponent(leftPanel);                        //将普通面板对象添加到分割面板的左侧
    …//此处省略了部分代码
    final JPanel rightPanel = new JPanel();                       //创建放于分割面板右侧的普通面板对象
❻   splitPane.setRightComponent(rightPanel);                      //将普通面板对象添加到分割面板的右侧
```

代码贴士

❶ setOrientation(int orientation)：该方法用于设置分割面板的分割方向，默认为在水平方向分割，即 HORIZONTAL_SPLIT；如果希望在垂直方向分割，则需要设置为 VERTICAL_SPLIT。

❷ setDividerLocation(int location)：该方法用于设置分割条的显示位置，入口参数为针对分割面板的绝对位置；该方法拥有一个重载方法 setDividerLocation(double proportionalLocation)，用来以百分比的形式设置分割条的显示位置，proportionalLocation 为 0～1.0 的双精度浮点值。

❸ setDividerSize (int width)：该方法用于设置分割条的宽度，默认宽度为 5 像素。

❹ setOneTouchExpandable(boolean expandable)：该方法用于设置是否支持快速展开/折叠的分割条，默认为 false，即不支持。需要注意的是，有些外观可能不支持该功能。

❺ setLeftComponent(Component comp)：该方法用来将指定的组件设置到分割面板的左侧或上方，等同于方法 setTopComponent(Component comp)。

❻ setRightComponent(Component comp)：该方法用来将指定的组件设置到分割面板的右侧或下方，等同于方法 setBottomComponent(Component comp)。

在系统提示区提供了时钟的功能，如图 4.18 所示。

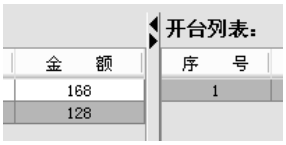


图 4.16　支持快速展开/折叠的分割条

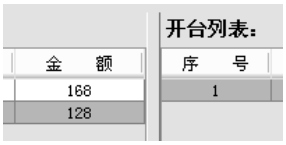


图 4.17　不支持快速展开/折叠的分割条

图 4.18　系统提示区的时钟

系统提示区的时钟是显示到标签组件上的。对标签组件的具体设置代码如下：

例程 11　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
timeLabel = new JLabel();                                         //创建用于显示时间的标签对象
timeLabel.setFont(new Font("宋体", Font.BOLD, 14));               //设置标签中的文字为宋体、粗体、14 号
timeLabel.setForeground(new Color(255, 0, 0));                    //设置标签中的文字为红色
timeLabel.setHorizontalAlignment(SwingConstants.CENTER);          //设置标签中的文字居中显示
clueOnPanel.add(timeLabel);                                       //将标签添加到上级容器中
new Time().start();                                               //开启线程
```

下面在 TipWizardFrame 类中创建一个内部类 Time，该类继承了线程类 Thread，并重构了 run()方法，每隔一秒修改一次用于显示时间标签中的时间信息。关键代码如下：

例程 12　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
class Time extends Thread {                                       //创建内部类
    public void run() {                                           //重构父类的方法
        while (true) {
```

```
                Date date = new Date();                                    //创建日期对象
                timeLabel.setText(date.toString().substring(11, 19));      //获取当前时间，并显示到时间标签中
                try {
                    Thread.sleep(1000);                                    //令线程休眠 1 秒
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

4.7　用户登录窗口设计

用户登录窗口是每一个应用软件都不可缺少的部分，其主要功能是保证用户的数据安全；同时用户登录窗口也是用户看到的第一个系统界面，因此，一个设计优秀的用户登录窗口，将有效地提高用户对系统的第一印象。

用户登录窗口的设计优秀与否，主要包括以下几个方面：

☑　美观大方。
☑　简单易懂。
☑　安全性高。
☑　使用方便。

要使用户登录界面美观大方，就离不开对图片的使用，但是 JPanel 类并不支持绘制背景图片的功能，即将组件绘制到图片的上方，可以通过重写 JPanel 类的 paintComponent(Graphics g)方法，实现支持绘制背景图片的功能。关键代码如下：

例程 13　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\mwing\MPanel.java

```
public class MPanel extends JPanel {
    private ImageIcon imageIcon;                                   //声明一个图片对象
    public MPanel(URL imgUrl) {
        super();                                                   //继承父类的构造方法
        setLayout(new GridBagLayout());                            //将布局管理器修改为网格组布局
        imageIcon = new ImageIcon(imgUrl);                         //根据传入的 URL 创建 ImageIcon 对象
        setSize(imageIcon.getIconWidth(), imageIcon.getIconHeight());//设置面板与图片等大
    }
    protected void paintComponent(Graphics g) {                    //重写 Jpanel 类的 paintComponent()方法
        super.paintComponent(g);                                   //调用 Jpanel 类的 paintComponent()方法
        Image image = imageIcon.getImage();                        //通过 ImageIcon 对象获得 Image 对象
        g.drawImage(image, 0, 0, null);                            //绘制 Image 对象，即将图片绘制到面板中
    }
}
```

利用通过继承 JPanel 类得到的 MPanel 类，就可以很方便地将图片设置为面板的背景图片了。这样，在设计用于用户登录界面的背景图片时，就可以将一些辅助信息设计到图片上了，如图 4.19 所示就是

为本系统的用户登录界面设计的背景图片。

利用图 4.19 所示图片作为用户登录界面的背景图片，在绘制用户登录界面时，就只需要利用代码绘制一个下拉列表框、一个密码框和 3 个按钮就可以了。绘制完成后的本系统登录界面效果如图 4.20 所示。

图 4.19 为用户登录界面设计的背景图片

图 4.20 酒店管理系统用户登录界面

首先创建一个用于用户登录界面的窗体，为窗体设置标题、大小等信息，并将一个支持背景图片功能的面板添加到窗体中，背景图片即图 4.18 所示的图片。关键代码如下：

例程 14 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\LandFrame.java

```
public class LandFrame extends JFrame {
    private JPasswordField passwordField;                          //密码框
    private JComboBox usernameComboBox;                            //用户名下拉列表框
    public LandFrame() {
        //首先设置窗口的相关信息
        super();                                                   //调用父类的构造方法
        setTitle(" T 科技");                                       //设置窗口的标题
❶       setResizable(false);                                       //设置窗口不可以改变大小
❷       setAlwaysOnTop(true);                                      //设置窗口总在最前方
        setBounds(100, 100, 428, 292);                             //设置窗口的大小
❸       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);            //设置当关闭窗口时执行的动作
        //下面将创建一个面板对象并添加到窗口的容器中
        final MPanel panel = new MPanel(this.getClass().getResource(
                "/img/land_background.jpg"));                      //创建一个面板对象
        panel.setLayout(new GridBagLayout());                      //设置面板的布局管理器为网格组布局
        getContentPane().add(panel, BorderLayout.CENTER);          //将面板添加到窗体中
        …//此处省略了部分代码
    }
}
```

代码贴士

❶ setResizable(boolean resizable)：该方法用于设置窗口的大小是否可以调整，默认为 true，即在默认情况下可以调整；当设置为 false 时，窗口大小将不可以调整，并且最大化按钮将不可用。

❷ setAlwaysOnTop(boolean alwaysOnTop)：该方法用于设置窗口是否永远在屏幕的最前方，默认为 false；当设置为 true 时，窗口将永远在屏幕的最前方。

❸ setDefaultCloseOperation(int operation)：该方法用于设置当用户关闭此窗体时执行的操作。必须为以下 4 个静态常量之一：（1）DO_NOTHING_ON_CLOSE（常量值为 0）——不执行任何操作；（2）HIDE_ON_CLOSE（常量值

为 1）——隐藏该窗体；（3）DISPOSE_ON_CLOSE（常量值为 2）——释放该窗体；（4）EXIT_ON_CLOSE（常量值为 3）——退出应用程序。默认值为静态常量 HIDE_ON_CLOSE。

下面将依次创建一个下拉列表框组件和一个密码框组件，并利用网格组布局管理器将它们添加到背景面板中，分别用于选择登录用户和输入登录密码。关键代码如下：

例程 15　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\LandFrame.java

```
//创建并设置用户名下拉菜单
usernameComboBox = new JComboBox();                                    //创建用户名下拉列表框组件对象
usernameComboBox.setMaximumRowCount(5);                                //设置下拉列表框最多可显示的选项数
usernameComboBox.addItem("请选择");                                    //为下拉列表框添加提示项
usernameComboBox
        .addActionListener(new UsernameComboBoxActionListener());//为下拉列表框添加事件监听器
final GridBagConstraints gridBagConstraints = new GridBagConstraints();//创建网格组布局管理器对象
gridBagConstraints.anchor = GridBagConstraints.WEST;                   //设置为靠左侧显示
gridBagConstraints.gridy = 1;                                          //设置行索引为 1
gridBagConstraints.gridx = 2;                                          //设置列索引为 2
panel.add(usernameComboBox, gridBagConstraints);                  //将组件按指定的布局管理器添加到面板中
//创建并设置密码框
passwordField = new JPasswordField();                                  //创建密码框组件对象
passwordField.setColumns(20);                                          //设置密码框可显示的字符数
passwordField.setText("      ");                                       //设置密码框默认显示 6 个空格
passwordField.addFocusListener(new PasswordFieldFocusListener());     //为密码框添加焦点监听器
final GridBagConstraints gridBagConstraints_1 = new GridBagConstraints();   //创建网格组布局管理器对象
gridBagConstraints_1.insets = new Insets(5, 0, 0, 0);                  //设置组件外部上方的填充量为 5 像素
gridBagConstraints_1.anchor = GridBagConstraints.WEST;                 //设置为靠左侧显示
gridBagConstraints_1.gridy = 2;                                        //设置行索引为 2
gridBagConstraints_1.gridx = 2;                                        //设置列索引为 2
panel.add(passwordField, gridBagConstraints_1);                   //将组件按指定的布局管理器添加到面板中
```

下面将创建一个用来显示按钮的面板，该面板的背景必须为透明的，否则将遮盖背景图片。该面板采用的布局管理器为水平箱布局，并且其占用网格组布局模式的两列。关键代码如下：

例程 16　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\LandFrame.java

```
//创建并设置一个用来添加 3 个按钮的面板
final JPanel buttonPanel = new JPanel();                               //创建一个用来添加按钮的面板
buttonPanel.setOpaque(false);                                          //设置面板的背景为透明
buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.X_AXIS));   //设置面板采用水平箱布局
final GridBagConstraints gridBagConstraints_4 = new GridBagConstraints();   //创建网格组布局管理器对象
gridBagConstraints_4.insets = new Insets(10, 0, 0, 0);            //设置组件外部上方的填充量为 10 像素
gridBagConstraints_4.gridwidth = 2;                                    //设置其占两列
gridBagConstraints_4.gridy = 3;                                        //设置行索引为 3
gridBagConstraints_4.gridx = 1;                                        //设置列索引为 1
panel.add(buttonPanel, gridBagConstraints_4);                     //将组件按指定的布局管理器添加到面板中
```

用来绘制“登录”、“重置”和“退出”按钮的代码基本相同，所以在这里只介绍用来绘制“登录”按钮的代码。因为按钮的显示内容均是通过图片实现的，所以在创建按钮对象时，建议设置为不

绘制按钮的边框，也不绘制按钮的内容区域，目的是使按钮变为透明的，避免在按钮图片的四周出现不希望的绘制效果；并且建议将按钮边框和标签之间的间隔设置为 0。还可以为不同状态的按钮设置不同效果的图片，如图 4.21 所示为默认情况下按钮的效果，图 4.22 中的“登录”按钮则展示了当光标位于按钮上方时按钮的效果，图 4.23 中的“登录”按钮则展示了当按钮被按下时按钮的效果。

用来绘制“登录”按钮的关键代码如下：

例程 17 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\LandFrame.java

```
    final JButton landButton = new JButton();                    //创建“登录”按钮组件对象
❶   landButton.setMargin(new Insets(0, 0, 0, 0));                //设置按钮边框和标签之间的间隔
❷   landButton.setContentAreaFilled(false);                       //设置不绘制按钮的内容区域
❸   landButton.setBorderPainted(false);                          //设置不绘制按钮的边框
    URL landUrl = this.getClass().getResource("/img/land_submit.png");
                                                                 //获得默认情况下“登录”按钮显示图片的 URL
❹   landButton.setIcon(new ImageIcon(landUrl));                  //设置默认情况下“登录”按钮显示的图片
    URL landOverUrl = this.getClass().getResource(
            "/img/land_submit_over.png");                        //获得当鼠标经过“登录”按钮时显示图片的 URL
❺   landButton.setRolloverIcon(new ImageIcon(landOverUrl));//设置当鼠标经过“登录”按钮时显示的图片
    URL landPressedUrl = this.getClass().getResource(
            "/img/land_submit_pressed.png");                     //获得当“登录”按钮被按下时显示图片的 URL
❻   landButton.setPressedIcon(new ImageIcon(landPressedUrl));//设置当“登录”按钮被按下时显示的图片
    landButton.addActionListener(new LandButtonActionListener());      //为“登录”按钮添加事件监听器
    buttonPanel.add(landButton);                                 //将“登录”按钮添加到用来添加按钮的面板中
```

代码贴士

❶ setMargin(Insets m)：该方法用于设置按钮边框和标签之间的间隔。该方法接受 Insets 类的实例，Insets 类仅提供了一个构造方法 Insets(int top, int left, int bottom, int right)，4 个入口参数依次为按钮上方、左侧、下方和右侧的间距。

❷ setContentAreaFilled(boolean contentAreaFilled)：该方法用于设置是否绘制按钮的内容区域，默认为 true，即默认为绘制；如果为按钮设置了图片，则在图片的四周可能出现灰色边框，所以在这里设置为不绘制。

❸ setBorderPainted(boolean borderPainted)：该方法用于设置是否绘制按钮的边框，默认为 true，即默认为绘制；如果为按钮设置了图片，则建议设置为 false，即不绘制按钮边框。

❹ setIcon(Icon defaultIcon)：该方法用于设置按钮在默认情况下显示的图片。

❺ setRolloverIcon(Icon rolloverIcon)：该方法用于设置当光标在按钮上方时显示的图片。

❻ setPressedIcon(Icon pressedIcon)：该方法用于设置当按钮被按下时显示的图片。

当第一次使用本系统时，在数据库中将不存在系统管理员。在这种情况下，系统将提供一个默认用户，供用户登录后添加管理员，如图 4.24 所示。添加管理员后，将不再提供系统默认用户，如图 4.25 所示。

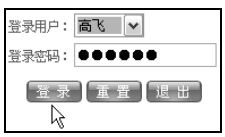


图 4.21 光标不在按钮上方

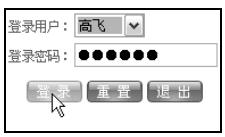


图 4.22 光标在按钮上方

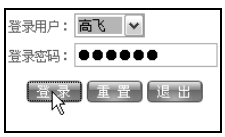


图 4.23 按钮被按下的瞬间

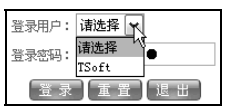


图 4.24 系统默认用户

图 4.25 系统管理员

用户单击“登录”按钮后，系统将首先判断是否选择了登录用户，然后再判断是系统默认用户，还是系统管理员，最后验证登录密码，如果通过验证则登录成功，否则登录失败并弹出提示。“登录”

按钮的关键代码如下：

例程 18　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\LandFrame.java

```
class LandButtonActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String username = usernameComboBox.getSelectedItem().toString(); //获得登录用户的名称
        if (username.equals("请选择")) {                                  //查看是否选择了登录用户
            JOptionPane.showMessageDialog(null, "请选择登录用户！", "友情提示",
                    JOptionPane.INFORMATION_MESSAGE);      //弹出提示
            resetUsernameAndPassword();                                   //恢复登录用户和登录密码
        }
        char[] passwords = passwordField.getPassword();                   //获得登录用户的密码
        String inputPassword = turnCharsToString(passwords);              //将密码从 char 型数组转换成字符串
        if (username.equals("TSoft")) {                                   //查看是否为默认用户登录
            if (inputPassword.equals("111")) {                            //查看密码是否为默认密码
                land(null);                                               //登录成功
                String infos[] = { "请立刻单击“用户管理”按钮添加用户！",
                        "添加用户后需要重新登录，本系统才能正常使用！" };      //组织提示信息
                JOptionPane.showMessageDialog(null, infos, "友情提示",
                        JOptionPane.INFORMATION_MESSAGE);                 //弹出提示
            } else {                                                      //密码错误
                JOptionPane.showMessageDialog(null, "默认用户“TSoft”的登录密码为“111”！",
                        "友情提示", JOptionPane.INFORMATION_MESSAGE);    //弹出提示
                passwordField.setText("111");                             //将密码设置为默认密码
            }
        } else {
            if (inputPassword.length() == 0) {                            //用户未输入登录密码
                JOptionPane.showMessageDialog(null, "请输入登录密码！", "友情提示",
                        JOptionPane.INFORMATION_MESSAGE);      //弹出提示
                resetUsernameAndPassword();                               //恢复登录用户和登录密码
            }
            Vector user = Dao.getInstance().sUserByName(username);     //查询登录用户
            String password = user.get(5).toString();                     //获得登录用户的密码
            if (inputPassword.equals(password)) {                         //查看登录密码是否正确
                land(user);                                               //登录成功
            } else {                                                      //登录密码错误
                JOptionPane.showMessageDialog(null, "登录密码错误，请确认后重新登录！",
                        "友情提示", JOptionPane.INFORMATION_MESSAGE);    //弹出提示
                resetUsernameAndPassword();                               //恢复登录用户和登录密码
            }
        }
    }
    private void resetUsernameAndPassword() {                             //恢复登录用户和登录密码
        usernameComboBox.setSelectedIndex(0);                             //恢复选中的登录用户为“请选择”项
        passwordField.setText("      ");                                  //恢复密码框的默认值为 6 个空格
        return;                                                           //直接返回
    }
    private void land(Vector user) {                                      //登录成功
        TipWizardFrame tipWizard = new TipWizardFrame(user);              //创建主窗体对象
        tipWizard.setVisible(true);                                       //设置主窗体可见
```

```
            setVisible(false);                                    //设置登录窗口不可见
        }
}
```

4.8 开台签单工作区设计

开台签单工作区是本系统最常用的工作区，所以需要将该工作区设计得更人性化和智能化。例如，在获取欲添加的菜品时，既可以通过菜品编号获得，又可以通过菜品助记码获得，并且默认菜品的数量为一个，因为这是最通用的。

4.8.1 开台签单工作区的功能概述

开台签单工作区的主要功能有开台、点菜、加菜、签单、查看开台信息和签单信息，开台签单工作区的效果如图 4.26 所示。

图 4.26 开台签单工作区效果

当顾客要求开台用餐时，首先在图 4.26 下方的“台号”下拉列表框中选择分配的餐台号，然后选择获取顾客点菜的方式，默认为通过助记码获取，也可以通过编号获取，这里假设以助记码获取。在输入商品助记码的过程中会在“商品名称”文本框中显示匹配商品的名称，并在“单位”文本框中显示该商品的销售单位。当在“商品名称”文本框中显示的为顾客所点的菜品时，如果点菜数量为 1，可以通过按 Enter 键将该菜品添加到签单列表中；如果需要修改数量，修改后则需要通过单击“开单”按钮添加到签单列表中。新点的菜品在表格的最前方会显示 NEW，在这种情况下可以选中后单击“取消”按钮取消该菜品。最后，在点菜结束后单击“签单”按钮，新点菜品前方的 NEW 会消失，在这种情况下将不允许取消，至此点菜完成。

如果顾客在用餐的过程中要求添加菜品，既可以在“台号”下拉列表框中选择要求添加菜品的餐台号后添加，也可以在“开台列表”中选择要求添加菜品的餐台号，因为它与“台号”下拉列表框是联动的，即当在“台号”下拉列表框中选择餐台号后，如果在“开台列表”中存在该台号，对应的行也将被选中；如果更改“开台列表”中的选中行，在“台号”下拉列表框中也将更改为选中的餐台号。

4.8.2 开台签单工作区技术分析

在开发开台签单工作区时，为了使系统更人性化、智能化，要充分利用各种事件监听器。

例如为“台号”下拉列表框添加 ActionListener 监听器，用来捕获下拉列表框选项被改变的事件，

目的是同步处理“开台列表”和“签单列表”中的信息；为“商品（编号/助记码）”文本框添加 KeyListener 监听器，用来捕获在该文本框中按键被释放的事件，目的是跟踪用户输入内容同步获取最接近的商品，尽量让用户输入最少的内容就能得到需要的商品；为“数量”文本框添加 FocusListener 监听器，用来捕获该文本框获得或失去焦点的事件，因为默认数量为 1，当获得焦点时将数量设置为空，用户在改变数量时就不用先删除默认数量 1 了，当失去焦点时，查看用户是否输入了数量，如果未输入，则恢复默认数量 1；为“开台列表”表格添加 MouseListener 监听器，用来捕获表格行被选中的事件，目的是同步处理“签单列表”和“台号”下拉列表框中的信息。

4.8.3　开台签单工作区的实现过程

开台签单使用的主要数据表：tb_desk、tb_menu

（1）为“台号”下拉列表框添加事件监听器，用来处理开台或点菜的相关信息。如果选中的台号尚未开台（即新开台），则取消选择“开台列表”中的选中行，并清空“签单列表”中的所有行；如果选中的台号已经开台（即添加菜品），并且在“开台列表”中该台号未被选中，则选中“开台列表”中的该台号，并刷新“签单列表”中的菜品信息，即显示为当前选中台号所点的菜品。关键代码如下：

例程 19　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
numComboBox.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
❶       int rowCount = rightTable.getRowCount();                    //获得“开台列表”中的行数，即已开台数
        if (rowCount > 0) {                                         //已经有开台
❷           String selectedDeskNum = numComboBox.getSelectedItem()
                    .toString();                                    //获得“台号”下拉列表框中选中的台号
            int needSelectedRow = -1;                               //默认选中的台号未开台
            opened: for (int row = 0; row < rowCount; row++) {      //通过循环查看选中的台号是否已经开台
❸               String openedDeskNum = rightTable.getValueAt(row, 1).toString(); //获得已开台的台号
                if (selectedDeskNum.equals(openedDeskNum)) {//查看选中的台号是否已经开台
                    needSelectedRow = row;                          //选中的台号已经开台
                    break opened;                                   //跳出循环
                }
            }
            if (needSelectedRow == -1) {                            //选中的台号尚未开台，即新开台
❹               rightTable.clearSelection();                        //取消选择“开台列表”中的选中行
                leftTableValueV.removeAllElements();                //清空“签单列表”中的所有行
❺               leftTableModel.setDataVector(leftTableValueV, leftTableColumnV);//刷新“签单列表”
            } else {                                                //选中的台号已经开台，即添加菜品
                if (needSelectedRow != rightTable.getSelectedRow()) {
                    //“台号”下拉列表框中选中的台号在“开台列表”中未被选中
                    rightTable.setRowSelectionInterval(needSelectedRow);//在“开台列表”中选中该台号
                    leftTableValueV.removeAllElements();    //清空“签单列表”中的所有行
                    leftTableValueV.addAll(menuOfDeskV
                            .get(needSelectedRow));  //将选中台号的签单列表添加到“签单列表”中
                    leftTableModel.setDataVector(leftTableValueV, leftTableColumnV);//刷新“签单列表”
```

```
                    leftTable.setRowSelectionInterval(0);       //选中“签单列表”中的第一行
                }
            }
        }
    }
});
```

代码贴士

❶ getRowCount()：该方法用于获取表格中拥有记录的行数，返回值为 int 型。

❷ getSelectedItem()：该方法用于获取下拉列表框中被选中的项目对象，返回值为 Object 型。

❸ getValueAt(int row, int column)：该方法用于获取指定单元格的值。需要注意的是，表格的行和列的索引均从 0 开始，返回值为 Object 型。

❹ clearSelection()：该方法用于取消选择当前表格中所有被选中的行。

❺ setDataVector(Vector dataVector, Vector columnIdentifiers)：该方法用于重新设置表格的列名和行数据。需要注意的是，第一个参数为用来封装表格行数据的向量，第二个参数为用来封装表格列名的向量。

（2）开发智能获取点菜功能，通过为文本框添加键盘事件监听器实现。当按下的是 Enter 键时，等同于单击“开单”按钮，执行开单操作，将在后面详细讲解具体操作过程；如果按下的不是 Enter 键，则获取输入的内容，同时判断输入的是商品编号，还是商品助记码，并按指定条件查询所有符合条件的菜品，如果存在符合条件的菜品，则获取第一个符合条件的菜品，并显示菜品名称和单位，否则将菜品名称和单位设置为空。关键代码如下：

例程 20 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
codeTextField.addKeyListener(new KeyAdapter() {
❶   public void keyReleased(KeyEvent e) {                    //通过键盘监听器实现智能获取菜品
❷       if (e.getKeyCode() == KeyEvent.VK_ENTER) {           //按下 Enter 键
            makeOutAnInvoice();                              //开单
        } else {
            String input = codeTextField.getText().trim();   //获得输入内容
            Vector vector = null;                            //符合条件的菜品集合
            if (input.length() > 0) {                        //输入内容不为空
❸               if (codeRadioButton.isSelected()) {          //按助记码查询
                    vector = dao.sMenuByCode(input);         //查询符合条件的菜品
                    if (vector.size() > 0)                   //存在符合条件的菜品
                        vector = (Vector) vector.get(0);     //获得第一个符合条件的菜品
                    else                                     //不存在符合条件的菜品
                        vector = null;
                } else {                                     //按编号查询
                    vector = dao.sMenuById(input);           //查询符合条件的菜品
                    if (vector.size() > 0)                   //存在符合条件的菜品
                        vector = (Vector) vector.get(0);     //获得第一个符合条件的菜品
                    else                                     //不存在符合条件的菜品
                        vector = null;
                }
            }
            if (vector == null) {                            //不存在符合条件的菜品
                nameTextField.setText(null);                 //设置“商品名称”文本框为空
```

```
                unitTextField.setText(null);                          //设置“单位”文本框为空
            } else {                                                  //存在符合条件的菜品
                nameTextField.setText(vector.get(3).toString());      //设置为符合条件的菜品名称
                unitTextField.setText(vector.get(5).toString());      //设置为符合条件的菜品单位
            }
        }
    }
});
```

代码贴士

❶ keyReleased()：当释放键盘中的按键后会触发该方法。

❷ getKeyCode()：该方法将返回一个 int 型值，该 int 型值代表触发此次事件的按键；静态常量 VK_ENTER 代表 Enter 键。

❸ isSelected()：该方法用来查看单选按钮是否处于选中状态，如果被选中则返回 true，否则返回 false。

下面的代码将实现一个智能化的用来填写数量的文本框，默认数量为 1，当文本框获得焦点时，自动将文本框设置为空；当文本框失去焦点时，将查看文本框是否输入了内容，如果未输入内容，则仍设置为默认数量 1。关键代码如下：

例程 21　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
amountTextField = new JTextField();                                   //创建“数量”文本框
amountTextField.addFocusListener(new FocusListener() {
❶   public void focusGained(FocusEvent e) {                           //当文本框获得焦点时执行
        amountTextField.setText(null);                                //设置“数量”文本框为空
    }
❷   public void focusLost(FocusEvent e) {                            //当文本框失去焦点时执行
        String amount = amountTextField.getText().trim();             //获得输入的数量
        if (amount.length() == 0)                                     //未输入数量
            amountTextField.setText("1");                             //恢复为默认数量 1
    }
});
amountTextField.setText("1");                                         //默认数量为 1
```

代码贴士

❶ focusGained(FocusEvent e)：当被监听的组件对象获得焦点时将触发该方法。

❷ focusLost(FocusEvent e)：当被监听的组件对象失去焦点时将触发该方法。

如果在输入商品编号或助记码时按下 Enter 键，或者单击“开单”按钮，将执行开台点菜操作。如果是新开台点菜，则需要先处理开台信息，即在“开台列表”中添加新开台的信息，然后再处理点菜信息，即在“签单列表”中添加新点菜的信息；如果是为已开台添加菜品，则直接处理点菜信息。关键代码如下：

例程 22　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
private void makeOutAnInvoice() {
    String deskNum = numComboBox.getSelectedItem().toString();        //获得台号
    String menuName = nameTextField.getText();                        //获得商品名称
    String menuAmount = amountTextField.getText();                    //获得数量
    if (deskNum.equals("请选择")) {                                   //验证是否已经选择台号
```

```
            JOptionPane.showMessageDialog(null, "请选择台号！", "友情提示",
                    JOptionPane.INFORMATION_MESSAGE);
            return;
        }
        if (menuName.length() == 0) {                                    //验证是否已经确定商品
            JOptionPane.showMessageDialog(null, "请录入商品名称！", "友情提示",
                    JOptionPane.INFORMATION_MESSAGE);
            return;
        }
        if (!Validate.execute("[1-9]{1}([0-9]{0,1})", menuAmount)) {     //验证数量是否有效，数量必须在 1~99 之间
            String info[] = new String[] { "您输入的数量错误！", "数量必须在 1-99 之间！" };
            JOptionPane.showMessageDialog(null, info, "友情提示", JOptionPane.INFORMATION_MESSAGE);
            return;
        }
        //处理开台信息
        int rightSelectedRow = rightTable.getSelectedRow();              //获得被选中的台号
        int leftRowCount = 0;                                            //默认点菜数量为 0
        if (rightSelectedRow == -1) {                                    //没有被选中的台号，即新开台
            rightSelectedRow = rightTable.getRowCount();                 //被选中的台号为新开的台
            Vector deskV = new Vector();                                 //创建一个代表新开台的向量对象
            deskV.add(rightSelectedRow + 1);                             //添加开台序号
            deskV.add(deskNum);                                          //添加开台号
            deskV.add(Today.getTime());                                  //添加开台时间
            rightTableModel.addRow(deskV);                               //将开台信息添加到“开台列表”中
            rightTable.setRowSelectionInterval(rightSelectedRow);        //选中新开的台
            menuOfDeskV.add(new Vector());                               //添加一个对应的签单列表
        } else {                                                         //选中的台号已经开台，即添加菜品
            leftRowCount = leftTable.getRowCount();                      //获得已点菜的数量
        }
        //处理点菜信息
        Vector vector = dao.sMenuByName(menuName);                       //获得被点菜品
        int amount = Integer.valueOf(menuAmount);                        //将菜品数量转为 int 型
        int unitPrice = Integer.valueOf(vector.get(5).toString());       //将菜品单价转为 int 型
        int money = unitPrice * amount;                                  //计算菜品消费额
        Vector<Object> menuV = new Vector<Object>();                     //创建一个代表新点菜的向量对象
        menuV.add("NEW");                                                //添加新点菜标记
        menuV.add(leftRowCount + 1);                                     //添加点菜序号
        menuV.add(vector.get(0));                                        //添加菜品编号
        menuV.add(menuName);                                             //添加菜品名称
        menuV.add(vector.get(4));                                        //添加菜品单位
        menuV.add(amount);                                               //添加菜品数量
        menuV.add(unitPrice);                                            //添加菜品单价
        menuV.add(money);                                                /添加菜品消费额
        leftTableModel.addRow(menuV);                                    //将点菜信息添加到“签单列表”中
        leftTable.setRowSelectionInterval(leftRowCount);                 //将新点菜设置为选中行
        menuOfDeskV.get(rightSelectedRow).add(menuV);                    //将新点菜信息添加到对应的签单列表
    }
```

在新添加菜品的前方会有一个 NEW 标记，确定点菜结束后单击“签单”按钮，将取消所有新添加菜品前方的 NEW 标记。在未取消 NEW 标记的情况下可以选中后单击“取消”按钮取消该菜品，如

果该餐台只点了该菜品，取消该菜品后将同时取消该餐台的开台信息；如果该餐台已经点了其他菜品，并且取消的不是最后点的菜品，还需要修改所点菜品的序号。关键代码如下：

例程 23　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
String NEW = leftTable.getValueAt(lSelectedRow, 0).toString();//获得选中菜品的新点菜标记
if (NEW.equals("")) {                                          //没有新点菜标记，不允许取消
    JOptionPane.showMessageDialog(null, "很抱歉，该商品已经不能取消！",
            "友情提示", JOptionPane.INFORMATION_MESSAGE);
    return;
} else {
    int rSelectedRow = rightTable.getSelectedRow();     //获得“开台列表”中的选中行，即取消菜品的台号
    int i = JOptionPane.showConfirmDialog(null, "确定要取消“" + rightTable.getValueAt(rSelectedRow, 1)
            + "”中的商品“" + leftTable.getValueAt(lSelectedRow, 3) + "”？",
            "友情提示", JOptionPane.YES_NO_OPTION);  //弹出提示信息确认是否取消
    if (i == 0) {                                              //确认取消
        leftTableModel.removeRow(lSelectedRow);                //从“签单列表”中取消菜品
        int rowCount = leftTable.getRowCount();                //获得取消后的点菜数量
        if (rowCount == 0) {                                   //未点任何菜品
            rightTableModel.removeRow(rSelectedRow); //取消开台
            menuOfDeskV.remove(rSelectedRow);                  //移除签单列表
        } else {
            if (lSelectedRow == rowCount) {                    //取消菜品为最后一个
                lSelectedRow -= 1;                             //设置最后一个菜品为选中的
            } else {                                           //取消菜品不是最后一个
                Vector<Vector<Object>> menus = menuOfDeskV.get(rSelectedRow);
                for (int row = lSelectedRow; row < rowCount; row++) {      //修改点菜序号
                    leftTable.setValueAt(row + 1, row, 1);
                    menus.get(row).set(1, row + 1);
                }
            }
            leftTable.setRowSelectionInterval(lSelectedRow);    //设置选中行
        }
    }
}
```

当用户要求添加菜品时，可以在“台号”下拉列表框中选择要求添加菜品的台号，也可以在“开台列表”中选择要求添加菜品的台号，在这里选择后将同步选中“台号”下拉列表框中的相应台号。关键代码如下：

例程 24　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
rightTable.addMouseListener(new MouseAdapter() {
❶   public void mouseClicked(MouseEvent e) {
        int rSelectedRow = rightTable.getSelectedRow();          //获得“开台列表”中的选中行
        leftTableValueV.removeAllElements();                     //清空“签单列表”中的所有行
        leftTableValueV.addAll(menuOfDeskV.get(rSelectedRow));//将选中台号的签单列表添加到“签单列表”中
        leftTableModel.setDataVector(leftTableValueV, leftTableColumnV);  //刷新“签单列表”
        leftTable.setRowSelectionInterval(0);                    //选中“签单列表”中的第一行
```

```
            //同步选中“台号”下拉菜单中的相应台号
❷           numComboBox.setSelectedItem(rightTable.getValueAt(rSelectedRow, 1));
        }
});
```

代码贴士

❶ mouseClicked(MouseEvent e)：当鼠标按键被按下并且在原地释放时触发该方法；假设为 JTable 表格添加该监听器，如果是在第一行上方按下鼠标按键，但是并不立即释放，而是等移动到其他行再释放被按下的鼠标按键，将不会触发该方法。

❷ setSelectedItem(Object selectedItem)：该方法用来设置下拉列表框中被选中的选项。

4.8.4 单元测试

在测试快速获取商品功能时，输入部分助记码后再输入一个空格，在“商品名称”文本框中仍然显示输入空格之前的商品名称，如图 4.27 所示。

图 4.27 输入空格后的效果

这其中有两个小错误，一个错误是输入空格后就不应该显示原来的商品名称，这是因为在获取输入内容时去掉了首尾空格。关键代码如下：

例程 25 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
String input = codeTextField.getText().trim();          //获取输入内容
```

另一个错误是该文本框就不应该允许输入空格，解决了这个错误，上面的小错误也就自然解决了。但是建议将去掉首尾空格的代码去掉，这样对提升软件性能还是有好处的。这也是一个良好的编码习惯，就是应该尽量避免编写没有必要的代码。

如果想令文本框不允许输入空格，可以通过重载 KeyListener 类的 keyTyped(KeyEvent e)方法实现，通过该方法的入口参数 e 的 getKeyChar()方法可以得到此次输入的字符，如果为空格则通过 consume()方法销毁此次事件。关键代码如下：

例程 26 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
public void keyTyped(KeyEvent e) {
    if (e.getKeyChar() == ' ')              //判断用户输入的是否为空格
        e.consume();                        //如果是空格则销毁此次按键事件
}
```

同样，对“数量”文本框也可以采用这种办法控制用户输入的内容，并且可以控制输入数量的最大位数和不允许输入的第一位为 0。关键代码如下：

例程 27 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
amountTextField.addKeyListener(new KeyAdapter() {
    public void keyTyped(KeyEvent e) {
```

```
            int length = amountTextField.getText().length();//获取当前数量的位数
            if (length < 2) {///位数小于两位
                String num = (length == 0 ? "123456789" : "0123456789"); //将允许输入的字符定义成字符串
                if (num.indexOf(e.getKeyChar()) < 0)        //查看按键字符是否包含在允许输入的字符中
                    e.consume();                            //如果不包含在允许输入的字符中则销毁此次按键事件
            } else {
                e.consume();                                //如果不小于数量允许的最大位数则销毁此次按键事件
            }
        }
});
```

4.9　自动结账工作区设计

4.9.1　自动结账工作区功能概述

自动结账工作区有两个主要功能，一个功能是自动计算当前选中餐台的消费金额，例如选中餐台“8001”，如图 4.28 所示，在自动结账工作区将显示该餐台的消费金额，如图 4.29 所示。

另一个功能是在结账时自动计算找零金额。用户输入“实收金额”后单击“结账”按钮，系统将自动计算出需要找零的金额，并弹出一个结账完成的提示框，如图 4.30 所示。

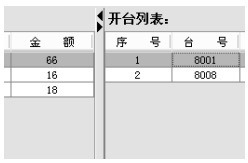


图 4.28　选中的台号为“8001”

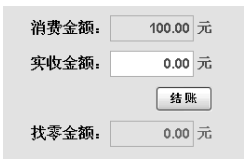


图 4.29　“8001”的消费金额

图 4.30　结账

4.9.2　自动结账工作区技术分析

如果要实现自动计算当前选中餐台消费金额的功能，就要随时监控“签单列表”中内容的变化。例如添加或取消菜品，或者是 “开台列表”中的选中行发生了改变，都将导致“签单列表”中的内容发生改变。如果希望随时监控表格中内容的变化，可通过为表格模型添加 TableModelListener 监听器实现，无论是向表格模型中添加行，还是从表格模型中移除行，以及修改表格中某一单元格的值，都将触发该事件。

4.9.3　自动结账工作区的实现过程

自动结账使用的主要数据表：tb_order_form、tb_order_item

自动结账工作区的开发步骤如下：

（1）实现自动计算当前选中餐台消费金额的功能，即为与“签单列表”对应的表格模型添加一个 TableModelListener 监听器，在监听器中通过循环重新计算该餐台的消费金额，并更新“消费金额”文本框。关键代码如下：

例程 28 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\ TipWizardFrame.java

```
leftTableModel.addTableModelListener(new TableModelListener() {
    public void tableChanged(TableModelEvent e) {          //通过表格模型监听器实现自动结账
        int rowCount = leftTable.getRowCount();            //获得签单列表中的行数
        float expenditure = 0.0f;                          //默认消费 0 元
        for (int row = 0; row < rowCount; row++) {         //通过循环计算消费金额
            expenditure += Float.valueOf(leftTable.getValueAt(row, 7).toString());          //累加消费金额
        }
        expenditureTextField.setText(expenditure + "0");   //更新“消费金额”文本框
    }
});
```

（2）实现结账功能。在结账前首先要判断是否有未签单的菜品，如果有则弹出提示信息，如果没有则获得消费金额和实收金额；然后判断实收金额是否小于消费金额，如果小于同样弹出提示，否则进行结账操作，将消费信息持久化到数据库，并弹出结账完成的提示框；最后将“实收金额”和“找零金额”文本框设置为默认值。关键代码如下：

例程 29 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\frame\TipWizardFrame.java

```
int rowCount = leftTable.getRowCount();                          //获得结账餐台的点菜数量
String NEW = leftTable.getValueAt(rowCount - 1, 0).toString();   //获得最后点菜的标记
if (NEW.equals("NEW")) {                                         //如果最后点菜被标记为 NEW，则弹出提示
    JOptionPane.showMessageDialog(null, "请先确定未签单商品的处理方式！",
            "友情提示", JOptionPane.INFORMATION_MESSAGE);
} else {
    float expenditure = Float.valueOf(expenditureTextField.getText());          //获得消费金额
    float realWages = Float.valueOf(realWagesTextField.getText());              //获得实收金额
    if (realWages < expenditure) {                               //如果实收金额小于消费金额，则弹出提示
        JOptionPane.showMessageDialog(null, "请输入实收金额！",
                "友情提示", JOptionPane.INFORMATION_MESSAGE);
❶       realWagesTextField.requestFocus();                       //并令“实收金额”文本框获得焦点
    } else {
        changeTextField.setText(realWages – expenditure + "0"); //计算并设置“找零金额”
        String[] values = {getNum(), rightTable.getValueAt(selectedRow, 1).toString(),
                Today.getDate() + " " + rightTable.getValueAt(selectedRow, 2),
                expenditureTextField.getText(), user.get(0).toString() };          //组织消费单信息
        dao.iOrderForm(values);                                  //持久化到数据库
        values[0] = dao.sOrderFormOfMaxId();                     //获得消费单编号
        for (int i = 0; i < rowCount; i++) {                     //通过循环获得各个消费项目的信息
            values[1] = leftTable.getValueAt(i, 2).toString();   //获得商品编号
            values[2] = leftTable.getValueAt(i, 5).toString();   //获得商品数量
            values[3] = leftTable.getValueAt(i, 7).toString();   //获得商品消费金额
            dao.iOrderItem(values);                              //持久化到数据库
        }
        JOptionPane.showMessageDialog(null, rightTable
                .getValueAt(selectedRow, 1) + "结账完成！", "友情提示",
                JOptionPane.INFORMATION_MESSAGE);    //弹出结账完成提示
❷       rightTableModel.removeRow(selectedRow);                  //从“开台列表”中取消开台
```

```
            leftTableValueV.removeAllElements();                    //清空“签单列表”
            leftTableModel.setDataVector(leftTableValueV, leftTableColumnV);        //刷新“签单列表”
            realWagesTextField.setText("0.00");                     //清空“实收金额”文本框
            changeTextField.setText("0.00");                        //清空“找零金额”文本框
            menuOfDeskV.remove(selectedRow);                        //从“签单列表”集合中移除已结账的签单列表
        }
    }
```

代码贴士

❶ requestFocus()：该方法用来为调用的组件对象请求获得焦点。

❷ removeRow(int rowIndex)：该方法用来从表格中移除指定行索引所代表的行。

4.10　结账报表工作区设计

4.10.1　结账报表工作区功能概述

本系统提供了 3 种方式的结账报表，分别是日结账报表、月结账报表和年结账报表，在结账报表工作区只提供了打开这 3 种结账报表功能的按钮，如图 4.31 所示。

日结账报表功能提供了对一日营业情况的统计，包括日开台数量、各个餐台的消费金额、菜品的消费情况、各个菜品的日销售情况，以及日营业额等，如图 4.32 所示。

图 4.31　结账报表工作区

编号	台号	开台时间	消费金额	肉片	纯生	可乐	水饺	清蒸鲤鱼
20071224001	8001	12:56:33	85		1	——	1	——
20071224002	8008	12:56:33	428		6	——	2	——
20071224003	8002	12:56:33	28	—	——	2	1	——
20071224004	8003	12:56:33	302		8	——	——	1
20071224005	8006	13:09:46	246	—	6	2	2	——
20071224006	8008	13:09:46	308		2	——	——	1
总计	——	——	1397		23	4	6	2

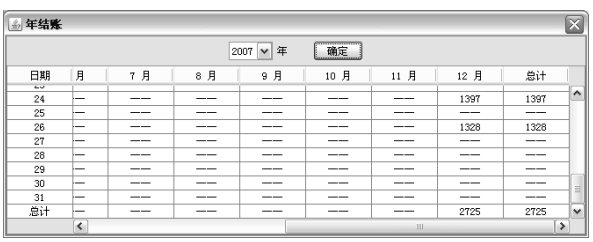
图 4.32　日结账报表

月结账报表功能提供了对一个月营业情况的统计，包括日开台总数、日总营业额、日开台的平均消费额、日开台的最大和最小消费额，以及当月的总开台数、月总营业额，以及一个月中的日平均营业额、一个月中开台的最大和最小消费额，如图 4.33 所示。

年结账报表功能提供了对一年营业情况的统计，包括一年中每天的营业额，每月的营业额、每月同一日期的总营业额，以及一年的营业额，如图 4.34 所示。

日期	开台总数	消费总额	平均消费额	最大消费额	最小消费额
24	6	1397	232	428	28
25	——	——	——	——	——
26	7	1328	189	428	100
27	——	——	——	——	——
28	——	——	——	——	——
29	——	——	——	——	——
30	——	——	——	——	——
31	——	——	——	——	——
总计	13	2725	209	428	28

图 4.33　月结账报表



日期	月	7 月	8 月	9 月	10 月	11 月	12 月	总计
24	—	——	——	——	——	——	1397	1397
25	—	——	——	——	——	——	——	——
26	—	——	——	——	——	——	1328	1328
27	—	——	——	——	——	——	——	——
28	—	——	——	——	——	——	——	——
29	—	——	——	——	——	——	——	——
30	—	——	——	——	——	——	——	——
31	—	——	——	——	——	——	——	——
总计	—	——	——	——	——	——	2725	2725

图 4.34　年结账报表

4.10.2 结账报表工作区技术分析

在实现结账报表功能时，有以下两个技术要点：

- ☑ 对日期有效性的控制。例如，在实现日结账功能时，无论用户修改了统计日期的年度和月份，都要影响到日下拉列表框中的可选项，包括大月（31 天）和小月（30 天）的变化，以及 2 月份在平年（28 天）和闰年（29 天）的变化，如果不能正确处理这些变化，将导致系统无法正常运行。其实在实现月结账报表和年结账报表时也会涉及这个问题，只是在系统界面上不会明显地体会到。解决该问题的大体思路是通过为年度和月份下拉列表框添加事件监听器，实现对日下拉列表框可选项的控制。
- ☑ 对统计表格的控制。当系统界面不能显示出所有统计记录时，只需要将表格放到滚动面板中，这个办法对系统界面不能显示出统计记录的所有行很有效，因为在移动垂直滚动条时，表格的列名并不会随之滚动，即表格的列名是永远可见的；但是当系统界面不能显示出统计记录的所有列时，这个办法就不是很好了，因为在移动水平滚动条时，表格的所有列都会随之滚动，导致最左侧的一列或几列不可见，而表格最左侧的一列或者是几列通常情况下也希望是永远可见的，即不会随着滚动条的移动而滚动。解决该问题的大体思路是实现两个表格，一个表格用来显示最左侧希望永远可见的一列或几列，另一个表格用来显示其他列，然后将两个表格并列显示。

4.10.3 结账报表工作区的实现过程

结账报表使用的主要数据表：tb_order_form、tb_order_item

首先解决在实现结账报表功能时日期的有效性问题，需要定义一个数组，用来存放各个月份拥有的天数，默认 2 月份为 28 天。为了方便使用，将月份与数组的索引一一对应，即不使用数组索引为 1 的位置。关键代码如下：

例程 30 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\check_out\DayDialog.java

```
private int daysOfMonth[] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
```

下面为年度下拉列表框添加事件监听器。首先获得选中的年度，并判断是平年还是闰年，以确定 2 月份的天数，即修改例程 30 中索引为 2 的值，如果为平年则修改为 28，为闰年则修改为 29；然后获得当前选中的月份，如果当前选中的为 2 月份，则继续获得日下拉列表框拥有可选项的数量，如果日下拉列表框拥有可选项的数量不等于例程 30 中索引为 2 的值，当日下拉列表框拥有可选项的数量为 28 时，则为日下拉列表框添加一个可选项“29”，否则从日下拉列表框中移除可选项“29”。关键代码如下：

例程 31 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\check_out\DayDialog.java

```
yearComboBox.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int year = (Integer) yearComboBox.getSelectedItem();      //获得选中的年度
        judgeLeapYear(year);                                      //判断是否为闰年，以确定 2 月份的天数
        int month = (Integer) monthComboBox.getSelectedItem(); //获得选中的月份
```

```
        if (month == 2) {                                          //如果选中的为 2 月
❶           int itemCount = dayComboBox.getItemCount();            //获得日下拉列表框当前的天数
            if (itemCount != daysOfMonth[2]) {                     //如果日下拉列表框当前的天数不等于 2 月份的天数
                if (itemCount == 28)                               //如果日下拉列表框当前的天数为 28 天
❷                   dayComboBox.addItem(29);                       //则添加为 29 天
                else                                               //否则日下拉列表框当前的天数则为 29 天
❸                   dayComboBox.removeItem(29);                    //则减少为 28 天
            }
        }
    }
});
```

代码贴士

❶ getItemCount()：该方法用来获得下拉列表框拥有选项的数量。

❷ addItem(Object item)：该方法用来向下拉列表框中添加指定选项。

❸ removeItem(Object item)：该方法用来从下拉列表框中移除指定选项。

下面为月份下拉列表框添加事件监听器。首先获得选中的月份，并获得日下拉列表框拥有可选项的数量。如果日下拉列表框拥有可选项的数量不等于当前选中月份拥有的天数，当日下拉列表框拥有可选项的数量大于当前选中月份拥有的天数时，则移除日下拉列表框中最大的可选项，并将日下拉列表框拥有可选项的数量减 1；否则将日下拉列表框拥有可选项的数量加 1，并添加到日下拉列表框的可选项中。关键代码如下：

例程 32　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\check_out\DayDialog.java

```
monthComboBox.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int month = (Integer) monthComboBox.getSelectedItem();          //获得选中的月份
        int itemCount = dayComboBox.getItemCount();//获得日下拉列表框当前的天数
        while (itemCount != daysOfMonth[month]) {     //如果日下拉列表框当前的天数不等于选中月份的天数
            if (itemCount > daysOfMonth[month]) {                       //如果大于选中月份的天数
                dayComboBox.removeItem(itemCount);                      //则移除最后一个可选项
                itemCount--;                                            //并将日下拉列表框当前的天数减 1
            } else {                                                    //否则小于选中月份的天数
                itemCount++;                                            //将日下拉列表框当前的天数加 1
                dayComboBox.addItem(itemCount);                         //并添加为可选项
            }
        }
    }
});
```

通过年度和月份下拉列表框的事件监听器对日下拉列表框可选项的控制，无论选择哪一年或哪一月，日下拉列表框提供的日期可选项都是一个有效的日期。

下面解决当系统界面不能显示出统计记录的所有列时，移动滚动条导致最左侧的一列或几列不可见的问题。首先通过实现抽象类 AbstractTableModel，编写一个用来创建固定列表格模型的类 FixedColumnTableModel。关键代码如下：

例程 33 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\mwing\FixedColumnTablePanel.java

```
class FixedColumnTableModel extends AbstractTableModel {
    public int getColumnCount() {                                    //返回固定列的数量
        return fixedColumn;
    }
    public int getRowCount() {                                       //返回行数
        return tableValueV.size();
    }
    public Object getValueAt(int rowIndex, int columnIndex) {        //返回指定单元格的值
        return tableValueV.get(rowIndex).get(columnIndex);
    }
    public String getColumnName(int columnIndex) {                   //返回指定列的名称
        return tableColumnV.get(columnIndex);
    }
}
```

然后再通过实现抽象类 AbstractTableModel，编写一个用来创建移动列表格模型的类 FloatingColumnTableModel。关键代码如下：

例程 34 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\mwing\FixedColumnTablePanel.java

```
class FloatingColumnTableModel extends AbstractTableModel {
    public int getColumnCount() {                                    //返回移动列的数量
        return tableColumnV.size() - fixedColumn;                    //需要扣除固定列的数量
    }
    public int getRowCount() {                                       //返回行数
        return tableValueV.size();
    }
    public Object getValueAt(int rowIndex, int columnIndex) {        //返回指定单元格的值
        return tableValueV.get(rowIndex).get(columnIndex + fixedColumn);//需要为列索引加上固定列的数量
    }
    public String getColumnName(int columnIndex) {                   //返回指定列的名称
        return tableColumnV.get(columnIndex + fixedColumn);          //需要为列索引加上固定列的数量
    }
}
```

注意 在例程 34 中，在处理与表格列有关的信息时，均需要在表格总列数的基础上减去固定列的数量。

下面通过实现接口 ListSelectionListener，编写一个用来同步两个表格中的选中行的事件监听器类 MListSelectionListener，即当选中固定列表格中的某一行时，监听器会同时选中移动列表格中的对应行，同样，当选中移动列表格中的某一行时，监听器会同时选中固定列表格中的对应行。关键代码如下：

例程 35 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\mwing\FixedColumnTablePanel.java

```
class MListSelectionListener implements ListSelectionListener {
    boolean isFixedColumnTable = true;                               //默认由选中固定列表格中的行触发
    public MListSelectionListener(boolean isFixedColumnTable) {
```

```
            this.isFixedColumnTable = isFixedColumnTable;
        }
        public void valueChanged(ListSelectionEvent e) {
            if (isFixedColumnTable) {                                          //由选中固定列表格中的行触发
                int selectedRow = fixedColumnTable.getSelectedRow();           //获得固定列表格中的选中行
                floatingColumnTable.setRowSelectionInterval(selectedRow);//同时选中移动列表格中的选中行
            } else {                                                           //由选中移动列表格中的行触发
                int selectedRow = floatingColumnTable.getSelectedRow();        //获得移动列表格中的选中行
                fixedColumnTable.setRowSelectionInterval(selectedRow);         //同时选中固定列表格中的选中行
            }
        }
}
```

注意　例程 35 实现的事件监听器要求两个表格必须均是单选模式的，即一次只允许选中一行。

最后依次创建固定列表格和移动列表格，并通过这两个表格的选择模型对象，为两个表格添加事件监听器；再创建一个滚动面板对象，并将固定列表格的列头对象添加到滚动面板的左上方；最后创建一个视口对象，先将固定表格对象添加到视口对象中，并将视口的首选大小设置为固定列表格的首选大小，并依次将视口和移动列表格添加到滚动面版的标题视口和默认视口中。关键代码如下：

例程 36　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\mwing\FixedColumnTablePanel.java

```
fixedColumnTableModel = new FixedColumnTableModel();                  //创建固定列表格模型对象
fixedColumnTable = new MTable(fixedColumnTableModel);                 //创建固定列表格对象
ListSelectionModel fixed = fixedColumnTable.getSelectionModel();      //获得选择模型对象
fixed.addListSelectionListener(new MListSelectionListener(true));     //添加行被选中的事件监听器
floatingColumnTableModel = new FloatingColumnTableModel();            //创建移动列表格模型对象
floatingColumnTable = new MTable(floatingColumnTableModel);           //创建移动列表格对象
ListSelectionModel floating = floatingColumnTable.getSelectionModel();//获得选择模型对象
floating.addListSelectionListener(new MListSelectionListener(false)); //添加行被选中的事件监听器
JScrollPane scrollPane = new JScrollPane();                           //创建一个滚动面板对象
scrollPane.setCorner(JScrollPane.UPPER_LEFT_CORNER, fixedColumnTable
        .getTableHeader());                                           //将固定列表格头放到滚动面板的左上方
JViewport viewport = new JViewport();                                 //创建一个用来显示基础信息的视口对象
viewport.setView(fixedColumnTable);                                   //将固定列表格添加到视口中
viewport.setPreferredSize(fixedColumnTable.getPreferredSize());//设置视口的首选大小为固定列表格的首选大小
scrollPane.setRowHeaderView(viewport);                                //将视口添加到滚动面板的标题视口中
scrollPane.setViewportView(floatingColumnTable);                      //将移动列表格添加到默认视口
```

4.10.4　单元测试

在测试年结账报表功能时，当系统界面不能显示出统计记录的所有列时，移动滚动条将最左侧的一列或几列不可见的问题解决了；但是当选中右侧移动列表格的多行时，在左侧固定列表格中只选中了一行，如图 4.35 所示，当选中右侧移动列表格的第 26～30 行时，在左侧固定列表格中只选中了第

26 行。

同样，当选中左侧固定列表格的多行时，在右侧移动列表格中也只选中了一行，如图 4.36 所示，当选中左侧固定列表格的第 26～30 行时，在右侧移动列表格中只选中了第 26 行。

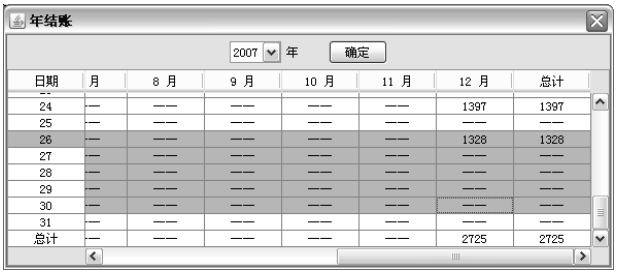


图 4.35　选中移动列表格的多行

图 4.36　选中固定列表格的多行

图 4.35 和图 4.36 所示的两种情况并不是想要的，这是因为在例程 35 中实现事件监听器时，并没有同步选中关联表中的所有对应行，而只是选中了关联表中的第一个对应行，但是表格却不是单选模式的，才导致出现了这种情况。解决的办法是将表格设置为单选模式，因为这两个表格均是通过 FixedColumnTablePanel 类的内部类 MTable 实现的，所以只需要在内部类 MTable 中重构其父类 JTable 的 getSelectionModel()方法。关键代码如下：

例程 37　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\mwing\FixedColumnTablePanel.java

```
public ListSelectionModel getSelectionModel() {
    ListSelectionModel selectionModel = super.getSelectionModel();
    selectionModel.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    return selectionModel;
}
```

4.11　后台管理工作区设计

后台管理工作区用来维护软件正常运行需要的一些信息，例如台号信息、菜系信息、菜品信息，只有填写了这些信息，才能进行开台，以至结账和生成报表。

4.11.1　后台管理工作区功能概述

在后台管理工作区提供了对台号、菜系和菜品信息的维护功能，在添加信息时，一是验证数据的合法性，例如在添加台号信息时，不小心将座位数输入为了 100，在单击“添加”按钮时将弹出座位数输入错误的提示，如图 4.37 所示。再就是查看新添加的信息是否已经存在，例如在添加菜系信息时，输入“炖炒类”后单击“添加”按钮，将弹出菜系已经存在的提示，因为添加同名的菜系是没有意义的，如图 4.38 所示。

在删除信息前，通常情况下建议弹出一个确认提示框，以免由于疏忽错误删除信息，如图 4.39 所示。

图 4.37　餐台座位数输入错误

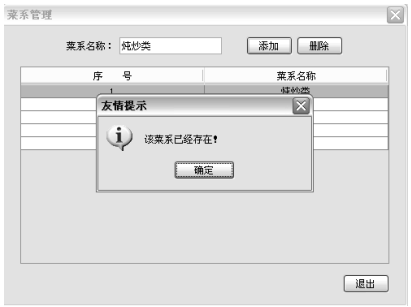


图 4.38　添加的菜系已经存在

图 4.39　删除菜品之前弹出的确认提示框

4.11.2　后台管理工作区技术分析

在对用户输入的数据进行验证时，如果某个数据不符合要求，通常情况下希望对应的组件获得焦点。如果是对用户输入的数据进行逐个验证，这个问题就很好解决了；但是当对用户输入的数据进行批量验证时，就很难判断是哪个组件接收的数据不符合要求了。这个问题可以通过 Java 的反射机制解决，通过 Java 反射可以轻松地实现对组件的内容进行批量验证，并且在接收数据不符合要求的情况下，可以直接令相应的组件获得焦点，另外通过为组件设置名称，还可以弹出一个很有针对性的提示。

例如，在实现添加菜品功能时，就是通过 Java 的反射机制实现的对 4 个文本框不允许为空的验证，在未输入单价的情况下直接单击“添加”按钮，就会弹出一个很有针对性的提示，如图 4.40 所示，单击提示框中的“确定”按钮后，“单价”文本框将获得焦点，如图 4.41 所示。

图 4.40　有针对性的提示内容

图 4.41　为空的“单价”文本框获得焦点

4.11.3　后台管理工作区的实现过程

后台管理包括对台号、菜系和菜品的管理，下面将依次讲解这 3 个功能的实现过程，以及一些典型的技巧。

1．实现台号管理功能

（1）实现添加台号的功能。在执行添加台号操作时，首先要判断台号和座位数是否有效，台号最多为 5 个字符，座位数不能大于 99 个，并且座位号允许重复；然后创建一个向量对象，用来封装新添加台号的信息，并添加到表格中；最后将新添加的台号信息保存到数据库中。关键代码如下：

例程 38 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\manage\DeskNumDialog.java

```
final JButton addButton = new JButton();                                    //创建添加台号按钮对象
addButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String num = numTextField.getText().trim();                         //获取台号，并去掉首尾空格
        String seating = seatingTextField.getText().trim();                 //获取座位数，并去掉首尾空格
        if (num.equals("") || seating.equals("")) {                         //查看是否输入了台号和座位数
            JOptionPane.showMessageDialog(null, "请输入台号和座位数！", "友情提示",
                    JOptionPane.INFORMATION_MESSAGE);
            return;
        }
        if (num.length() > 5) {                                             //查看台号的长度是否超过了 5 位
            JOptionPane.showMessageDialog(null, “台号最多只能为 5 个字符！”, “友情提示”,
                    JOptionPane.INFORMATION_MESSAGE);
            numTextField.requestFocus();                                    //为“台号”文本框请求获得焦点
            return;
        }
        if (!Validate.execute("[1-9]{1}([0-9]{0,1})", seating)) {  //验证座位数是否在 1～99 之间
            String[] infos = { "座位数输入错误！", "座位数必须在  1～99  之间！" };
            JOptionPane.showMessageDialog(null, infos, “友情提示”, JOptionPane.INFORMATION_
MESSAGE);
            seatingTextField.requestFocus();                                //为“座位数”文本框请求获得焦点
            return;
        }
        if (dao.sDeskByNum(num) != null) {                                  //查看该台号是否已经存在
            JOptionPane.showMessageDialog(null, “该台号已经存在！”, “友情提示”,
                    JOptionPane.INFORMATION_MESSAGE);
            numTextField.requestFocus();                                    //为“台号”文本框请求获得焦点
            return;
        }
        int row = table.getRowCount();                                      //获得当前拥有台号的个数
        Vector newDeskNumV = new Vector();                                  //创建一个代表新台号的向量
        newDeskNumV.add(new Integer(row + 1));                              //添加序号
        newDeskNumV.add(num);                                               //添加台号
        newDeskNumV.add(seating);                                           //添加座位数
        tableModel.addRow(newDeskNumV);                                     //将新台号信息添加到表格中
        table.setRowSelectionInterval(row, row);                            //设置新添加的台号为选中的
        numTextField.setText(null);                                         //将“台号”文本框设置为空
        seatingTextField.setText(null);                                     //将“座位”数文本框设置为空
        dao.iDesk(num, seating);                                            //将新添加的台号信息保存到数据库中
        JDBC.closeConnection();                                             //关闭数据库连接
    }
});
addButton.setText("添加");
```

（2）实现删除台号的功能。在执行删除台号操作前，首先要判断是否选中了要删除的台号；然后弹出提示确认是否真的删除，如果真的要删除，还要判断该餐台是否正在被使用；最后执行删除操作。关键代码如下：

例程 39　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\manage\DeskNumDialog.java

```
final JButton delButton = new JButton();                                    //创建删除台号按钮对象
delButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int selectedRow = table.getSelectedRow();                           //获得选中的餐台
        if (selectedRow == -1) {                                            //未选中任何餐台
            JOptionPane.showMessageDialog(null, "请选择要删除的餐台！", "友情提示",
                    JOptionPane.INFORMATION_MESSAGE);
        } else {
            String deskNum = table.getValueAt(selectedRow, 1).toString();       //获得选中餐台的编号
            for (int row = 0; row < openedDeskTable.getRowCount(); row++) {//查看该餐台是否正在被使用
                if (deskNum.equals(openedDeskTable.getValueAt(row, 1))) {
                    JOptionPane.showMessageDialog(null, "该餐台正在使用，不能删除！", "友情提示",
                            JOptionPane.INFORMATION_MESSAGE);
                    return;                                                 //该餐台正在使用，不能删除，返回
                }
            }
            String infos[] = new String[] {                                 //组织确认信息
                    "确定要删除餐台：",
                    "　　台　号：" + deskNum,
                    "　　座位数：" + table.getValueAt(selectedRow, 2) .toString() };
            int i = JOptionPane.showConfirmDialog(null, infos, "友情提示",
                    JOptionPane.YES_NO_OPTION);                             //弹出确认提示
            if (i == 0) {                                                   //确认删除
                dao.dDeskByNum(deskNum);                                    //从数据库中删除
                tableModel.setDataVector(dao.sDesk(), columnNameV);//刷新表格
                int rowCount = table.getRowCount();                         //获得删除后拥有的餐台数
                if (rowCount > 0) {                                         //还拥有餐台
                    if (selectedRow == rowCount)                            //删除的为最后一个餐台
                        selectedRow -= 1;                                   //将选中的餐台前移一行
                    table.setRowSelectionInterval(selectedRow, selectedRow);      //设置当前选中的餐台
                }
                JDBC.closeConnection();                                     //关闭数据库连接
            }
        }
    }
});
delButton.setText("删除");
```

2．实现菜系管理功能

（1）实现添加菜系的功能。在执行添加菜系操作时，首先要判断菜系名称的长度是否超出了允许的最大长度，并查看该菜系名称是否已经存在；然后创建一个向量对象，用来封装新添加菜系的信息，并添加到表格中；最后将新添加的菜系信息保存到数据库中。关键代码如下：

例程 40　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\manage\SortDialog.java

```
final JButton addButton = new JButton();                                    //创建添加菜系名称按钮对象
addButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
```

```
            String sortName = sortNameTextField.getText().trim();     //获得菜系名称，并去掉首尾空格
            if (sortName.equals("")) {                                //查看是否输入了菜系名称
                JOptionPane.showMessageDialog(null, "请输入菜系名称！", "友情提示",
                        JOptionPane.INFORMATION_MESSAGE);
                return;
            }
            if (sortName.length() > 10) {                             //查看菜系名称的长度是否超过了 10 个汉字
                JOptionPane.showMessageDialog(null, "菜系名称最多只能为 10 个汉字！",
                        "友情提示", JOptionPane.INFORMATION_MESSAGE);
                return;
            }
            if (dao.sSortByName(sortName).size() > 0) {               //查看该菜系名称是否已经存在
                JOptionPane.showMessageDialog(null, "该菜系已经存在！", "友情提示",
                        JOptionPane.INFORMATION_MESSAGE);
                return;
            }
            int row = tableModel.getRowCount();                       //获得当前拥有菜系名称的个数
            Vector newSortV = new Vector();                           //创建一个代表新菜系名称的向量
            newSortV.add(new Integer(row + 1));                       //添加序号
            newSortV.add(sortName);                                   //添加菜系名称
            tableModel.addRow(newSortV);                              //将新菜系名称信息添加到表格中
            table.setRowSelectionInterval(row, row);                  //设置新添加的菜系名称为选中的
            sortNameTextField.setText(null);                          //将“菜系名称”文本框设置为空
            dao.iSort(sortName);                                      //将新添加的菜系名称信息保存到数据库中
            JDBC.closeConnection();                                   //关闭数据库连接
        }
    });
    addButton.setText("添加");
```

（2）实现删除菜系的功能。在执行删除菜系操作前，首先要判断是否有菜系被选中；然后弹出提示确认是否真的删除；最后执行删除操作，如果删除的是表格中的最后一行，则选中删除表格中的最后一行，如果删除的不是最后一行，则选中删除同一位置的表格行。关键代码如下：

例程 41 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\manage\SortDialog.java

```
final JButton delButton = new JButton();                              //创建删除菜系名称按钮对象
delButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int row = table.getSelectedRow();                             //获得选中的菜系
        String delSortName = (String) table.getValueAt(row, 1);       //获得选中的菜系名称
        int j = JOptionPane.showConfirmDialog(null, "确定要删除菜系“" + delSortName
                + "”？", "友情提示", JOptionPane.YES_NO_OPTION);//弹出确认提示
        if (j == 0) {                                                 //确认删除
            tableModel.removeRow(row);                                //从表格中移除菜系信息
            int rowCount = table.getRowCount();                       //获得删除后拥有的菜系数
            if (rowCount > 0) {                                       //还拥有菜系
                if (row < table.getRowCount()) {                      //删除的不是位于表格最后的菜系
                    for (int i = row; i < table.getRowCount(); i++) {
                        table.setValueAt(i + 1 + "", i, 0);           //修改位于删除菜系之后的序号
```

```
                    }
                    table.setRowSelectionInterval(row, row);  //设置上移到删除行索引的菜系为被选中
                } else {                                      //删除的是位于表格最后的菜系
                    table.setRowSelectionInterval(row - 1, row - 1); //设置当前位于表格最后的菜系被选中
                }
            }
            dao.dSortByName(delSortName);                     //从数据库中删除菜系
            JDBC.closeConnection();                           //关闭数据库连接
        }
    }
});
delButton.setText("删除");
```

3．实现菜品管理功能

（1）实现添加菜品的功能。在执行添加菜品操作时，首先通过 Java 反射机制验证 4 个文本框是否为空，如果为空则弹出要求填写信息的提示框，并且通过获取组件的标识名称，组织出有针对性的提示信息，还要为空的文本框请求获得焦点，之后再对这些信息进行具体的验证；然后创建一个向量对象，用来封装新添加菜品的信息，并添加到表格中；最后将新添加的菜品信息保存到数据库中。关键代码如下：

例程 42　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\manage\MenuDialog.java

```
final JButton addButton = new JButton();
addButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
❶       Field[] fields = MenuDialog.class.getDeclaredFields();//通过 Java 反射获取 MenuDialog 类的所有属性
        for (int i = 0; i < fields.length; i++) {
            Field field = fields[i];                          //获得指定属性
❷           if (field.getType().equals(JTextField.class)) {   //只验证 JtextField 类型的属性
❸               field.setAccessible(true);                    //私有属性必须设置为 true 才允许访问
                JTextField textField = null;                  //声明一个 JTextField 类型的对象
                try {
❹                   textField = (JTextField) field.get(MenuDialog.this);      //获得本类中的相应对象
                } catch (Exception exception) {
                    exception.printStackTrace();
                }
                if (textField.getText().trim().equals("")) {  //文本框为空
                    JOptionPane.showMessageDialog(null, "请填写商品“"
❺                       + textField.getName() + "”！", "友情提示",
                        JOptionPane.INFORMATION_MESSAGE);     //弹出需要输入信息的提示
❻                   textField.requestFocus();                 //令文本框获得焦点
                    return;                                   //返回
                }
            }
        }
        if (sortComboBox.getSelectedIndex() == 0) {           //单独验证下拉列表框
            JOptionPane.showMessageDialog(null, "请选择商品所属“菜系”！", "友情提示",
                    JOptionPane.INFORMATION_MESSAGE);
```

```
        return;
    }
    String menu[] = new String[6];                                  //创建一个数组，用来保存菜品信息
    menu[0] = numTextField.getText().trim();                        //获得菜品编号
    menu[1] = nameTextField.getText().trim();                       //获得菜品名称
    menu[2] = codeTextField.getText().trim();                       //获得菜品助记码
    menu[3] = sortComboBox.getSelectedItem().toString();            //获得菜品所属菜系
    menu[4] = unitTextField.getText().trim();                       //获得菜品单位
    menu[5] = unitPriceTextField.getText().trim();                  //获得菜品单价
    if (menu[1].length() > 10) {
        JOptionPane.showMessageDialog(null, "菜品名称最多只能为 10 个汉字！",
                "友情提示", JOptionPane.INFORMATION_MESSAGE);
        nameTextField.requestFocus();
        return;
    }
    if (menu[2].length() > 10) {
        JOptionPane.showMessageDialog(null, "助记码最多只能为 10 个字符！",
                "友情提示", JOptionPane.INFORMATION_MESSAGE);
        codeTextField.requestFocus();
        return;
    }
    if (menu[4].length() > 2) {
        JOptionPane.showMessageDialog(null, "单位最多只能为 2 个汉字！",
                "友情提示", JOptionPane.INFORMATION_MESSAGE);
        unitTextField.requestFocus();
        return;
    }
    if (!Validate.execute("[1-9]{1}[0-9]{0,3}", menu[5])) {
        String infos[] = { "单价输入错误！", "单价必须在 1~9999" };
        JOptionPane.showMessageDialog(null, infos, "友情提示", JOptionPane.INFORMATION_MESSAGE);
        unitPriceTextField.requestFocus();
        return;
    }
    if (dao.sMenuByName(menu[1]) != null) {
        JOptionPane.showMessageDialog(null, "该菜品已经存在！", "友情提示",
                JOptionPane.INFORMATION_MESSAGE);
        nameTextField.requestFocus();
        return;
    }
    int row = tableModel.getRowCount();                             //获得当前拥有的菜品数量
    Vector newMenuV = new Vector();
    newMenuV.add(row + 1);                                          //添加序号
    for (int i = 0; i < menu.length; i++) {
        newMenuV.add(menu[i]);                                      //添加菜品信息
    }
    tableModel.addRow(newMenuV);                                    //将新菜品添加到表格中
    table.setRowSelectionInterval(row, row);                        //选中新添加的菜品
    Vector sortVector = (Vector) dao.sSortByName(menu[3]).get(0);           //获得所属菜系
    menu[3] = sortVector.get(1).toString();                         //设置菜系主键
```

```
            dao.iMenu(menu);                                    //将新菜品信息保存到数据库
            JDBC.closeConnection();                             //关闭数据库连接
        }
});
addButton.setText("添加");
```

代码贴士

❶ getDeclaredFields(): 该方法返回一个 Field 型数组，在数组中包含调用类的所有属性，包括公共、保护、默认（包）访问和私有字段，但不包括继承的字段。

❷ getType(): 该方法返回一个 Class 对象，它标识了此属性的类型，如果只想对类型为 JTextField 的属性进行操作，可以通过代码“getType().equals(JTextField.class)”查看属性的类型是否为 JTextField。

❸ setAccessible(boolean accessible): 在默认情况下通过 Java 反射是不允许访问私有属性的，如果通过该方法将 accessible 属性设置为 true，则允许访问私有属性。

❹ get(Object obj): 该方法用来返回指定对象上此 Field 表示的字段的值。如果该值为基本类型，将自动将其包装为封装类型。MenuDialog.this 代表获得本类中此 Field 表示的字段的值。

❺ getName(): 该方法用来获得组件的名称，主要是为增强软件的人性化特点而设计的。例如这里就利用这个人性化的设计，很方便地弹出了准确的提示信息。

❻ requestFocus(): 该方法用来为组件请求获得焦点，主要是为增强软件的人性化特点而设计的。例如这里就利用这个人性化的设计，很方便地令不符合条件的文本框获得焦点。

（2）实现删除菜品的功能。在执行删除菜品操作前，首先要判断是否存在被选中的菜品；然后弹出提示确认是否真的删除；最后，如果删除的不是表格的最后一行，还要修改其后未删除菜品的序号。关键代码如下：

例程 43　代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\manage\MenuDialog.java

```
final JButton delButton = new JButton();
delButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
❶       int row = table.getSelectedRow();                       //获得选中的菜品
        String delMenuName = table.getValueAt(row, 2).toString();
        String info = "确定要删除菜品“" + delMenuName + "”？";
        int j = JOptionPane.showConfirmDialog(null, info, "友情提示",
                JOptionPane.YES_NO_OPTION);                     //弹出确认提示框
        if (j == 0) {                                           //确认删除
❷           tableModel.removeRow(row);                          //从表格中移除菜品信息
            int rowCount = table.getRowCount();                 //获得删除后拥有的菜品数
            if (rowCount > 0) {                                 //还拥有菜品
                if (row < table.getRowCount()) {                //删除的不是位于表格最后的菜系
                    for (int i = row; i < table.getRowCount(); i++) {
                        table.setValueAt(i + 1 + "", i, 0);     //修改位于删除菜系之后的序号
                    }
                    table.setRowSelectionInterval(row, row);  //设置上移到删除行索引的菜系为被选中
                } else {
                    table.setRowSelectionInterval(row - 1, row - 1); //设置当前位于表格最后的菜系被选中
                }
            }
```

```
                dao.dMenuByName(delMenuName);                    //从数据库中删除菜品
                JDBC.closeConnection();                          //关闭数据库连接
            }
        }
});
delButton.setText("删除");
```

代码贴士

❶ getSelectedRow()：该方法用来获得表格中选中行的索引，返回值为 int 型。如果没有选中行，则返回-1；如果有多行被选中，则返回所有选中行中索引值最小的索引。

❷ removeRow(int rowIndex)：该方法用来从表格中移除位于指定索引位置的行，表格的行索引从 0 开始。

4.11.4 单元测试

当测试菜品管理功能时，在不填写任何菜品信息的情况下直接单击“添加”按钮，弹出如图 4.42 所示的提示信息。

图 4.42 通过 Java 反射验证是否为空弹出的错误提示

这是因为没有为文本框组件设置标识名称，默认情况下文本框组件的标识名称为空，所以在通过文本框组件的 getName()方法获得文本框组件的标识名称时才是“null”，解决的办法是通过 setName(String name)方法为文本框组件设置标识名称。关键代码如下：

例程 44 代码位置：光盘\mr\04\DrinkeryManage\src\com\mwq\manage\MenuDialog.java

```
nameTextField = new JTextField();                    //创建“名称”文本框对象
nameTextField.setName("名称");                       //为“名称”文本框设置标识名称
codeTextField = new JTextField();                    //创建“助记码”文本框对象
codeTextField.setName("助记码");                     //为“助记码”文本框设置标识名称
unitTextField = new JTextField();                    //创建“单位”文本框对象
unitTextField.setName("单位");                       //为“单位”文本框设置标识名称
unitPriceTextField = new JTextField();               //创建“单价”文本框对象
unitPriceTextField.setName("单价");                  //为“单价”文本框设置标识名称
```

4.12 开发技巧与难点分析

作为一个软件开发人员，在设计开发应用软件的过程中，要时刻从软件使用者的角度出发，力求开发出一款功能实用、界面简单、操作人性化并且智能化的产品，只有这样的产品，才更容易被用户接受。

笔者在开发本系统的过程中，就是严格按照这些要求实施的。主要有以下几点：

1．通过输入少量内容就可以快速获取相关产品

通过 KeyListener 监听器，可以很方便地捕获各种键盘事件。KeyListener 监听器通过 3 个接口方法捕获 3 种不同类型的键盘事件：keyPressed(KeyEvent e)方法用来捕获键盘中的某个按键被按下的事件，当某个按键被按下时，将执行该方法；keyReleased(KeyEvent e)方法用来捕获键盘中的某个按键被释放的事件，当某个按键被释放时，将执行该方法；keyTyped(KeyEvent e)方法用来捕获键入键盘中的某个键的事件，当键入某个键时，将执行该方法。

“按下键”和“释放键”事件是低级别事件，它们依赖于平台和键盘布局。只要按下或释放按键就会生成这些事件，这些事件是获取不生成字符输入的键（如动作键、组合键等）的唯一方式。通过 KeyEvent 类的 getKeyCode()方法可以获取代表按下或释放键的虚拟键码，虚拟键码用于报告按下了键盘上的哪个键，而不是通过一个或多个击键组合所生成的字符（如 A 是由 Shift＋a 生成的）。

“键入键”事件是高级别事件，通常情况下不依赖于平台或键盘布局。在输入 Unicode 字符时会生成此类事件，不生成 Unicode 字符的键是不会生成键入键事件的（如动作键、组合键等）。最简单的情况是按下单个键（如 a）将产生键入键事件，但是经常是通过一系列按键（如 Shift＋a）来产生字符，并且按下键事件和键入键事件的映射关系可能是多对一或多对多的。键释放时通常情况下不需要生成一个键入键事件，但是在某些情况下，只有释放某个键才会生成键入键事件（如在 Windows 中通过 Alt-Numpad 方法来输入 ASCII 序列）。

利用组件的 addKeyListener(KeyListener keyListener)方法可以将该监听器对象注册到组件中，这样在按下、释放或键入键生成键盘事件时，将调用监听器对象中的相关方法，并传递过来一个 KeyEvent 对象。

2．人性化控制商品数量 focusLost(FocusEvent e)

通过 FocusListener 监听器，可以很方便地捕获关于焦点的事件。FocusListener 监听器通过两个接口方法捕获两种不同类型的焦点事件：focusGained(FocusEvent e)方法用来捕获组件获得焦点的事件，当其监听的组件获得焦点时，将执行该方法；focusLost(FocusEvent e)方法用来捕获组件失去焦点的事件，当其监听的组件失去焦点时，将执行该方法。

焦点事件分为两个级别，分别是持久性的和暂时性的。如果焦点直接从一个组件移动到另一个组件，如通过调用 requestFocus()方法，或者用户通过 Tab 键遍历组件时，则为持久性焦点更改事件。如果是由于另一个操作间接引起的组件暂时失去焦点，如释放窗口或拖动滚动条，则为暂时性焦点更改事件。在这种情况下，一旦该操作结束，将自动恢复到原始焦点状态。对于释放窗口的情况，只要重新激活窗口就能恢复到原始焦点状态。持久性焦点事件和暂时性焦点事件通过 FOCUS_GAINED 和 FOCUS_LOST 区分，可以通过 isTemporary()方法判断事件的级别。

利用组件的 addFocusListener(FocusListener focusListener)方法可以将该监听器对象注册到组件中，这样在组件获得或失去焦点时，将调用监听器对象中的相关方法，并传递过来一个 FocusEvent 对象。

3．系统自动结账

通过 TableModelListener 监听器，可以很方便地捕获由表格模型发生变化产生的事件，包括向表格模型中添加行、从表格模型中移除行，以及修改某一单元格的值。TableModelListener 监听器只提供了一个接口方法 tableChanged(TableModelEvent e)。通过 TableModelEvent 对象可以判断触发此次事件的

具体原因，例如通过 getType()方法的返回值判断是由向表格模型中添加行触发的，还是由从表格模型中移除行或修改某一单元格的值触发的，当返回值为静态常量 INSERT 时，说明是由向表格模型中添加行触发的；当返回值为静态常量 UPDATE 时，说明是由从表格模型中移除行触发的；当返回值为静态常量 DELETE 时，则是由修改某一单元格的值触发的。

利用组件的 addTableModelListener (TableModelListener tableModelListener)方法可以将该监听器对象注册到组件中，这样在表格模型发生变化时，将调用监听器对象中的 tableChanged(TableModelEvent e)方法，并传递过来一个 TableModelEvent 对象。

4.13　使用 Visio 逆向生成数据库 E-R 图

利用 Visio 也可以根据现有数据库生成数据库 E-R 图，具体步骤如下：

（1）在 Visio 中新建一个“数据库”/“数据库模型图”，如图 4.43 所示。

（2）新建数据库模型图后，在菜单栏中将添加一个“数据库”菜单，选择“数据库”/“反向工程”命令，将打开如图 4.44 所示的对话框。

（3）图 4.44 所示的对话框用来选择数据库连接，如果没有相应的连接，则单击右侧的“新建”按钮建立相应的连接，这里是为本章数据库建立连接，如图 4.45 和图 4.46 所示。

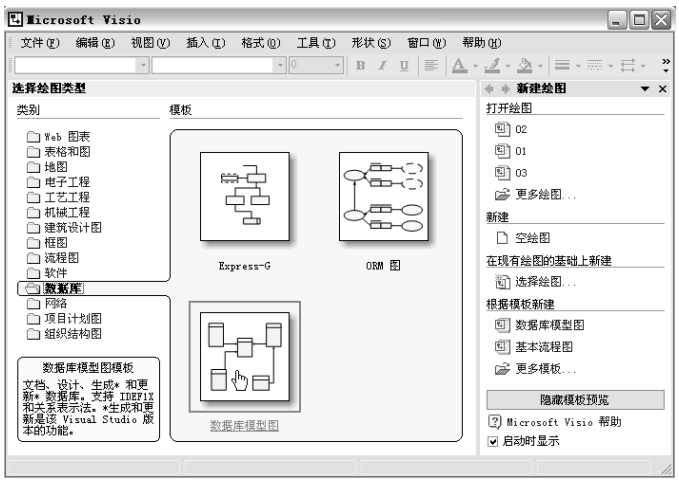


图 4.43　新建数据库模型图

图 4.44　选择数据库连接

图 4.45　设置数据库连接信息

（4）成功连接数据库后，将打开如图 4.47 所示的对话框，从中选择执行反向工程的表和视图。

（5）在图 4.48 所示的对话框中可以选择是否自动添加反向工程项目的形状到当前页，这里选中“是，将形状添加到当前页”单选按钮，然后单击“完成”按钮，将生成如图 4.49 所示的数据库 E-R 图。

图 4.46　选择本章使用的数据库

图 4.47　选择表和视图

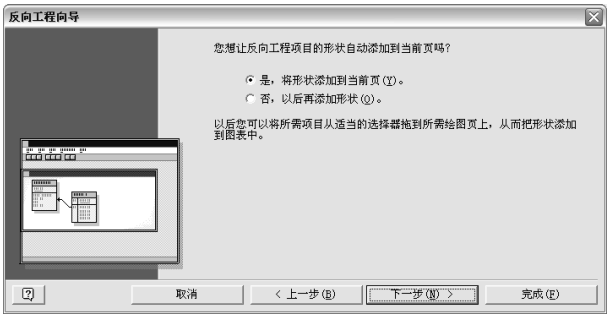


图 4.48　选择是否自动添加反向工程项目的形状到当前页

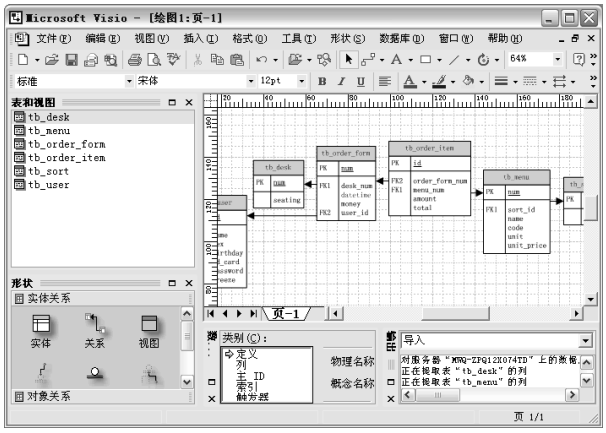


图 4.49　生成的数据库 E-R 图

4.14　本 章 小 结

本章通过一个典型酒店管理系统，既为读者展示了酒店管理系统的业务流程，又为读者展示了酒店管理系统的基本开发思路和实施方法。通过对本章的学习，读者可以了解到 Java 应用程序的开发流程；酒店管理系统的业务流程和软件结构；快速获取商品的实现方法；人性化控制商品数量的实现方法；系统自动结账的实现方法；另外，在本系统中还提供了典型的结账报表功能。

第 5 章

图书馆管理系统

（Swing+SQL Server 2000 实现）

进入 21 世纪以来，信息技术从根本上推动了图书馆的飞速发展，计算机和计算机管理系统已成为图书馆进行图书管理的主要设备和系统。虽然目前很多大型的图书馆已经有一整套比较完善的管理系统，但是在一些中小型的图书馆中，大部分工作仍需手工完成，工作起来效率比较低，不便于动态、及时地调整图书结构。为了更好地适应当前图书馆的管理需求，解决手工管理中存在的弊端，越来越多的中小型图书馆正在逐步向计算机信息化管理转变。图书馆管理系统将先进的信息技术运用到图书馆管理和服务中，从而改变了图书馆的传统管理模式。

通过阅读本章，可以学习到：

- 掌握图书馆管理系统的开发过程
- 掌握使用 PowerDesigner 建模
- 掌握在系统开发中实现 Action 接口
- 掌握如何在菜单栏中添加图标
- 掌握如何使用格式化文本框

5.1　开 发 背 景

××高校拥有一个小型图书馆，为全校师生提供一个阅读、学习的空间。近年来，随着生源不断扩大，图书馆的规模也随之扩大，图书数量也相应地大量增加，有关图书的各种信息成倍增加。面对如此庞大的信息量，校领导决定使用一套合理、有效、规范、实用的图书馆管理系统，对校内图书资料进行统一、集中的管理。

笔者受该高校的委托，开发一个图书馆管理系统，开发宗旨是实现图书管理的系统化、规范化和自动化，达成图书资料集中、统一管理的目标。

5.2　需 求 分 析

图书馆管理系统是图书馆管理工作中不可缺少的部分，对于图书馆的管理者和使用者来说都非常重要，但长期以来，人们使用传统的手工方式或性能较低的图书馆管理系统管理图书馆的日常事务，操作流程比较繁琐，效率相当低。而一个成功的图书馆管理系统应提供快速的图书信息检索功能、快捷的图书借阅、归还流程，为管理者与读者提供充足的信息和快捷的数据处理手段。笔者通过对一些典型图书馆管理系统的考察，从读者与图书馆管理员的角度出发，本着以读者借书、还书快捷、方便的原则，要求本系统应具有以下特点：

- ☑ 具有良好的系统性能，友好的用户界面。
- ☑ 较高的处理效率，便于使用和维护。
- ☑ 采用成熟技术开发，使系统具有较高的技术水平和较长的生命周期。
- ☑ 系统尽可能简化图书馆管理员的重复工作，提高工作效率。
- ☑ 简化数据查询、统计难度。

5.3　系 统 设 计

5.3.1　系统目标

根据以上的需求分析以及与用户的沟通，该系统应达到以下目标：

- ☑ 界面设计友好、美观。
- ☑ 数据存储安全、可靠。
- ☑ 信息分类清晰、准确。
- ☑ 强大的查询功能，保证数据查询的灵活性。
- ☑ 操作简单易用、界面清晰大方。
- ☑ 系统安全、稳定。
- ☑ 开发技术先进、功能完备、扩展性强。
- ☑ 占用资源少、对硬件要求低。

☑ 提供灵活、方便的权限设置功能，使整个系统的管理分工明确。

5.3.2 系统功能结构

图书馆管理系统分为 4 大功能模块，分别为基础数据维护、图书借阅管理、新书订购管理和系统维护。本系统各个部分及其包括的具体功能模块如图 5.1 所示。

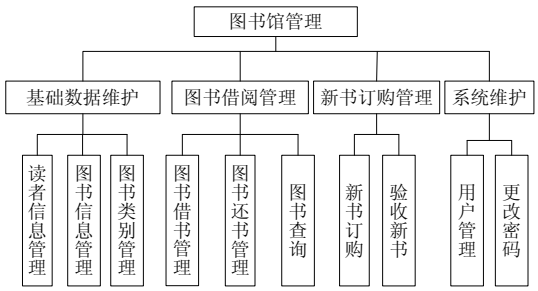


图 5.1 图书馆管理系统功能结构

5.3.3 系统流程图

图书馆管理系统的系统流程如图 5.2 所示。

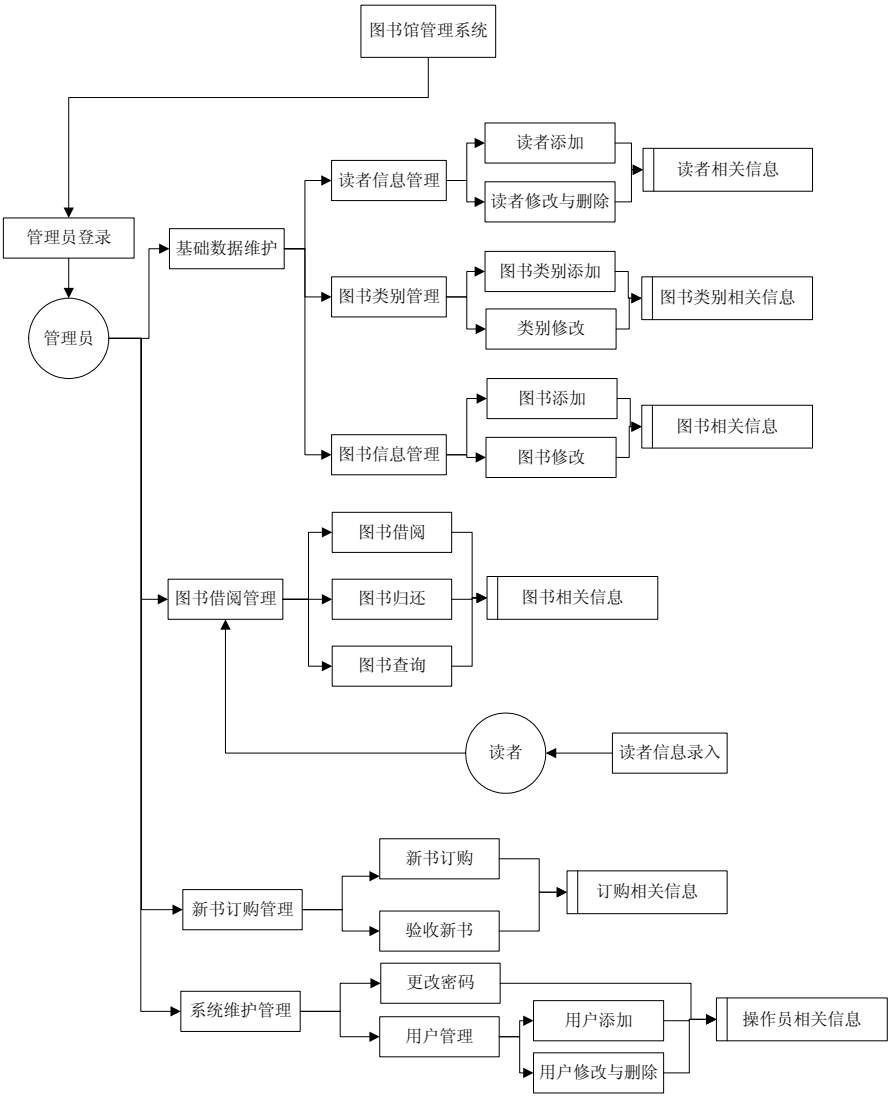


图 5.2 图书馆管理系统流程图

5.3.4　系统预览

图书馆管理系统由多个程序界面组成，下面仅列出几个典型界面，其他界面请读者参见光盘中的源程序。

读者相关信息添加界面如图 5.3 所示，该界面用于将读者相关信息添加至数据表中。读者信息修改与删除界面如图 5.4 所示，该界面用于展示读者相关信息，并且提供了修改与删除功能。

图 5.3　读者相关信息添加界面
（光盘\…\readerAddIFrame.java）

图 5.4　读者信息修改与删除界面
（光盘\…\readerModAndDelIFrame.java）

新书订购管理界面如图 5.5 所示，主要实现新书订购功能。图书验收界面如图 5.6 所示，主要实现新书验收功能。

图 5.5　新书订购管理界面
（光盘\…\newBookOrderIFrame.java）

图 5.6　图书验收界面（光盘\…\newBookCheckIFrame.java）

说明　由于路径太长，因此省略了部分路径，省略的路径是“TM\05\library Manager\src\com\wsy\iframe”。

5.3.5　构建开发环境

在开发图书馆管理系统时，需要具备下面的开发环境。

☑　操作系统：Windows 2003 以上。

☑ Java 开发包：JDK 6.0 以上。
☑ 数据库：SQL Server 2000。
☑ 开发工具：Eclipse 3.2 以上。

5.3.6 文件夹组织结构

在编写代码之前，可以将系统中可能用到的文件夹先创建出来，这样不但方便以后的开发工作，也可以规范系统的整体架构。笔者在开发图书馆管理系统时，设计了如图 5.7 所示的文件夹架构图。在开发时将所创建的文件保存在相应的文件夹中即可。

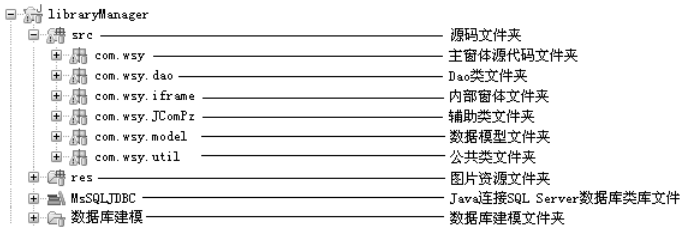


图 5.7 图书馆管理系统文件夹组织结构

5.4 数据库设计

5.4.1 数据库分析

SQL Server 2000 具有很强的完整性与可伸缩性，具有较低的价格比与性能比，考虑到本系统的稳定性与可靠性以及开发程序与用户需求，笔者决定在设计该系统时选择 SQL Server 2000 数据库来满足系统的需求。

5.4.2 数据库概念设计

根据以上对系统所作的需求分析、系统设计，规划出本系统中使用的数据库实体分别为图书信息实体、图书分类实体、图书订购实体、读者信息实体、操作员信息实体、图书借阅信息实体、库存信息实体。其中图书信息实体与图书订购实体、图书分类实体、图书订购实体、图书借阅信息实体、库存信息实体都具有关系，而读者信息实体与图书借阅信息实体同样具有关系。下面将介绍几个关键实体的 E-R 图。

☑ 图书信息实体

图书信息实体包括图书编号、图书类别编号、书名、作者、译者、出版社、价格、出版时间等属性。其中图书编号为图书信息实体的主键，图书类别编号为图书信息实体的外键，与图书类别实体具有外键关系。图书信息实体的 E-R 图如图 5.8 所示。

☑ 读者信息实体

读者信息实体包括条形码、姓名、性别、年龄、电话、押金、生日、职业、证件类型、办证日期、最大借书数量、证件号码等属性。条形码作为本实体的唯一标识。其中，在性别属性标识信息中，“1”代表此读者为男性，“2”代表此读者为女性；最大借书数量属性设置默认值为 3；而在证件属性标识

信息中，“0”代表身份证，“1”代表军人证，“2”代表学生证，“3”代表工作证。读者信息实体的 E-R 图如图 5.9 所示。

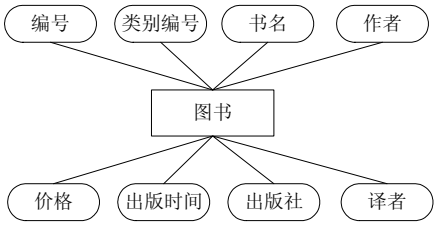


图 5.8　图书信息实体 E-R 图

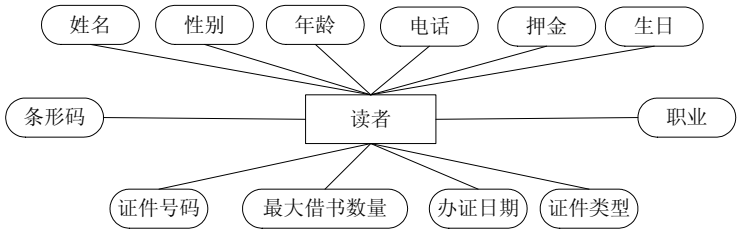


图 5.9　读者信息实体 E-R 图

☑　图书借阅信息实体

图书借阅信息实体包括编号、图书编号、读者编号、操作员编号、是否归还、借阅日期、归还日期等属性。编号作为图书借阅信息实体的唯一标识，它包括两个外键，分别为图书编号与读者编号，图书借阅信息实体以这两个外键与图书信息实体、读者信息实体建立了关系。图书借阅信息实体的 E-R 图如图 5.10 所示。

☑　图书分类实体

图书分类实体包括编号、类别名称等属性。图书分类实体与图书信息实体以图书类别编号建立了关系。图书分类实体的 E-R 图如图 5.11 所示。

☑　图书订购实体

图书订购实体主要包括图书编号、订购日期、订购数量、操作员、是否验收和折扣等属性。图书订购实体以图书编号与图书信息实体建立了关系。图书订购实体的 E-R 图如图 5.12 所示。

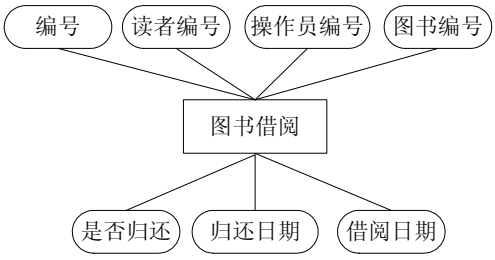


图 5.10　图书借阅信息实体 E-R 图

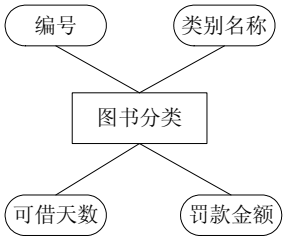


图 5.11　图书分类实体 E-R 图

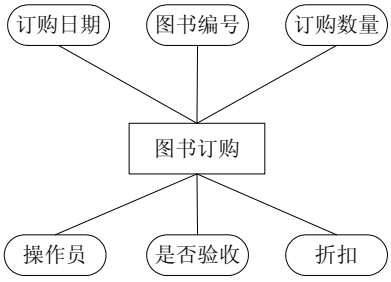


图 5.12　图书订购实体 E-R 图

☑　操作员信息实体

操作员信息实体主要包括编号、姓名、性别、年龄、身份证号、工作日期、电话、是否为管理员和密码等属性。其中，性别属性信息中，“1”代表男性，“2”代表女性；是否为管理员属性信息中，“0”代表当前用户不是管理员，“1”代表当前用户是管理员。操作员信息实体的 E-R 图如图 5.13 所示。

☑　库存信息实体

库存信息实体主要包括编号、库存数量等属性。库存信息实体以库存编号与图书信息实体建立了关系。库存信息实体的 E-R 图如图 5.14 所示。

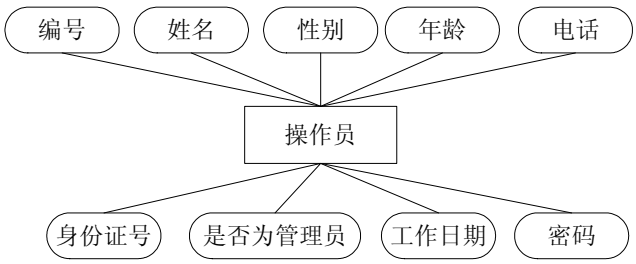


图 5.13　操作员信息实体 E-R 图

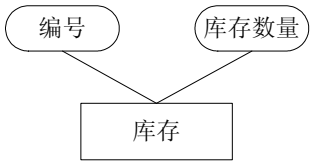


图 5.14　库存信息实体 E-R 图

5.4.3　使用 PowerDesigner 建模

在数据库概念设计中已经分析了本系统中主要的数据库实体对象，通过这些实体可以得出数据表结构的基本模型，最终这些实体将被创建成数据表，形成完整的数据结构。

笔者使用 PowerDesigner 软件对数据进行了建模操作，创建完成的数据库模型如图 5.15 所示。

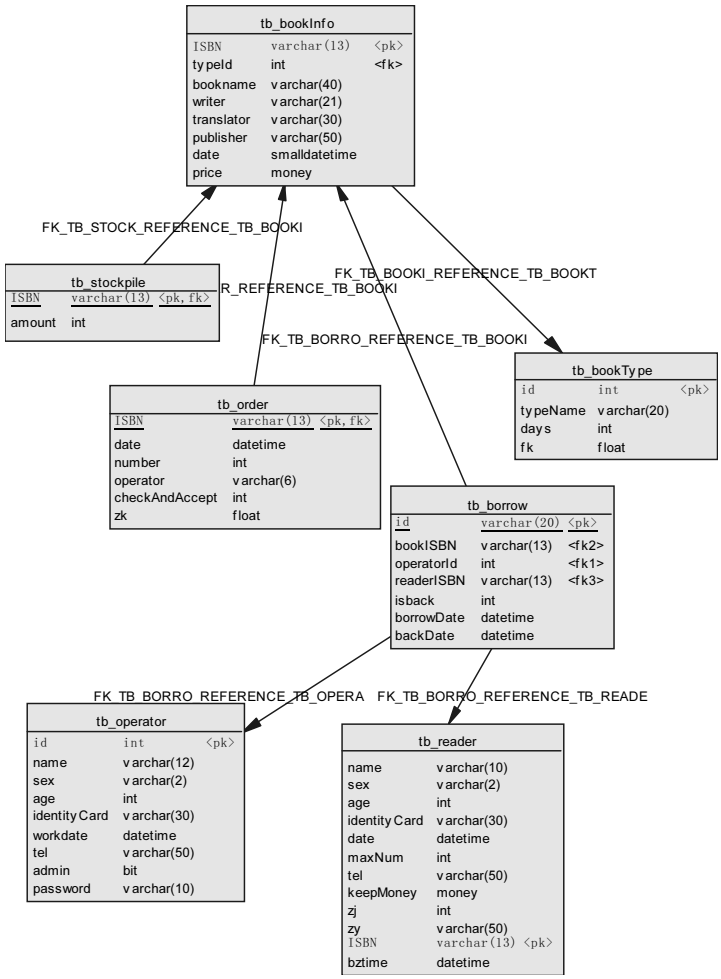


图 5.15　图书馆管理系统的数据库模型

5.5　公共模块设计

在开发过程中，经常会用到一些公共模块，如数据库连接及操作的类、限制文本框输入长度的类以及描述组合框索引与内容的类等，因此在开发系统前首先需要设计这些公共模块。下面将具体介绍图书馆管理系统中公共模块的设计过程。

5.5.1　数据库连接及操作类的编写

数据库连接及操作类通常包括连接数据库的方法 getConnection()、执行查询语句的方法 executeQuery()、执行更新操作的方法 executeUpdate()、关闭数据库连接的方法 close()。下面将详细介绍如何编写图书馆管理系统中的数据库连接及操作的类 Dao.java。步骤如下：

（1）指定类 Dao.java 保存的包，并导入所需的类包，本例将其保存到 com.wsy.dao 包中。关键代码如下：

例程 01　代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
package com.wsy.dao;                          //指定类的包名称
import java.sql.Connection;                   //导入进行数据库连接时所使用的 java.sql.Connection 类
import java.sql.DriverManager;                //导入进行数据库连接时所使用的 java.sql.DriverManager 类
import java.sql.ResultSet;                    //导入进行数据表查询时所使用的 java.sql.ResultSet 类
import java.sql.SQLException;                 //导入进行数据库操作时捕捉异常使用的 java.sql.SQLException 类
```

注意　包语句以关键字 package 后面紧跟一个包名称，然后以分号“;”结束；包语句必须出现在 import 语句之前；一个.java 文件只能有一个包语句。

（2）在 Dao.java 类的构造方法中创建数据库连接操作。在此类中首先定义数据库连接驱动包名、数据库连接路径、数据库连接用户名、密码等静态变量，然后在构造函数中实现数据库连接操作。在数据库连接代码中需要添加 try...catch 关键字，捕捉数据库连接时可能抛出的异常。关键代码如下：

例程 02　代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
//定义驱动包名称
protected static String dbClassName = "com.microsoft.jdbc.sqlserver.SQLServerDriver";
protected static String dbUrl = "jdbc:microsoft:sqlserver://localhost:1433;"
        + "DatabaseName=db_library;SelectMethod=Cursor";          //定义数据库连接路径
protected static String dbUser = "sa";                            //定义数据库连接用户名
protected static String dbPwd = "";                               //定义数据库连接密码
protected static String second = null;
private static Connection conn = null;                            //定义一个数据库连接
private Dao() {
    try {                                                         //捕捉数据库连接异常
        if (conn == null) {                                       //如果连接为空
            Class.forName(dbClassName).newInstance();             //装载 SQL Server 驱动
```

```
                conn = DriverManager.getConnection(dbUrl, dbUser, dbPwd);   //获取数据库连接
            }
            else                                                            //如果连接不为空
                return;                                                     //返回
        } catch (Exception ee) {
            ee.printStackTrace();                                           //捕捉数据库连接异常
        }
    }
```

（3）创建执行查询语句的方法 executeQuery()，其返回值为 ResultSet 结果集。首先需要初始化 Dao 对象，调用构造函数，从而获取数据库连接。有一点值得注意，就是在创建数据库连接前首先判断数据库连接是否为空，如果为空再创建数据库连接，避免造成程序资源的浪费。executeQuery()方法的关键代码如下：

例程 03 代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
private static ResultSet executeQuery(String sql) {
    try {                                                                   //捕捉数据库操作异常
        if(conn==null)                                                      //数据库连接如果为空
❶       new Dao();
        return
    conn.createStatement(
❷               ResultSet.TYPE_SCROLL_SENSITIVE,
❸               ResultSet.CONCUR_UPDATABLE).executeQuery(sql);  //返回一个 ResultSet 结果集
    } catch (SQLException e) {
        e.printStackTrace();                                                //捕捉异常
        return null;
    } finally {
    }
}
```

代码贴士

❶ 调用构造函数创建数据库连接。

❷ ResultSet.TYPE_SCROLL_SENSITIVE: 常量允许记录指针向前或向后移动，且当 ResultSet 对象变动记录指针时，会影响记录指针的位置。这种类型的设置使结果集受到其他用户所作更改的影响。例如当一个用户正在浏览记录时，其他用户的操作使数据库中的数据发生了变化，这时当前用户所获取的记录集中的数据也会同步发生改变。

❸ ResultSet.CONCUR_UPDATABLE: 这种类型的设置支持对 ResultSet 的动态更新。

（4）创建执行更新操作的方法 executeUpdate()，它的返回值为 int 型的整数，此返回值代表数据表更新操作是否成功，返回 1 代表成功，返回-1 代表没有成功。executeUpdate ()方法的关键代码如下：

例程 04 代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
private static int executeUpdate(String sql) {
    try {                                                                   //捕捉数据库操作异常
        if(conn==null)                                                      //如果数据库连接为空
            new Dao();                                                      //获取数据库连接
        return conn.createStatement().executeUpdate(sql);                   //进行数据库更新操作
    } catch (SQLException e) {
```

```
            System.out.println(e.getMessage());                    //打印捕捉的异常
            return -1;                                             //返回-1
        } finally {
        }
}
```

（5）为了避免运行程序时资源的浪费，优化项目运行速度，需要在完成数据库操作后，关闭数据库连接，所以笔者在 Dao.java 类中创建了关闭数据库连接的方法 close()。为了使数据库连接在程序结束后确定会被关闭，在 close()方法中加入了 finally 字段，在 finally 块中将数据库连接置空。关键代码如下：

例程 05　代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
public static void close() {
    try {                                                          //捕捉异常
        conn.close();                                              //关闭数据库连接
    } catch (SQLException e) {
        e.printStackTrace();                                       //捕捉异常
    }finally{
        conn = null;                                               //在最终执行块中将数据库连接置空
    }
}
```

5.5.2　MenuActions 类的编写

通常激活同一个命令有多种方式，用户可以通过工具栏中按钮、菜单选择特定的功能。在本系统中，最常用的命令就是弹出内部窗体，笔者将本系统中需要弹出的内部窗体命令统一放入 MenuActions 类中，这样触发任何一种组件事件时，都会按照统一的方式处理。

Swing 包提供了一个非常有用的机制，用来封装命令，并将其连接到多个事件源，这种机制就是 Action 接口。Action 接口有如下方法：

- ☑ public void actionPerformed(ActionEvent e)。
- ☑ public Object getValue(String key)。
- ☑ public void putValue(String key, Object value)。
- ☑ public boolean isEnabled()。
- ☑ public void setEnabled(boolean b)。
- ☑ public void addPropertyChangeListener(PropertyChangeListener listener)。
- ☑ public void removePropertyChangeListener(PropertyChangeListener listener)。

其中第一个方法在实现 ActionListener 接口的程序中经常看到，实际上 Action 接口扩展了 ActionListener 接口。

getValue()与 putValue()方法用来存储与提取动作对象的预定义名称与值。例如：

```
action.putValue(Action.SMALL_ICON,new ImageIcon("*.gif"));          //将图标存储到动作对象中
```

表 5.1 列举了几种常用的动作对象的预定义名称。

表 5.1　动作对象的预定义名称

名　　称	值	名　　称	值
NAME	名称，显示在按钮或菜单上	SHORT_DESCRIPTION	简单提示说明，当鼠标放在按钮或菜单上时出现提示
SMALL_ICON	小图标，显示在按钮或菜单上	LONG_DESCRIPTION	详细提示说明

setEnabled()方法用于开启或禁用动作对象，isEnabled()方法用于检查动作是否启用。

实现 Action 接口需要将接口中的所有方法都实现，所以在通常情况下都使用实现该接口的 AbstractAction 类，本系统中的 MenuActions 类正是继承了 AbstractAction 类，在 MenuActions 类中只要重写 AbstractAction 类中的 actionPerformed()方法即可。

下面以系统中的“更改密码”菜单项为例说明 MenuActions 类的编写。以下代码是在 MenuActions 类中创建一个内部类，这个内部类用于创建菜单栏中“更改密码”菜单项的动作对象，在此类的构造函数中创建组件的提示说明，在 actionPerformed()方法执行“更改密码”窗体的弹出操作。关键代码如下：

例程 06　代码位置：光盘\TM\05\src\com\wsy\MenuActions.java

```
private static class PasswordModiAction extends AbstractAction {
    PasswordModiAction() {
        putValue(Action.NAME,"更改密码");                          //将菜单项名称设置为“更改密码”
        putValue(Action.LONG_DESCRIPTION, "修改当前用户密码");
        putValue(Action.SHORT_DESCRIPTION, "更换密码");    //在“更改密码”菜单项显示的提示文字
        //putValue(Action.SMALL_ICON,CreatecdIcon.add("bookAddtb.jpg"));
        //将图标存储到动作对象中
        //setEnabled(false);                                       //使动作禁用
    }
    public void actionPerformed(ActionEvent e) {
        if (!frames.containsKey("更改密码")||frames.get("更改密码").isClosed()) {
            GengGaiMiMa iframe=new GengGaiMiMa();          //初始化更改密码内部窗体
            frames.put("更改密码", iframe);
            Library.addIFame(frames.get("更改密码"));          //将内部窗体添加到外部窗体中
        }
    }
}
```

将此内部类的对象作为 MenuActions 类的成员变量，然后再使用 static 定义一个静态区域进行初始化。类在被加载时，首先执行 static 定义的静态区域内部的代码，且只会被执行一次。关键代码如下：

```
public static PasswordModiAction MODIFY_PASSWORD;          //修改密码窗体动作
static {
    MODIFY_PASSWORD = new PasswordModiAction();            //初始化修改密码内部类对象
}
```

同理，菜单栏中其他菜单项与子菜单中的菜单项也是以相同方式被封装到 MenuActions 类中的。当某个组件需要使用这个动作对象时，以按钮为例，可以使用如下代码实现：

```
JButton button=new JButton(MenuActions.MODIFY_PASSWORD);
```

5.5.3　限制文本框长度类的编写

在 Swing 语言创建的窗体中，当 JTextField 组件创建时，可以指定文本框的宽度。例如：

```
JPanel panel=new JPanel();                          //创建面板
JTextField textField=new JTextField(20);            //创建文本框
panel.add(textField);                               //将文本框添加到面板中
```

但在 JTextField 的构造器中设定的宽度并不是用户能输入的字符个数上限，用户可以在文本框中输入一个更长的字符串，此时需要限制用户输入字符串的长度，笔者创建了限制文本框输入长度的类 MyDocument.java。创建此类的步骤如下：

（1）创建 MyDocument.java 类，此类继承 PlainDocument 类。关键代码如下：

例程 07　代码位置：光盘\TM\05\src\com\wsy\util\MyDocument.java

```
public class MyDocument extends PlainDocument{
}
```

（2）在 MyDocument.java 类中创建两个构造函数，其中一个是有参数的，另一个是无参数的。关键代码如下：

例程 08　代码位置：光盘\TM\05\src\com\wsy\util\MyDocument.java

```
public MyDocument(int newMaxLength){                //设置文本框的最大长度
    super();                                        //执行父类构造方法
    maxLength = newMaxLength;                       //将参数赋予类成员变量
}
public MyDocument(){                                //无参的构造函数
    this(10);                                       //将数值 10 赋予类成员变量
}
```

（3）重载父类方法 insertString()，在此方法中限定文本框允许输入的字符串长度。关键代码如下：

例程 09　代码位置：光盘\TM\05\src\com\wsy\util\MyDocument.java

```
public void insertString(int offset, String str, AttributeSet a)
        throws BadLocationException {
    if (getLength() + str.length() > maxLength) {   //这里假定限制长度为 10
        return;                                     //返回
    } else {
        super.insertString(offset, str, a);
    }
}
```

（4）在程序设计中，当需要限制用户输入字符串长度时，可以使用如下代码：

```
JTextField textField = new JTextField("请输入 13 位书号",13);    //初始化文本框
textField.setDocument(new MyDocument(13));                      //设置“书号”文本框最大输入值为 13 位
```

5.5.4 描述组合框索引与内容类的编写

在程序编写的过程中，经常会遇到组合框组件的应用。有时要在窗体中的组合框中显示具体内容，通常需要在数据库中存储此组合框的索引值，这时便需要使用一种数据结构将组合框中的内容与索引值联系在一起。java.util.Map 形式是比较好的选择，可以使用 Map 接口中的 put()方法将索引值与具体内容放入集合中，当得到索引值时获取具体内容可以使用 Map 接口中的 get(key)方法。描述组合框索引与内容类的编写步骤如下：

（1）创建组合框组件的索引值与其所对应的内容的 Item.java 类，这个类中不仅包含代表组合框索引的成员变量 id 和代表组合框内容的成员变量 name，还包括这两个成员变量的 setXXX()、getXXX()方法。关键代码如下：

例程 10 代码位置：光盘\TM\05\src\com\wsy\JComPz\Item.java

```
package com.wsy.JComPz;
public class Item {
❶    public String id;                                    //组合框索引值
❷    public String name;                                  //组合框内容
     public String getId() {                              //id 对应的 getXXX() 方法
          return id;
     }
     public void setId(String id) {                       // id 对应的 setXXX() 方法
          this.id = id;
     }
     public String getName() {                            //name 对应的 getXXX() 方法
          return name;
     }
     public void setName(String name) {                   //name 对应的 setXXX() 方法
          this.name = name;
     }
     public String toString() {                           //重写 Object 类中的 toString() 方法
          return getName();
     }
}
```

代码贴士

❶ id：表示组合框中索引值的变量。

❷ name：表示组合框中具体内容的变量。

（2）创建 MapPz.java 类，使用 Map 关联组合框的索引与组合框的具体内容。这里以图书类别编号与图书类别创建组合框为例，首先在此类中初始化 Map 集合，取图书类别相关内容，将图书类别相关内容放入 Item 类中；然后将图书类别编号与图书类别名称放入 Map 集合中，可以使用 put()方法；最后返回类型为 Map 的集合。关键代码如下：

例程 11 代码位置：光盘\TM\05\src\com\wsy\JComPz\MapPz.java

```
package com.wsy.JComPz;
public class MapPz {
     static Map map = new HashMap();                      //初始化 Map 接口
```

```
    public static Map getMap() {
        List list = Dao.selectBookCategory();                    //获取图书类别相关内容
        for (int i = 0; i < list.size(); i++) {                  //循环操作
            BookType booktype = (BookType) list.get(i);          //取得集合中的值
            Item item = new Item();                              //初始化 Item 对象
            item.setId(booktype.getId());                        //将图书类别编号放入 Item 类中
            item.setName(booktype.getTypeName());                //将图书类别名称放入 Item 类中
            map.put(item.getId(), item);                         //将图书类别编号与 item 对象放入 map
        }
        return map;                                              //返回集合
    }
}
```

（3）上述代码中用到了 Dao.java 类中的 selectBookCategory()方法，此方法用于查询图书类别相关信息，首先将数据库查询的相关信息放入 JavaBean 中，然后将 JavaBean 对象添加到 list 集合中，最终将结果以 List 形式返回。关键代码如下：

例程 12　代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
public static List selectBookCategory() {
    List list=new ArrayList();                                   //初始化 List 对象
❶   String sql = "select * from tb_bookType";                    //查询图书类别表 SQL 语句
❷   ResultSet rs = Dao.executeQuery(sql);                        //执行 SQL 语句，返回 ResultSet 对象
    try {
        while (rs.next()) {                                      //循环结果集
❸           BookType bookType=new BookType();                    //初始化 BookType 对象
            bookType.setId(rs.getString("id"));                  //将数据库中查询 id 值赋予到 JavaBean 中
            //将数据库中查询 typeName 值赋予到 JavaBean 中
            bookType.setTypeName(rs.getString("typeName"));
            bookType.setDays(rs.getString("days"));          //将数据库中查询 days 值赋予到 JavaBean 中
            bookType.setFk(rs.getString("fk"));                  //将数据库中查询 fk 值赋予到 JavaBean 中
            list.add(bookType);                                  //将 JavaBean 对象添加到 list 中
        }
    } catch (Exception e) {
        e.printStackTrace();                                     //捕捉异常
    }
    Dao.close();                                                 //关闭数据库连接
    return list;                                                 //将集合返回
}
```

代码贴士

❶ sql：查询 tb_bookType（图书类别表）的全部内容。

❷ rs：执行 SQL 语句后返回的 ResultSet 结果集。

❸ bookType：实例化 BookType 类对象。

5.5.5　在 JLable 上添加图片类的编写

为了美化窗体，通常需要在窗体上添加图片。一般情况下使用如下方式添加图片：

☑　在窗体上添加 Jpanel。

☑ 在 JPanel 上添加 Jlable。

☑ 将图片初始化为 ImageIcon 对象。

☑ 使用 JLabel.setIcon(ImageIcon)代码实现在窗体上添加图片功能。

在这里笔者将上述操作封装在公共类中，命名为 CreatecdIcon.java 类，在此类中定义一个返回 ImageIcon 类对象的方法，此方法以当前图片的文件名称为参数初始化一个 ImageIcon 类对象。关键代码如下：

例程 13 代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
package com.wsy.util;
public class CreatecdIcon {
    public static ImageIcon add(String ImageName){                    //返回 ImageIcon 类型的对象
        URL IconUrl = Library.class.getResource("/"+ImageName);//当前图片的路径
        ImageIcon icon=new ImageIcon(IconUrl);                         //将路径封装到 ImageIcon 对象中
        return icon;                                                   //返回 icon 对象
    }
}
```

说明 Library.class.getResource("/1.jpg")指代的图片为项目名称下 res 文件下的图片，实际上“/”指代的路径为项目名称中的 res 文件。

当需要在 JLable 中添加图片时，可以使用如下代码：

```
final JLabel headLogo = new JLabel();                                  //创建 Jlable 对象
//使用 CreadedIcon 类中的 add()方法返回一个 ImageIcon 对象
ImageIcon bookModiAndDelIcon=CreatecdIcon.add("bookModiAndDel.jpg");
headLogo.setIcon(bookModiAndDelIcon);                                  //设置 Jlable 图片
```

5.6 主窗体设计

5.6.1 主窗体概述

管理员通过“系统登录”模块的验证后，可以登录到图书馆管理系统的主窗体。系统主窗体主要包括菜单栏、工具栏。用户在菜单栏中单击任一菜单项即可执行相应的功能；工具栏为用户提供了经常使用的功能按钮。主窗体的运行效果如图 5.16 所示。

图 5.16 图书馆管理系统主窗体的运行效果

5.6.2 主窗体技术分析

系统主窗体主要包括菜单栏与工具栏。

单击窗体顶端的菜单栏中的某一菜单项，可以打开下拉菜单，其中包含菜单项与子菜单项。当用户单击下拉菜单中某一菜单项时，窗体中所有的菜单都会被关闭。图 5.17 展示了一个典型的菜单栏。

图 5.17　主窗体的菜单栏

在主窗体的设计中，需要创建菜单栏与工具栏，此时用到了 JMenuBar 类与 JToolBar 类来创建菜单栏与工具栏。

菜单栏的创建比较简单，使用 JMenuBar 的构造函数初始化菜单栏即可。一般情况下将菜单栏显示在框架顶部。例如：

```
JMenuBar menuBar=new JMenuBar();                    //创建菜单栏
frame.setJMenu=new JMenu(menuBar);                  //将菜单栏放入顶层框架
```

对于每个菜单，需要创建一个对象，实际上就是菜单项名称。例如：

```
JMenu editMenu=new JMenu("图书类别管理");            //在菜单栏中创建顶层菜单
```

最后将顶层菜单添加到菜单栏中，可以使用 JMenuBar 类的 add()方法进行添加。例如：

```
menuBar.add(editMenu);                              //将顶层菜单添加到菜单栏中
```

可以在顶层菜单中添加菜单项、分割符与子菜单。其中菜单项可以使用 JMenuItem 类的构造函数进行初始化，然后使用 JMenuBar 类的 add()方法进行添加；分割符可以使用 JMenuBar 类的 addSeparator()方法进行添加；子菜单栏实质上也是一个菜单栏，与顶层菜单栏创建方式相同，可以使用 JMenuBar 类的 add()方法将子菜单项添加到顶层菜单中。例如：

```
JMenuItem exit=new JMenuItem("退出");               //在菜单栏中创建菜单项
editMenu.add(exit);                                 //将菜单项添加到顶层菜单中
editMenu.addSeparator();                            //添加分割符
JMenu bookTypeAdd=new JMenu("图书添加");             //创建子菜单
editMenu.add(bookTypeAdd);                          //在顶层菜单中添加子菜单
```

当用户选择一个菜单时，会引发一个动作事件，需要为每个菜单项添加监听器，重写 ActionListener 接口中的 actionPerformed()方法，在此方法中为菜单栏添加业务逻辑。例如：

```
exit.addActionListener(new ActionListener{
    public void actionPerformed(final ActionEvent e) {//实现 ActionListener 接口中的 actionPerformed()方法
    }
});
```

注意　这里为菜单项添加监听事件的代码使用了匿名内部类的形式，其优点是可以简化编码，而且在内部类中可以轻易使用外部类定义的局部变量，否则需要将这些局部变量声明为 final 类型的变量；而其缺点是程序的可读性较差，作为初学者可以不必使用匿名内部类形式创建组件事件。

通常情况下，菜单项触发命令通过实现 Action 接口即可。这里笔者设计了公共类 MenuAction.java，它继承了 AbstractAction 类，由于 AbstractAction 类实现了 Action 接口，所以继承 AbstractAction 类就等于实现了 Action 接口，因此这里使菜单栏添加 MenuAction 类对象即可。例如：

```
JMenu sysManageMenu = new JMenu();                          //系统维护
JMenu userManageMItem = new JMenu("用户管理");               //用户管理
sysManageMenu.add(MenuActions.MODIFY_PASSWORD);             //添加修改密码动作对象
```

主窗体中工具栏的创建也非常简单。工具栏为系统提供了迅速访问常用命令的一系列按钮。可以使用如下代码创建工具栏：

```
JToolBar bar=new JToolBar();
```

完成主窗体中工具栏的创建后同样需要添加 MenuActions 类对象实现工具栏事件，由于需要在工具栏中添加图标，所以将动作对象添加到按钮组件中，然后为按钮设置图标。例如：

```
//将图书信息修改对象附加给按钮组件
JButton bookModiAndDelButton=new JButton(MenuActions.BOOK_MODIFY);
ImageIcon bookmodiicon=CreatecdIcon.add("bookModiAndDeltb.jpg");   //创建图标方法
bookModiAndDelButton.setIcon(bookmodiicon)                         //为按钮设置图标
```

由于这些动作对象在 MenuActions 类中设计时有名称，所以使用如下代码可以使按钮只显示图标，不显示文字。例如：

```
bookModiAndDelButton.setHideActionText(true);                //使按钮文字隐藏
```

最后将按钮添加到工具栏中，例如：

```
toolBar.add(bookModiAndDelButton);
```

5.6.3 主窗体的实现过程

主窗体的实现步骤如下：

（1）创建 Library 类，在它的构造函数中设置主窗体相关属性，如窗体大小、窗体标题等相关属性，还可以为窗体设置背景图片，并调用创建菜单栏与工具栏的方法，在主窗体中创建菜单栏与工具栏。关键代码如下：

例程 14 代码位置：光盘\TM\05\src\com\wsy\Library.java

```
public Library() {
    super();                                                 //调用父类构造函数
    //将窗体关闭
    setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    setLocationByPlatform(true);
    setSize(800, 600);                                       //设置主窗体大小
    setTitle("图书馆管理系统");                               //设置窗体标题
❶   JMenuBar menuBar = createMenu();                        //调用创建菜单栏的方法
```

```
❷   setJMenuBar(menuBar);                                          //在主窗体中添加菜单栏
❸   JToolBar toolBar = createToolBar();                            //调用创建工具栏的方法
    getContentPane().add(toolBar, BorderLayout.NORTH);             //以 BorderLayout 布局添加工具栏
    final JLabel label = new JLabel();                             //初始化背景标签
//为桌面面板添加组件监听事件
DESKTOP_PANE.addComponentListener(new ComponentAdapter() {
        public void componentResized(final ComponentEvent e) {    //重写组件大小更改时的方法
            Dimension size = e.getComponent().getSize();           //获取当前组件的大小
            label.setSize(e.getComponent().getSize());             //将背景标签设置为组件的大小
            //将图片放置在背景标签中
            label.setText("<html><img width=" + size.width + " height="
                    + size.height + " src='"
                    + this.getClass().getResource("/backImg.jpg")
                    + "'></html>");
        }
    });
    DESKTOP_PANE.add(label,new Integer(Integer.MIN_VALUE));        //将背景标签添加到背景面板中
    getContentPane().add(DESKTOP_PANE);                            //将背景面板添加到窗体中
}
```

代码贴士

❶ menuBar：实例化一个 MenuBar 类对象，创建菜单栏。

❷ setJMenuBar()：在主窗体中添加菜单栏的方法。

❸ toolBar：实例化一个 ToolBar 类对象，创建工具栏。

（2）编写创建菜单栏方法，可以初始化 JMenuBar 类对象创建顶层菜单，并在顶层菜单上添加相关菜单项与子菜单，然后为菜单栏添加图标，为菜单栏添加图标可以使用 JMenu 类中的 setIcon()方法进行添加。关键代码如下：

例程 15　代码位置：光盘\TM\05\src\com\wsy\Library.java

```
private JMenuBar createMenu() {                                    //创建菜单栏的方法
    JMenuBar menuBar = new JMenuBar();                             //创建菜单栏
    JMenu bookOrderMenu = new JMenu();                             //初始化新书订购管理菜单
    bookOrderMenu.setIcon(CreatecdIcon.add("xsdgcd.jpg"));         //为新书订购菜单栏添加图片
    bookOrderMenu.add(MenuActions.NEWBOOK_ORDER);                  //添加新书订购动作对象，弹出新书订购窗体
    bookOrderMenu.add(MenuActions.NEWBOOK_CHECK_ACCEPT);//添加新书验收动作对象，弹出新书订购窗体
    JMenu baseMenu = new JMenu();                                  //初始化基础数据维护菜单
    baseMenu.setIcon(CreatecdIcon.add("jcsjcd.jpg"));              //为基础维护菜单栏添加图片
    {
        JMenu readerManagerMItem = new JMenu("读者信息管理");//添加“读者信息管理“子菜单项
        readerManagerMItem.add(MenuActions.READER_ADD);            //添加弹出读者添加窗体动作对象
        readerManagerMItem.add(MenuActions.READER_MODIFY);         //添加弹出读者修改与删除窗体对象
        JMenu bookTypeManageMItem = new JMenu("图书类别管理");//添加“图书类别管理”子菜单项
        bookTypeManageMItem.add(MenuActions.BOOKTYPE_ADD);//添加弹出图书类别添加窗体动作对象
        //添加弹出图书类别修改与删除窗体动作对象
        bookTypeManageMItem.add(MenuActions.BOOKTYPE_MODIFY);
        JMenu menu = new JMenu("图书信息管理");                    //添加“图书信息管理”子菜单项
        menu.add(MenuActions.BOOK_ADD);                            //添加弹出图书添加动作对象
```

```
        menu.add(MenuActions.BOOK_MODIFY);                          //添加弹出图书修改与删除动作对象
        baseMenu.add(readerManagerMItem);                           //在顶层菜单中添加“读者信息管理”子菜单
        baseMenu.add(bookTypeManageMItem);                          //在顶层菜单中添加“图书类别管理”子菜单
        baseMenu.add(menu);                                         //在顶层菜单中添加“图书信息管理”子菜单
        baseMenu.addSeparator();                                    //添加分割符
        baseMenu.add(MenuActions.EXIT);                             //在顶层菜单中添加退出菜单项
    }
    JMenu borrowManageMenu = new JMenu();                           //借阅管理菜单
    borrowManageMenu.setIcon(CreatecdIcon.add("jyglcd.jpg"));       //为菜单添加图标
    borrowManageMenu.add(MenuActions.BORROW);                       //添加图书借阅动作对象
    borrowManageMenu.add(MenuActions.GIVE_BACK);                    //添加图书归还动作对象
    borrowManageMenu.add(MenuActions.BOOK_SEARCH);                  //添加图书搜索动作对象
    JMenu sysManageMenu = new JMenu();                              //系统维护菜单
    sysManageMenu.setIcon(CreatecdIcon.add("jcwhcd.jpg"));          //为菜单添加图标
    JMenu userManageMItem = new JMenu("用户管理");                  //初始化“用户管理”子菜单项
    userManageMItem.add(MenuActions.USER_ADD);                      //添加用户添加动作对象
    userManageMItem.add(MenuActions.USER_MODIFY);                   //添加用户修改与删除动作对象
    sysManageMenu.add(MenuActions.MODIFY_PASSWORD);                 //添加更改密码动作对象
    sysManageMenu.add(userManageMItem);                             //将子菜单添加到顶层菜单中
    menuBar.add(baseMenu);                                          //添加基础数据维护菜单到菜单栏
    menuBar.add(bookOrderMenu);                                     //添加新书订购管理菜单到菜单栏
    menuBar.add(borrowManageMenu);                                  //添加借阅管理菜单到菜单栏
    menuBar.add(sysManageMenu);                                     //添加系统维护菜单到菜单栏
    return menuBar;                                                 //返回菜单栏
    }
```

（3）编写创建工具栏的方法，创建工具栏可以使用 JToolBar 类，创建工具栏后将所有的图标添加至工具栏中，可以为每个图标添加提示信息。由于在创建 MenuActions 类时已经为每个内部窗体动作添加了提示信息，所以这里可以不为图标添加提示信息。关键代码如下：

例程 16 代码位置：光盘\TM\05\src\com\wsy\Library.java

```
private JToolBar createToolBar() {                                  //创建工具栏的方法
    JToolBar toolBar = new JToolBar();                              //创建工具栏
    toolBar.setFloatable(false);                                    //取消工具栏浮动（不能拖动到别的位置）
    toolBar.setBorder(new BevelBorder(BevelBorder.RAISED));         //设置工具栏的边框
    //为按钮添加图书添加动作对象
    JButton bookAddButton=new JButton(MenuActions.BOOK_ADD);
    //添加工具栏图标
    ImageIcon icon=new ImageIcon(Library.class.getResource("/bookAddtb.jpg"));
    bookAddButton.setIcon(icon);                                    //为按钮添加图标
    bookAddButton.setHideActionText(true);                          //使按钮上的文字隐藏
    toolBar.add(bookAddButton);                                     //在工具栏中添加此按钮
    //为按钮添加图书修改与删除动作对象
    JButton bookModiAndDelButton=new JButton(MenuActions.BOOK_MODIFY);
    ImageIcon bookmodiicon=CreatecdIcon.add("bookModiAndDeltb.jpg");//创建图标方法
    bookModiAndDelButton.setIcon(bookmodiicon);                     //为按钮添加图标
    bookModiAndDelButton.setHideActionText(true);                   //使按钮上的文字隐藏
    toolBar.add(bookModiAndDelButton);                              //在工具栏中添加此按钮
    //为按钮添加图书类别添加动作对象
```

```
JButton bookTypeAddButton=new JButton(MenuActions.BOOKTYPE_ADD);
//创建图标方法
ImageIcon bookTypeAddicon=CreatecdIcon.add("bookTypeAddtb.jpg");
bookTypeAddButton.setIcon(bookTypeAddicon);                    //为按钮添加图标
bookTypeAddButton.setHideActionText(true);                     //使按钮上的文字隐藏
toolBar.add(bookTypeAddButton);                                //在工具栏中添加此按钮
//为按钮添加图书借阅动作对象
JButton bookBorrowButton=new JButton(MenuActions.BORROW);
//创建图标方法
ImageIcon bookBorrowicon=CreatecdIcon.add("bookBorrowtb.jpg");
bookBorrowButton.setIcon(bookBorrowicon);                      //为按钮添加图标
bookBorrowButton.setHideActionText(true);                      //使按钮上的文字隐藏
toolBar.add(bookBorrowButton);                                 //在工具栏中添加此按钮
//为按钮添加图书订购动作对象
JButton bookOrderButton=new JButton(MenuActions.NEWBOOK_ORDER);
//创建图标方法
ImageIcon bookOrdericon=CreatecdIcon.add("bookOrdertb.jpg");
bookOrderButton.setIcon(bookOrdericon);                        //为按钮添加图标
bookOrderButton.setHideActionText(true);                       //使按钮上的文字隐藏
toolBar.add(bookOrderButton);                                  //在工具栏中添加此按钮
//为按钮添加图书验收动作对象
JButton bookCheckButton=new JButton(MenuActions.NEWBOOK_CHECK_ACCEPT);
//创建图标方法
ImageIcon bookCheckicon=CreatecdIcon.add("newbookChecktb.jpg");
bookCheckButton.setIcon(bookCheckicon);                        //为按钮添加图标
bookCheckButton.setHideActionText(true);                       //使按钮上的文字隐藏
toolBar.add(bookCheckButton);                                  //在工具栏中添加此按钮
//为按钮添加读者添加动作对象
JButton readerAddButton=new JButton(MenuActions.READER_ADD);
ImageIcon readerAddicon=CreatecdIcon.add("readerAddtb.jpg");//创建图标方法
readerAddButton.setIcon(readerAddicon);                        //为按钮添加图标
readerAddButton.setHideActionText(true);                       //使按钮上的文字隐藏
toolBar.add(readerAddButton);                                  //在工具栏中添加此按钮
//为按钮添加读者修改与删除动作对象
JButton readerModiAndDelButton=new JButton(MenuActions.READER_MODIFY);
ImageIcon readerModiAndDelicon=CreatecdIcon.add("readerModiAndDeltb.jpg");//创建图标方法
readerModiAndDelButton.setIcon(readerModiAndDelicon);          //为按钮添加图标
readerModiAndDelButton.setHideActionText(true);                //使按钮上的文字隐藏
toolBar.add(readerModiAndDelButton);                           //在工具栏中添加此按钮
//为按钮添加退出动作对象
JButton ExitButton=new JButton(MenuActions.EXIT);
ImageIcon Exiticon=CreatecdIcon.add("exittb.jpg");             //创建图标方法
ExitButton.setIcon(Exiticon);                                  //为按钮添加图标
ExitButton.setHideActionText(true);                            //使按钮上的文字隐藏
toolBar.add(ExitButton);                                       //在工具栏中添加此按钮
return toolBar;                                                //返回工具栏
}
```

（4）在 Library.java 类中的主函数中调用登录窗体，如果登录成功，初始化 Library.java 对象；如

果登录失败，则弹出提示对话框。关键代码如下：

例程 17 代码位置：光盘\TM\05\src\com\wsy\Library.java

```
public static void main(String[] args) {
    try {
        UIManager.setLookAndFeel(UIManager
                .getSystemLookAndFeelClassName());
        new BookLoginIFrame();                          //初始化登录窗体
    } catch (Exception ex) {
        ex.printStackTrace();                           //捕捉异常
    }
}
```

5.7 登录模块设计

5.7.1 登录模块概述

登录模块是图书馆管理系统的入口，在运行本系统后，首先进入的便是登录窗体。在该窗体中，系统管理员可以通过输入正确的管理员名称与密码登录到系统，当用户没有输入管理员名称或密码时，系统将会弹出相应的提示信息。系统登录模块的运行效果如图 5.18 所示。

图 5.18 图书馆管理系统登录窗体

注意 在实现系统登录前，需要在 SQL Server 数据库中手动添加一条系统管理员的数据（管理员名为 tsoft、密码为 111、拥有所有权限），即在操作员信息表 tb_operator 中各添加一条数据。SQL 语句代码如下：

```
#添加管理员信息
INSERT INTO tb_operator (name, password) VALUES ('tsoft', '111')
```

5.7.2 登录模块技术分析

在本系统中，登录模块窗体继承了 JFrame 类。在设计登录窗体前，需要初始化 JPanel 组件，然后设置 JPanel 的布局。依据登录模块的整体布局，笔者在登录窗体中使用了 BorderLayout 布局管理器。

BorderLayout 布局管理器是 JFrame 的默认布局管理器，它可以让程序员选择每个组件的摆放位置，可以选择将组件放在窗体的北部、中部、南部、东部或者西部。例如：

```
class Mypanel extends JPanel{
    setLayout(new BorderLayout());
    add(button,BorderLayout.SOUTH);
}
```

笔者将放置图片的 JLable 摆放在面板的北部，装载登录文本框的面板放置在中部，南部放置装载按钮的面板，中部的面板使用 GridLayout 布局管理器。GridLayout 布局管理器按照行列来排列所有的组件。可以使用如下代码设置网格初始化网格布局管理器：

```
panel.setLayout(new GridLayout(5,4));                    //在初始化时分别指定网格的行数与列数
```

在南部的面板中使用 FlowLayout 布局管理器。

中部的面板放置用户名标签、用户名文本框与密码标签、密码文本框，其中用户名文本框使用 JTextField 组件，密码文本框使用 JPasswordField 组件，可以在初始化文本框时指定文本框的列数与文本框中的初始值。例如：

```
JTextField textField=new JTextField("Default input",20)  //指定文本框列数与初始值
JPasswordField password = new JPasswordField(20);        //初始化密码框
```

为了增加登录窗体的美观，将密码框的回显字符设置为“*”。可以使用如下代码进行设置：

```
password.setEchoChar('*');                               //设置密码框的回显字符
```

当窗体设计完成后，需要进行管理员登录验证操作，这时需要为“登录”按钮添加按钮监听事件。可以将按钮监听事件写入内部类中，它实现 ActionListener 接口，在内部类中重写 actionPerformed()方法实现登录验证操作。

5.7.3　登录模块的实现过程

开发登录模块的具体步骤如下：

（1）在 BookLoginIFrame 类构造函数中设计登录窗体的整体布局，包括添加窗体关闭按钮、最小化按钮、设置窗体大小等相关属性。关键代码如下：

例程 18　代码位置：光盘\TM\05\src\com\wsy\iframe\BookLoginIFrame.java

```
public BookLoginIFrame() {
❶   super();
❷   final BorderLayout borderLayout = new BorderLayout();   //初始化 BorderLayout 布局管理器
❸   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);         //设置窗体关闭形式
❹   getContentPane().setLayout(borderLayout);               //面板使用 BorderLayout 布局管理器
❺   setTitle("图书馆管理系统登录");                          //设置窗体标题
    final JPanel panel_2 = new JPanel();
    final GridLayout gridLayout = new GridLayout(0, 2);    //初始化网格布局管理器
    gridLayout.setHgap(5);                                 //设置水平间距
    gridLayout.setVgap(20);                                //设置垂直间距
    panel_2.setLayout(gridLayout);                         //设置面板使用网格布局管理器
    final JLabel label = new JLabel();                     //创建放置用户名文字的标签
    label.setHorizontalAlignment(SwingConstants.CENTER);   //将标签中的文字设置在标签中间
    panel_2.add(label);                                    //在面板中添加标签
    label.setText("用　户　名：");                          //设置标签上的文字为“用户名”
    username = new JTextField(20);                         //初始化“用户名”文本框
    panel_2.add(username);                                 //在面板中添加此文本框
```

```
        final JLabel label_1 = new JLabel();
        label_1.setHorizontalAlignment(SwingConstants.CENTER);
        panel_2.add(label_1);
        label_1.setText("密          码：");                    //设置标签文字为“密码”
        password = new JPasswordField(20);                    //初始化“密码”文本框
        password.setEchoChar('*');                            //设置密码框的回显字符
        panel_2.add(password);                                //在面板中添加“密码”文本框
        final JPanel panel_1 = new JPanel();
        panel.add(panel_1, BorderLayout.SOUTH);               //设置在南部面板的布局
        login=new JButton();                                  //初始化登录按钮
        login.addActionListener(new BookLoginAction());       //为“登录”按钮添加监听事件
        login.setText("登录");                                //为按钮赋予名称
        panel_1.add(login);                                   //在面板中添加按钮
        reset=new JButton();                                  //初始化“重置”按钮
        reset.addActionListener(new BookResetAction());       //为“重置”按钮添加监听事件
        reset.setText("重置");                                //为按钮赋予名称
        panel_1.add(reset);                                   //在面板中添加“重置”按钮
        final JLabel tupianLabel = new JLabel();
❻       ImageIcon loginIcon=CreatecdIcon.add("login.JPG");    //在北部面板中添加图片
❼       tupianLabel.setIcon(loginIcon);                       //将图片放置在背景标签中
        setVisible(true);                                     //设置窗体可视
}
```

代码贴士

❶ super：调用父类的构造函数。

❷ borderLayout：实例化 BorderLayout 类，初始化边界布局管理器。

❸ setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)：设置窗体关闭的方式。

❹ getContentPane().setLayout(borderLayout)：设置容器使用边界布局管理器。

❺ setTitle()：设置窗体标题。

❻ loginIcon：调用 CreatecdIcon 类的 add()方法返回 ImageIcon 类对象。

❼ Label.setIcon(loginIcon)：使用 JLable 类中 setIcon()方法将图片放置在背景标签中，作为窗体背景。

（2）为方便在登录验证时取值传值，需要创建一个对应于 tb_operator 表字段的 JavaBean，这个类除了以数据表字段命名的成员变量外，还创建与成员变量相对应的 setXXX()、getXXX()方法。关键代码如下：

例程 19 代码位置：光盘\TM\05\src\com\wsy\model\Operater.java

```
public class Operater {
    private String id;                                        //操作员编号
    private String name;                                      //名称
    private String grade;                                     //级别
    private String password;                                  //密码
    public String getGrade() {                                //grade 对应的 getXXX()方法
        return grade;
    }
    public void setGrade(String grade) {                      //grade 对应的 setXXX()方法
        this.grade = grade;
```

```
    }
    public String getId() {                                    //id 对应的 getXXX()方法
        return id;
    }
    public void setId(String id) {                             //id 对应的 setXXX()方法
        this.id = id;
    }
    public String getName() {                                  //name 对应的 getXXX()方法
        return name;
    }
    public void setName(String name) {                         //name 对应的 setXXX()方法
        this.name = name;
    }
    public String getPassword() {                              //password 对应的 getXXX()方法
        return password;
    }
    public void setPassword(String password) {                 //password 对应的 setXXX()方法
        this.password = password;
    }
}
```

（3）为了在其他窗体中取得当前登录用户名称，需要在 BookLoginIFrame.java 类中创建一个 Operater 类型的成员变量，同时创建对应的 setXXX()与 getXXX()方法，这样在其他窗体中如果需要显示当前登录用户的名称，只需要使用 BookLoginIFrame.java 类中的 getXXX()方法取得 Operater 类型的对象即可。关键代码如下：

例程 20　代码位置：光盘\TM\05\src\com\wsy\iframe\BookLoginIFrame.java

```
private static Operater user;                                  //Operater 对象
public static Operater getUser() {                             //getXXX()方法
    return user;
}
public static void setUser(Operater user) {                    //setXXX()方法
    BookLoginIFrame.user = user;
}
```

（4）分别为“登录”按钮与“重置”按钮设置监听事件。在“登录”按钮监听事件中，首先判断“用户名”与“密码”文本框是否为空，如果为空，说明用户没有输入，此时需要弹出提示对话框；当用户输入用户名与密码后，需要以这两个文本框的值作为参数调用 Dao 类中的验证管理员登录的方法，如果验证成功，进入系统；如果失败，弹出提示对话框。“重置”按钮监听事件实现起来相对比较简单，只要将“用户名”文本框的值与“密码”文本框的值置空即可。关键代码如下：

例程 21　代码位置：光盘\TM\05\src\com\wsy\iframe\BookLoginIFrame.java

```
class BookLoginAction implements ActionListener {
    public void actionPerformed(final ActionEvent e) {
        user = Dao.check(username.getText(), password.getText());      //调用 Dao 类中验证登录方法
        if (user.getName() != null) {                                  //如果验证成功
            try {
```

```
                    Library frame = new Library();                          //初始化系统主窗体
                    frame.setVisible(true);                                 //使主窗体可视
                    BookLoginIFrame.this.setVisible(false);                 //使本窗体隐藏
                } catch (Exception ex) {
                    ex.printStackTrace();                                   //捕捉异常
                }
            } else {
            //如果没有验证成功，弹出对话框
                JOptionPane.showMessageDialog(null, "只有管理员才可以登录！");
                username.setText("");                                       //将“用户名”文本框置空
                password.setText("");                                       //将“密码”文本框置空
            }
        }
    }
    private class BookResetAction implements ActionListener {
        public void actionPerformed(final ActionEvent e){
            username.setText("");                                           //将“用户名”文本框置空
            password.setText("");                                           //将“密码”文本框置空

        }
    }
```

（5）在 Dao 类中创建登录验证方法，在此方法中查询文本框中输入字符串是否与操作员数据表数据匹配，并且是否为管理员，以上条件都满足，登录验证才成功。关键代码如下：

例程 22 代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
public static Operater check(String name, String password) {
    int i = 0;
    Operater operater=new Operater();                                       //初始化 Operater 类对象
❶   String sql = "select *    from tb_operator where name='" + name
            + "' and password='" + password + "'and admin=1";   //管理员登录验证 SQL 语句
❷   ResultSet rs = Dao.executeQuery(sql);                                   //执行 SQL 语句
    try {
        while (rs.next()) {                                                 //循环结果集
            String names = rs.getString(2);                                 //将数据库中名称存入变量 names 中
            operater.setId(rs.getString("id"));                             //将数据库中取出的值放入 JavaBean 中
            operater.setName(rs.getString("name"));
            operater.setGrade(rs.getString("admin"));
            operater.setPassword(rs.getString("password"));
            if (names != null) {                                            //如果变量 names 值不为空
                i = 1;                                                      //将变量 i 赋值为 1
            }
        }
    } catch (Exception e) {
        e.printStackTrace();                                                //捕捉异常
    }
❸   Dao.close();                                                           //关闭数据库连接
    return operater;                                                        //将 operater 对象返回
}
```

代码贴士

❶ sql：查询数据表 tb_operator（管理员信息），验证输入的用户名和密码以及是否为管理员与表中相符。

❷ Dao.executeQuery(sql)：执行 SQL 语句，并返回 ResultSet 类型的结果集。

❸ Dao.close()：关闭数据库连接。

5.8　图书信息管理模块设计

5.8.1　图书信息管理模块概述

图书信息管理模块主要包括图书信息添加、图书信息修改两个功能。

在图书信息添加窗体中管理员可以输入图书相关信息，包括名称、类别、图书条形码等。

进入图书信息修改窗体后首先在表格中显示所有图书的相关信息，管理员可以选择表格中需要修改的某一行数据，这时在窗体下方的文本框中显示相应的内容。

图书信息添加包括图书相关信息的添加，其中“出版社”与“类别”相关信息使用组合框组件在窗体中体现，比较特别的是“类别”组合框中的值是由数据库中的图书分类表取得的，除此之外其余图书相关信息字段以文本框的形式在窗体中体现，等待用户输入相关信息。同时在“添加”按钮监听事件中，限制用户输入非法字符串等操作，如果用户没有在窗体必添文本框中输入字符串而单击“添加”按钮，系统会弹出错误提示对话框。

图书信息修改主要实现图书相关信息的修改，首先查询图书信息表中的内容，放置到表格中，在表格监听事件中将表格内容放置在相应文本框中，用户可以通过修改文本框的内容修改图书相关信息。

图书信息添加与修改窗体的运行效果如图 5.19 和图 5.20 所示。

图 5.19　图书信息添加窗体

图 5.20　图书信息修改窗体

5.8.2　图书信息管理模块技术分析

设计图书信息管理窗体首先设计窗体布局。图书信息添加窗体中主要包括文本框与组合框以及按钮图片背景标签，布局比较简单，可以使用 BorderLayout 布局管理器。首先将图片背景标签放在窗体北部，将承载着图书信息文本框的面板放在窗体中部，最后将承载着按钮群的面板放置在窗体南部。

图书信息修改窗体比较复杂，除了图书相关信息文本框之外，窗体中还需要摆放一个带滚动条的表格。在这里窗体同样使用 BorderLayout 布局管理器，首先将背景图片标签摆放到窗体北部，在窗体的中部摆放一个面板，面板中放置表格与图书相关信息文本框，最后将按钮放在窗体的南部。

设计窗体布局后，需要进行事件处理。在本模块设计中，除了为“修改”按钮添加监听事件之外，还需要为表格添加鼠标监听事件。按钮监听事件的语法读者可以参看主窗体技术分析章节，添加鼠标监听事件可以使用如下代码：

```
table.addMouseListener(new MouseAdapter() {                    //鼠标监听事件
    public void mouseClicked(final MouseEvent e) {
        …//可以进行相关操作
    }
});
```

在添加图书信息时，需要对用户输入的字符串进行限制，包括字符串位数、字符串内容等，这时需要为相关文本框添加键盘监听事件。可以使用如下代码：

```
ISBN.addKeyListener(new KeyAdapter() {                         //键盘监听事件
    public void keyPressed(final KeyEvent e) {
        …//相关操作
    }
});
```

在添加图书信息时，为了避免用户添加相同的图书条形码，引发异常，需要在用户输入图书条形码后与数据库中图书信息表中的条形码进行比较，这时需要在“图书条形码”文本框中添加焦点监听事件。可以使用如下代码：

```
ISBN.addFocusListener(new FocusAdapter(){                      //焦点监听事件
    public void focusLost(FocusEvent e){
        …//相关操作
    }
});
```

注意 这里鼠标监听事件与键盘监听事件并没有实现 MouseListener 接口与 KeyListener 接口，而是继承这两个接口的相应的适配器，分别为 MouseAdapter 类与 KeyAdapter 类。适配器定义了接口中所有的方法，但这些方法什么也不做，使用适配器的目的是为了节约时间，因为实现接口需要将接口中的所有方法都实现，但其中有些方法可能是程序中不需要实现的。

5.8.3 图书信息管理模块的实现过程

1. 图书信息添加

图书信息添加模块的开发步骤如下：

（1）创建图书信息添加窗体，可以在构造函数中对此窗体进行布局，由于需要在主窗体内部弹出图书信息添加窗体，所以这里使用内部框架的机制，BookAddIFrame.java 类继承了 JInternalFrame 类。

关键代码如下：

例程 23　代码位置：光盘\TM\05\src\com\wsy\iframe\BookAddIFrame.java

```
public class BookAddIFrame extends JInternalFrame {
    public BookAddIFrame() {
        super();
        final BorderLayout borderLayout = new BorderLayout();
        getContentPane().setLayout(borderLayout);
        setIconifiable(true);                                    //设置窗体可最小化
        setClosable(true);                                       //设置窗体可关闭
        setTitle("图书信息添加");                                 //设置窗体标题
        setBounds(100, 100, 396, 260);                           //设置窗体位置和大小
        …//部分代码省略
❶       ISBN = new JTextField("请输入 13 位书号",13);             //初始化“图书条形码”文本框
❷       ISBN.setDocument(new MyDocument(13));                    //设置“书号”文本框最大输入值为 13
❸       ISBN.addKeyListener(new ISBNkeyListener());    //为“图书条形码”文本框添加键盘监听事件
❹       ISBN.addFocusListener(new ISBNFocusListener());//为“图书条形码”文本框添加焦点监听事件
        panel.add(ISBN);                                         //将“图书条形码”文本框添加到面板中
        bookType = new JComboBox();                              //初始化组合框
        //将组合框转换为 DefaultComboBoxModel 形式
        bookTypeModel= (DefaultComboBoxModel)bookType.getModel();
❺       List list=Dao.selectBookCategory();                      //从数据库中取出图书类别
        for(int i=0;i<list.size();i++){                          //循环结果集
            BookType booktype=(BookType)list.get(i);//将数据库读取内容放入图书类别 JavaBean 中
            Item item=new Item();                                //初始化 Item 对象
❻           item.setId((String)booktype.getId());                //将图书类别编号放入 Item 类中
            //将图书类别名称放入 Item 类中
❼           item.setName((String)booktype.getTypeName());
            //将 Item 对象添加到组合框中，在组合框中显示 Item 类中定义的 toString() 方法中的内容
❽           bookTypeModel.addElement(item);
        }
        …//省略部分代码
        publisher = new JComboBox();                             //初始化出版社组合框
        //将出版社信息放入 array 数组中
        String[]array=new String[]{"***出版社","**信息出版社","**大型出版社","***小型出版社"};
        //将数组内容添加到组合框中
        publisher.setModel(new DefaultComboBoxModel(array));
        …//省略部分代码
        //初始化日期格式
❾       SimpleDateFormat myfmt=new SimpleDateFormat("yyyy-MM-dd");
        //将文本框格式化为指定日期格式
❿       pubDate= new JFormattedTextField(myfmt.getDateInstance());
        pubDate.setValue(new java.util.Date());                  //将当前日期赋予到文本框中
        price= new JTextField();
        price.setDocument(new MyDocument(5));                    //设置“单价”文本框只能输入 5 位
        price.addKeyListener(new NumberListener());              //限制“单价”文本框只能输入数字
        …//省略部分代码
        buttonadd= new JButton();                                //初始化按钮
        //添加按钮监听事件
```

```
        buttonadd.addActionListener(new addBookActionListener());
        …//省略部分代码
        //初始化背景图片
        ImageIcon bookAddIcon=CreatecdIcon.add("newBookorderImg.jpg");
        label_5.setIcon(bookAddIcon);                          //在背景标签中添加图片
        label_5.setPreferredSize(new Dimension(400, 80));      //设置图片大小
        setVisible(true);                                      //显示窗体可关闭
    }
}
```

代码贴士

❶ ISBN：创建图书条形码文本框，通过实例化 JTextField 类。

❷ setDocument(new MyDocument(13))：调用 setDocument()方法，设置图书条形码文本框的最大长度为 13。

❸ addKeyListener()：为“图书条形码”文本框添加监听事件。

❹ addFocusListener()：为“图书条形码”文本框添加失去焦点事件。

❺ selectBookCategory()：查询图书类别表中的内容。

❻ setId()：将图书类别表中的编号放入 Item 类中的 setId()方法中。

❼ setName()：将图书类别表中的内容放入 Item 类中的 setName()方法中。

❽ bookTypeModel.addElement(item)：将 Item 类对象添加到组合框模型中。

❾ myfmt：创建 SimpleDateFormat 类对象，格式化日期格式为 yyyy-MM-dd 格式。

❿ new JFormattedTextField(myfmt.getDateInstance())：实例化 JFormattedTextField 文本框，格式化日期文本框。

（2）在图书信息添加窗体中添加按钮监听事件，在事件中的 actionPerformed()方法中进行图书信息添加操作，可以将图书信息添加方法在 Dao 类中编写。关键代码如下：

例程 24 代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
public static int Insertbook(String ISBN,String typeId,String bookname,String writer,String translator,String publisher,Date date,Double price){
    int i=0;
    try{
        //图书信息添加 SQL 语句
        String sql="insert into tb_bookInfo(ISBN,typeId,bookname,writer,translator,publisher,date,price)values('"+ISBN+"','"+typeId+"','"+bookname+"','"+writer+"','"+translator+"','"+publisher+"','"+date+"',"+price+")";
    i=Dao.executeUpdate(sql);                                  //执行 SQL 语句
    }catch(Exception e){
        System.out.println(e.getMessage());                    //显示异常字符串
    }
    Dao.close();                                               //关闭数据库连接
    return i;                                                  //将执行结果返回
}
```

在 Dao 类中编写添加图书信息操作方法后，此方法可以在按钮事件的 actionPerformed()方法中调用。关键代码如下：

例程 25 代码位置：光盘\TM\05\src\com\wsy\iframe\BookAddIFrame.java

```
class addBookActionListener implements ActionListener {        //添加按钮的单击事件监听器
    public void actionPerformed(final ActionEvent e) {
```

```
    if(ISBN.getText().length()==0){                          //判定图书条形码是否为空
            //如果为空，弹出提示对话框
            JOptionPane.showMessageDialog(null, "书号文本框不可以为空");
            return;                                          //返回，不进行以下代码操作
        }
        if(ISBN.getText().length()!=13){                     //如果图书条形码位数小于 13 位
            //弹出提示对话框
            JOptionPane.showMessageDialog(null, "书号文本框输入位数为 13 位");
            return;                                          //返回，不进行以下代码操作
        }
        if(bookName.getText().length()==0){                  //判定“书名”文本框是否为空
            //弹出提示对话框
            JOptionPane.showMessageDialog(null, "图书名称文本框不可以为空");
            return;                                          //返回，不进行以下代码操作
        }
        if(writer.getText().length()==0){                    //判定“作者”文本框是否为空
            JOptionPane.showMessageDialog(null, "作者文本框不可以为空");
            return;                                          //返回，不进行以下代码操作
        }
        if(pubDate.getText().length()==0){                   //判定“出版日期”文本框是否为空
            //弹出提示对话框
            JOptionPane.showMessageDialog(null, "出版日期文本框不可以为空");
            return;                                          //返回，不进行以下代码操作
        }
        if(price.getText().length()==0){                     //判定“单价”文本框是否为空
            //弹出提示对话框
            JOptionPane.showMessageDialog(null, "单价文本框不可以为空");
            return;                                          //返回，不进行以下代码操作
        }
        String ISBNs=ISBN.getText().trim();                  //获取“图书条形码”文本框中的内容
        Object selectedItem = bookType.getSelectedItem();    //获取图书分类组合框中的内容
        if (selectedItem == null)                            //如果组合框中没有内容
            return;                                          //返回，不进行以下代码操作
        Item item = (Item) selectedItem;                     //将组合框中内容强制转换为 Item 类
        String bookTypes=item.getId();                       //获取图书分类编号
        String translators=translator.getText().trim();      //获取“译者”文本框内容
        String bookNames=bookName.getText().trim();          //获取“书名”文本框内容
        String writers=writer.getText().trim();              //获取“作者”文本框内容
        String publishers=(String)publisher.getSelectedItem();  //在组合框中获取出版社名称
        String pubDates=pubDate.getText().trim();            //获取“出版日期”文本框内容
        String prices=price.getText().trim();                //获取“单价”文本框内容
        int i=Dao.Insertbook(ISBNs,bookTypes, bookNames, writers, translators, publishers,
java.sql.Date.valueOf(pubDates),Double.parseDouble(prices));  //执行 Dao 类中的插入图书信息操作
        if(i==1){
            //如果添加成功，弹出提示对话框
            JOptionPane.showMessageDialog(null, "添加成功");
            doDefaultCloseAction();                          //关闭当前窗口
        }
    }
}
```

（3）除了“添加”按钮监听事件外，还要控制图书条形码文本框只能输入数字字符串的键盘监听事件，在重写的 keyTyped()方法中，定义管理员允许输入的字符，如果用户输入字符与上述字符不匹配，将销毁当前输入字符。关键代码如下：

例程 26 代码位置：光盘\TM\05\src\com\wsy\iframe\BookAddIFrame.java

```
class NumberListener extends KeyAdapter {
    public void keyTyped(KeyEvent e) {                    //键盘输入监听事件
        String numStr="0123456789."+(char)8;              //定义键盘可以输入的字符，其中 char(8)是 Enter 键
        if(numStr.indexOf(e.getKeyChar())<0){             //如果当前输入字符不在定义字符串中
            e.consume();                                  //将当前输入字符销毁
        }
    }
}
```

在 BookAddIFrame.java 类中定义完成键盘监听事件后，“图书条形码”文本框可以使用如下代码调用事件：

```
ISBN.addKeyListener(new ISBNkeyListener());
```

（4）为“关闭”按钮添加按钮监听事件，主要将当前窗口关闭。关键代码如下：

例程 27 代码位置：光盘\TM\05\src\com\wsy\iframe\BookAddIFrame.java

```
class CloseActionListener implements ActionListener {    //添加“关闭”按钮的事件监听器
    public void actionPerformed(final ActionEvent e) {
        doDefaultCloseAction();                           //关闭当前窗口
    }
}
```

2．图书信息修改

图书信息修改模块的开发步骤如下：

（1）与图书信息添加窗体设计相同，图书信息修改窗体也继承了内部框架，同样在构造函数中初始化窗体属性、设计布局。关键代码如下：

例程 28 代码位置：光盘\TM\05\src\com\wsy\iframe\BookModiAndDelIFrame.java

```
public BookModiAndDelIFrame() {
    super();
    …//省略部分代码
    final JButton button = new JButton();                          //初始化“修改”按钮
    button.addActionListener(new addBookActionListener());         //为“修改”按钮添加监听事件
    …//省略部分代码
    final JButton button_1 = new JButton();                        //初始化“关闭”按钮
    button_1.addActionListener(new ActionListener() {              //为“关闭”按钮添加监听事件
        public void actionPerformed(final ActionEvent e) {         //重写 actionPerformed() 方法
            doDefaultCloseAction();                                //关闭当前窗口
        }
    });
    final JLabel headLogo = new JLabel();                          //初始化背景图片标签
```

```
        //创建背景图片
        ImageIcon bookModiAndDelIcon=CreatecdIcon.add("bookModiAndDel.jpg");
        headLogo.setIcon(bookModiAndDelIcon);                    //将背景图片添加到背景标签
        …//省略部分代码
        Object[][] results=getFileStates(Dao.selectBookInfo());  //初始化一个二维数组
        columnNames = new String[]{"图书编号", "图书类别", "图书名称", "作者", "译者", "出版商", "出版日期",
                "价格"};                                         //将表格标题放置在一个一维数组中
        table = new JTable(results,columnNames);                 //初始化表格
        table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF); //设置表格宽自动改变关闭
        //鼠标单击表格中的内容产生事件，将表格中的内容放入文本框中
        table.addMouseListener(new TableListener());
        scrollPane.setViewportView(table);                       //使表格具有滚动条
        …//省略部分代码
        setVisible(true);                                        //显示窗体可关闭
    }
```

（2）初始化窗体表格组件。首先创建为图书信息修改窗体中表格组件内容赋值的方法，此方法的参数是 List 类型的集合。在 Dao 类中创建查询图书相关信息的方法返回 List 集合可以作为此方法的参数，这个方法返回一个二维数组。关键代码如下：

例程 29　代码位置：光盘\TM\05\src\com\wsy\iframe\BookModiAndDelIFrame.java

```
private Object[][] getFileStates(List list){
    Object[][]results=new Object[list.size()][columnNames.length]; //初始化二维数组
    for(int i=0;i<list.size();i++){
        BookInfo bookinfo=(BookInfo)list.get(i);                 //将 List 集合中的数据强制转换为 BookInfo 对象
        results[i][0]=bookinfo.getISBN();                        //将图书条形码内容放入二维数组中
        results[i][1]=bookinfo.getTypeid();                      //将图书类别编号内容放入二维数组中
        results[i][2]=bookinfo.getBookname();                    //将图书名称内容放入二维数组中
        results[i][3]=bookinfo.getWriter();                      //将作者内容放入二维数组中
        results[i][4]=bookinfo.getTranslator();                  //将译者内容放入二维数组中
        results[i][5]=bookinfo.getPublisher();                   //将出版社内容放入二维数组中
        results[i][6]=bookinfo.getDate();                        //将图书出版日期内容放入二维数组中
        results[i][7]=bookinfo.getPrice();                       //将图书价格内容放入二维数组中
    }
    return results;                                              //返回二维数组
}
```

在 Dao 类中创建查询图书相关信息的方法。关键代码如下：

例程 30　代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
public static List selectBookInfo(String ISBN) {
    List list=new ArrayList();                                   //初始化 List 类型对象
    //查询图书相关信息 SQL 语句
    String sql = "select *    from tb_bookInfo where ISBN='"+ISBN+"'";
    ResultSet rs = Dao.executeQuery(sql);                        //执行 SQL 语句
    try {
        while (rs.next()) {                                      //循环查询结果集
            BookInfo bookinfo=new BookInfo();                    //创建与图书信息表字段相关的 JavaBean
```

```
                bookinfo.setISBN(rs.getString("ISBN"));    //将数据库中取得的图书条形码放入 JavaBean 中
                bookinfo.setTypeid(rs.getString("typeid")); //将数据库中取得的图书类别编号放入 JavaBean 中
                bookinfo.setBookname(rs.getString("bookname"));//将数据库中取得的图书名称放入 JavaBean 中
                bookinfo.setWriter(rs.getString("writer"));  //将数据库中取得的图书作者放入 JavaBean 中
                bookinfo.setTranslator(rs.getString("translator"));//将数据库中取得的图书译者放入 JavaBean 中
                bookinfo.setPublisher(rs.getString("publisher"));//将数据库中取得的出版社放入 JavaBean 中
                bookinfo.setDate(rs.getDate("date"));       //将数据库中取得的出版日期放入 JavaBean 中
                bookinfo.setPrice(rs.getDouble("price"));   //将数据库中取得的图书价格放入 JavaBean 中
                list.add(bookinfo);                         //将 JavaBean 对象添加到 list 集合中
            }
        } catch (Exception e) {
            e.printStackTrace();                            //捕捉异常
        }
        Dao.close();                                        //关闭数据库连接
        return list;
    }
```

创建完成窗体内表格组件后，需要为表格组件添加鼠标监听事件，以便用户单击表格中某一行记录后，相应地将表格中的数据放置在文本框中。关键代码如下：

例程 31 代码位置：光盘\TM\05\src\com\wsy\iframe\BookModiAndDelIFrame.java

```
class TableListener extends MouseAdapter {
    public void mouseClicked(final MouseEvent e) {
        String ISBNs, typeids, bookNames,writers,translators,publishers,dates,prices;
        //返回第一个选定行的索引，如果没有选定，返回-1
        int selRow = table.getSelectedRow();
        ISBNs = table.getValueAt(selRow, 0).toString().trim();      //获取选定行第一列的值
        typeids = table.getValueAt(selRow, 1).toString().trim();    //获取选定行第二列的值
        bookNames = table.getValueAt(selRow, 2).toString().trim();  //获取选定行第三列的值
        writers = table.getValueAt(selRow, 3).toString().trim();    //获取选定行第四列的值
        translators = table.getValueAt(selRow, 4).toString().trim(); //获取选定行第五列的值
        publishers = table.getValueAt(selRow, 5).toString().trim(); //获取选定行第六列的值
        dates = table.getValueAt(selRow, 6).toString().trim();      //获取选定行第七列的值
        prices = table.getValueAt(selRow, 7).toString().trim();     //获取选定行第八列的值
        ISBN.setText(ISBNs);                                        //将表格第一列的值放入图书编号文本框
❶       bookTypeModel.setSelectedItem(map.get(typeids));            //将图书类别名称内容放入组合框中
        bookName.setText(bookNames);                                //将图书名称信息放入文本框中
        writer.setText(writers);                                    //将作者信息放入文本框中
        translator.setText(translators);                            //将译者信息放入文本框中
        publisher.setText(publishers);                              //将出版社信息放入文本框中
        pubDate.setText(dates);                                     //将出版日期信息放入文本框中
        price.setText(prices);                                      //将图书价格信息放入文本框中
    }
}
```

代码贴士

❶ map.get(typeids)：将图书类别编号与图书类别名称放入 Map 数据结构中，所以此处根据图书类别编号返回图书类别名称。

（3）为“修改”按钮添加按钮监听事件。首先需要在 Dao 类中创建修改图书相关信息的方法。关键代码如下：

例程 32　代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
public static int Insertbook(String ISBN,String typeId,String bookname,String writer,String translator,String
publisher,Date date,Double price){
    int i=0;
    try{
        //创建数据库插入语句
        String sql="insert into tb_bookInfo(ISBN,typeId,bookname,writer,translator,publisher,date,price)
values('"+ISBN+"','"+typeId+"','"+bookname+"','"+writer+"','"+translator+"','"+publisher+"','"+date+"','"+price+"')";
        i=Dao.executeUpdate(sql);                          //执行数据库插入语句
    }catch(Exception e){
        System.out.println(e.getMessage());                //将捕捉异常打印
    }
    Dao.close();                                           //关闭数据库连接
    return i;                                              //将执行结果返回
}
```

在 BookModiAndDelIFrame 类中为“修改”按钮添加按钮监听事件，实现 ActionListener 接口中的 actionPerformed()方法，在这个方法中不仅需要调用 Dao 类中的图书修改方法，还要限制所有文本框字符串的非法输入，同时为了使图书信息表修改完成后，在窗体中的表格即时显示修改内容，需要将表格模型重新赋予表格中。关键代码如下：

例程 33　代码位置：光盘\TM\05\src\com\wsy\iframe\BookModiAndDelIFrame.java

```
class addBookActionListener implements ActionListener {
    public void actionPerformed(final ActionEvent e) {
        //修改图书信息表
        if(ISBN.getText().length()==0){                    //如果图书条形码为空
            //弹出提示对话框
            JOptionPane.showMessageDialog(null, "书号文本框不可以为空或者输入数字不可以大于 13 个");
            return;
        }
        …//部分代码省略
        int i=Dao.Updatebook(ISBNs, bookTypes, bookNames, writers, translators, publishers,
Date.valueOf(pubDates), Double.parseDouble(prices));       //执行图书相关信息修改操作
        if(i==1){                                          //如果修改成功
            JOptionPane.showMessageDialog(null, "修改成功");   //弹出修改成功对话框
            Object[][] results=getFileStates(Dao.selectBookInfo());  //初始化二维数组
            table.setModel(model);                         //将模型重新赋予表格
            model.setDataVector(results, columnNames);     //将修改完成的数据放入模型中
        }
    }
}
```

5.8.4 单元测试

图书条形码作为图书相关信息表的主键，在图书添加的过程中由管理员输入，如果管理员输入的图书条形码在数据库中已经存在，根据主键规则，会发生以下异常，如图 5.21 所示。

图 5.21　主键重复引发异常

这时需要为“书号”文本框添加焦点事件，重写 focusLost()方法，此方法为失去焦点方法，可以在此方法中判断用户输入的字符串是否与数据库中的图书条形码重复，实质上就是在用户鼠标焦点离开图书条形码文本框时进行判断。判断操作可以在 Dao 类中创建数据库查询方法。关键代码如下：

例程 34　代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
public static List selectBookInfo(String ISBN) {
    List list=new ArrayList();                                          //初始化 List 类型集合
    String sql = "select *  from tb_bookInfo where ISBN='"+ISBN+"'";    //创建 SQL 语句
    ResultSet rs = Dao.executeQuery(sql);                               //执行 SQL 语句
    try {
        while (rs.next()) {                                             //循环结果集
            BookInfo bookinfo=new BookInfo();                           //初始化 BookInfo 对象
            //将数据库中获取内容相应地放入 BookInfo 中
            …//部分代码省略
            list.add(bookinfo);                                         //将 BookInfo 对象添加到 list 集合中
        }
    } catch (Exception e) {
        e.printStackTrace();                                            //捕捉异常
    }
    Dao.close();                                                        //关闭数据库连接
    return list;                                                        //返回集合
}
```

然后在焦点事件中的 focusLost()方法中查询用户输入的条形码是否与数据库中的数据重复，如果重复则弹出相应提示对话框。关键代码如下：

例程 35　代码位置：光盘\TM\05\src\com\wsy\iframe\BookAddIFrame.java

```
class ISBNFocusListener extends FocusAdapter {
    public void focusLost(FocusEvent e){
        if(!Dao.selectBookInfo(ISBN.getText().trim()).isEmpty()){//如果用户输入的字符串与数据库中的值重复
            JOptionPane.showMessageDialog(null, "添加书号重复！");      //弹出提示对话框
            return;                                                     //返回，不执行以下代码
        }
    }
}
```

5.9　图书借阅、归还模块设计

5.9.1　图书借阅、归还模块概述

图书借阅模块主要用于管理读者借阅图书的信息。管理员输入读者条形码、图书条形码后，在读者相关信息文本框以及图书相关信息文本框中相应显示此读者和书籍的相关内容，这时在窗体表格组件中显示读者信息、图书信息以及借书日期、还书日期等相关字段，当管理员单击“借出当前书籍”按钮，此读者与图书将被存放到借阅表中。

图书归还模块主要实现读者还书功能。当读者需要还书时，管理员输入读者条形码，按 Enter 键，在窗体表格中显示读者借阅图书的相关信息，在表格中单击某一行数据，在罚款相关文本框中会显示相应的内容等，最后管理员单击“图书归还”按钮，完成图书归还操作。

图书借阅模块的运行效果如图 5.22 所示，图书归还模块的运行效果如图 5.23 所示。

图 5.22　图书借阅管理窗体

图 5.23　图书归还管理窗体

5.9.2　图书借阅、归还模块技术分析

图书借阅模块主要用到了键盘监听事件的 keyTyped()方法，重写此方法，使用 KeyEvent 类中的 getKeyChar()方法获取当前按键的键值，如果是 Enter 键，查询此读者的相关内容，同理可以查询图书的相关内容。例如：

```
class ISBNListenerlostFocus extends KeyAdapter {
    public void keyTyped(KeyEvent e) {
        if (e.getKeyChar() == '\n') {                    //判断是否按 Enter 键
        …//进行相关操作
    }
}
```

获取读者与图书的相关信息后，在“借出当前图书”按钮监听事件中将相关信息插入图书借阅表中。这里存在一个难点，就是如何获取应还日期。在图书类别表中可以获取此类图书允许借阅的天数，将当前借阅时间加上此类图书允许借阅的天数，即可得到图书应还的时间。在这里使用 Date 类中的 getDate()方法获取当前时间是某月的某一天，然后将此数值与此类图书允许借阅的天数求和后再使用 setDate()方法获取应还时间。例如：

```
java.util.Date date = new java.util.Date();
date.setDate(date.getDate() + Integer.parseInt(days));
```

图书借阅模块窗体中设置了一个清除表格内容的按钮——“清除所有记录”，在此按钮的监听事件中，可以使用如下代码将表格中的内容删除：

```
model.removeRow(table.getRowCount()-1);
```

其中，table.getRowCount()是获取表格行数的方法。

在图书借阅模块窗体中还设置了一个显示当前时间的文本框，文本框中的时间是动态变化的。这里使用了 Timer 类机制，将 Timer 对象放入文本框的监听事件中即可。例如：

```
class TimeActionListener implements ActionListener{
    public TimeActionListener(){
        Timer t=new Timer(1000,this);
        t.start();
    }
}
```

图书归还模块设计与图书借阅模块设计原理基本相同，唯一不同的是按钮事件实现的功能不同，图书归还模块主要是在“图书归还”按钮事件中将图书借阅表中的“是否归还”字段设置为 0，0 代表借阅图书已经归还，1 代表借阅图书没有归还，表中“是否归还”字段默认值为 1。

由于图书归还窗体中设计了显示图书借阅相关信息的表格，表格内容需要查询多表字段内容，所以需要应用内联接机制进行查询。

内联接用于返回所有连接表中具有匹配值的行，而排除所有其他的行。其语法格式如下：

```
SELECT fieldlist
FROM    table1 [INNER] JOIN table2
ON table1.column=table2.column
```

5.9.3 图书借阅、归还模块的实现过程

1. 图书借阅

开发图书借阅模块的步骤如下：

（1）在类构造函数中创建窗体布局以及相关属性。关键代码如下：

例程 36　代码位置：光盘\TM\05\src\com\wsy\iframe\BookBorrowIFrame.java

```
public class BookBorrowIFrame extends JInternalFrame {
    public BookBorrowIFrame() {
        …//省略部分代码
❶       table = new JTable();                                          //初始化表格
❷       table.setModel(model);                                         //将模型添加到表格中
        …//省略部分代码
        readerISBN = new JTextField();                                 //初始化“读者条形码”文本框
        readerISBN.setDocument(new MyDocument(13));        //使“读者条形码”文本框只能输入 13 位
❸       readerISBN.addKeyListener(new ISBNListenerlostFocus());  //为“读者条形码”文本框添加监听事件
        …//省略部分代码
        final JButton buttonBorrow = new JButton();                    //初始化按钮
        buttonBorrow.setText("借出当前图书");
        buttonBorrow.addActionListener(new BorrowActionListener()); //为按钮添加事件
        final JButton buttonClear = new JButton();                     //初始化按钮
        buttonClear.setText("清除所有记录");
        buttonClear.addActionListener(new ClearActionListener(model));     //为按钮添加事件
        setVisible(true);
}
```

代码贴士

❶ table：实例化 table 类创建表格。

❷ table.setModel(model)：将模型赋予到表格中。

❸ addKeyListener()：为“读者条形码”文本框添加监听事件。

（2）为“读者条形码”文本框添加键盘监听事件。当用户输入读者条形码，按 Enter 键后，触发“读者条形码”文本框键盘监听事件。在 keyTyped()方法中，调用 Dao 类中的查询读者相关信息方法，如果在数据库中没有查询到相关信息，弹出相应提示对话框；如果查询到结果，最后将查询结果放入相应文本框中。关键代码如下：

例程 37　代码位置：光盘\TM\05\src\com\wsy\iframe\BookBorrowIFrame.java

```
class ISBNListenerlostFocus extends KeyAdapter {
    public void keyTyped(KeyEvent e) {
        if (e.getKeyChar() == '\n') {                          //判断是否按 Enter 键
            String ISBNs = readerISBN.getText().trim();        //获取读者条形码
            List list = Dao.selectReader(ISBNs);               //查询读者信息
            if (list.isEmpty() && !ISBNs.isEmpty()) {          //如果没有查询到结果
                JOptionPane.showMessageDialog(null,            //弹出相应对话框
                        "此读者编号没有注册，查询输入读者编号是否有误！");
            }
            for (int i = 0; i < list.size(); i++) {            //循环结果集
                Reader reader = (Reader) list.get(i);          //将结果放入与数据库相对应的 JavaBean 中
                readerName.setText(reader.getName());          //将读者名称放入 JavaBean 中
                number.setText(reader.getMaxNum());            //将读者最大借书量放入 JavaBean 中
                //将读者押金放入 JavaBean 中
```

```
                keepMoney.setText(reader.getKeepMoney() + "");
            }
        }
    }
}
```

（3）同理，在“图书条形码”文本框键盘监听事件中获取“图书条形码”文本框内容，调用 Dao 类中的查询图书相关信息的方法，将图书信息放入相应的文本框中，同时需要将读者信息、图书信息、还书时间、借书时间放入表格中。关键代码如下：

例程 38 代码位置：光盘\TM\05\src\com\wsy\iframe\BookBorrowIFrame.java

```
class bookISBNListenerlostFocus extends KeyAdapter {
    public void keyTyped(KeyEvent e) {
        if (e.getKeyChar() == '\n') {                          //判断是否按 Enter 键
            String ISBNs = bookISBN.getText().trim();          //获取图书条形码
            List list = Dao.selectBookInfo(ISBNs);             //查询图书相关信息
            for (int i = 0; i < list.size(); i++) {            //循环结果集
                BookInfo book = (BookInfo) list.get(i);        //将结果放入数据库相对应的 JavaBean 中
                bookName.setText(book.getBookname());          //将图书名称放入 JavaBean 中
                //将图书类别编号放入 JavaBean 中
                bookType.setText(String.valueOf(map.get(book.getTypeid())));
                price.setText(String.valueOf(book.getPrice())); //将图书价格放入 JavaBean 中
            }
            String days = "0";
            //查询图书类别表的相关内容
            List list2 = Dao.selectBookCategory(bookType.getText().trim());
            for (int j = 0; j < list2.size(); j++) {           //循环结果集
                BookType type = (BookType) list2.get(j);       //将结果放入数据库相对应的 JavaBean 中
                days = type.getDays();                         //获取本类图书允许借阅的时间
            }
            String readerISBNs = readerISBN.getText().trim();
            List list5 = Dao.selectReader(readerISBNs);        //查询此读者是否在 tb_reader 表中
            List list4 = Dao.selectBookInfo(ISBNs);            //查询此书是否在 tb_bookInfo 表中
            if (!readerISBNs.isEmpty() && list5.isEmpty()) {   //如果没有查询出结果，弹出相应对话框
                JOptionPane.showMessageDialog(null,
                        "此读者编号没有注册，查询输入读者编号是否有误！");
                return;                                        //返回，不再执行以下代码
            }
            if (list4.isEmpty() && ! ISBNs.isEmpty()) {        //如果没有查询出结果，弹出相应对话框
                JOptionPane.showMessageDialog(null,
                        "本图书馆没有此书，查询输入图书编号是否有误！");
                return;                                        //返回，不再执行以下代码
            }
            if (Integer.parseInt(number.getText().trim()) <= 0) {  //如果统计借书量的数字为 0
                //弹出超过最大借书量对话框
                JOptionPane.showMessageDialog(null, "借书量已经超过最大借书量！");
                return;
            }
```

```
                add();                                                    //调用添加表格行方法
                number.setText(String.valueOf(Integer.parseInt(number.getText()
                        .trim()) - 1));                                   //表格每次添加一行，可借书数量减 1
            }
        }
}
```

（4）在 BookBorrowIFrame 类中创建一个表格行添加的方法 add()，在“图书条形码”文本框键盘监听事件中调用，实现在管理员输入完图书条形码后，按 Enter 键，在窗体表格中添加一行数据的功能。在 add()方法中，将图书条形码、读者条形码、当前时间、应还时间放入数组中，最后将数组添加到表格模型中作为表格新增的一行数据。关键代码如下：

例程 39　代码位置：光盘\TM\05\src\com\wsy\iframe\BookBorrowIFrame.java

```
public final void add() {
    String str[] = new String[4];                                         //初始化数组
    str[0] = bookISBN.getText().trim();                                   //获取图书条形码
    str[1] = String.valueOf(myfmt.format(new java.util.Date()));          //获取当前时间
    str[2] = getBackTime().toLocaleString();                              //获取归还时间
    str[3] = readerISBN.getText().trim();                                 //获取读者条形码
    model.addRow(str);                                                    //将数组添加到表格模型中
}
```

（5）这里需要创建取得应还时间的方法。在 Dao 类中定义一个取得当前书籍允许借阅时间的方法，取得当前书籍允许借阅的最大天数，取当前时间与此天数的加和，即可返回应还时间。关键代码如下：

例程 40　代码位置：光盘\TM\05\src\com\wsy\iframe\BookBorrowIFrame.java

```
public Date getBackTime() {                                               //获取还书时间
    String days = "0";
    List list2 = Dao.selectBookCategory(bookType.getText().trim());       //查询图书类别方法
    for (int j = 0; j < list2.size(); j++) {
        BookType type = (BookType) list2.get(j);
        days = type.getDays();                                            //获取当前图书允许借阅的天数
    }
    java.util.Date date = new java.util.Date();
    date.setDate(date.getDate() + Integer.parseInt(days));                //获取图书应还时间
    return date;                                                          //返回书籍应还时间
}
```

（6）最后在“借阅当前图书”按钮监听事件中，将相关信息存入图书借阅表中，如果操作成功，提示相应对话框。关键代码如下：

例程 41　代码位置：光盘\TM\05\src\com\wsy\iframe\BookBorrowIFrame.java

```
class BorrowActionListener implements ActionListener {
    public void actionPerformed(final ActionEvent e) {
        …//省略部分代码
        int i=Dao.InsertBookBorrow(bookISBNs, readerISBNs, operatorId,
java.sql.Timestamp.valueOf(borrowDate), java.sql.Timestamp.valueOf(backDate)) //将借阅相关信息插入图书借阅表中
        if(i==1){                                                         //如果插入成功
```

```
                //弹出相应对话框
                JOptionPane.showMessageDialog(null, "图书借阅完成！");
                doDefaultCloseAction();                                    //关闭当前窗口
            }
        }
}
```

2．图书归还

实现图书归还模块的具体步骤如下：

（1）需要实现管理员输入读者条形码后，在窗体表格中显示相关内容的查询方法。这个方法与其他数据库操作方法相同，同样在 Dao 类中进行定义。由于需要查询的内容不在数据库中同一数据表中，所以需要利用表关系进行内联接查询。其中用到的表包括 tb_borrow（图书借阅信息表）、tb_reader（读者信息表）、tb_bookInfo（图书信息表），这 3 个表的关系如图 5.24 所示。

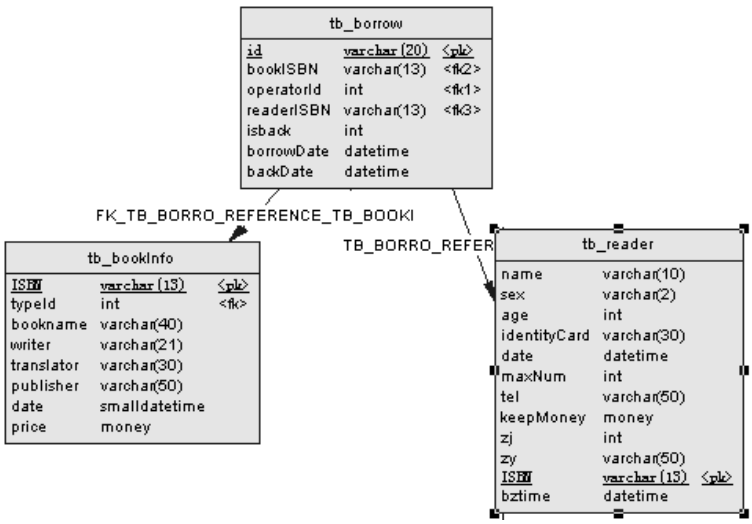


图 5.24　图书借阅表、图书信息表、读者信息表关系图

从图 5.24 中可以看出，图书信息表与读者信息表中的主键是图书借阅表中的两个外键，如果需要取得这 3 个表中的字段内容，可以使用外键关系进行查询。关键代码如下：

例程 42　代码位置：光盘\TM\05\src\com\wsy\iframe\BookBackIFrame.java

```
public static List selectBookBack(String readerISBN) {
    List list=new ArrayList();                                    //初始化 list 集合
❶   String sql = "SELECT a.ISBN AS bookISBN, a.bookname, a.typeId ,b.operatorId, b.borrowDate,
b.backDate, c.name AS readerName, c.ISBN AS readerISBN FROM tb_bookInfo a INNER JOIN tb_borrow
b ON a.ISBN = b.bookISBN INNER JOIN tb_reader c ON b.readerISBN = c.ISBN WHERE (c.ISBN =
'"+readerISBN+"' and isback=1)";
    ResultSet rs = Dao.executeQuery(sql);                         //执行 SQL 语句
    try {
        while (rs.next()) {                                       //循环结果集
            Back back=new Back();                                 //初始化 Back 类对象
            back.setBookISBN(rs.getString("bookISBN"));           //将查询内容放入 Back 类中
            …//省略部分代码
            list.add(back);                                       //将 Back 对象添加到 list 集合中
        }
    } catch (Exception e) {
        e.printStackTrace();                                      //捕捉异常
    }
```

```
    Dao.close();                                                    //关闭数据库连接
    return list;                                                    //返回结果集
}
```

代码贴士

❶ 将 tb_bookInfo 取别名为 a，将 tb_borrow 取别名为 b，将 tb_reader 取别名为 c，首先使 a 表与 b 表进行内联接操作，取满足条件 a.ISBN=b.bookISBN 的所有记录，然后此记录再与 c 表进行内联接操作，取满足条件 c.ISBN=b.readerISBN 的所有记录，并且在 c 表中取当前用户没有归还书籍的相关记录。

定义完成上述方法后，可以在 BookBackIFrame 类中定义一个创建表格的 add()方法，在此方法中调用 Dao 类中的查询方法，然后在“读者条形码”文本框中添加监听事件，在 actionPerformed()方法中调用 add()方法，实现将查询结果添加到表格中的功能。关键代码如下：

例程 43　代码位置：光盘\TM\05\src\com\wsy\iframe\BookBackIFrame.java

```
public final void add() {
    System.out.println("testadd");
    String readerISBNs=readerISBN.getText().trim();                 //获取读者条形码
    List list=Dao.selectBookBack(readerISBNs);                      //调用 Dao 类查询方法
    for(int i=0;i<list.size();i++){                                 //循环结果集
        Back back=(Back)list.get(i);                                //将结果放入 Back 类中
        String str[] = new String[7];                               //初始化一维数组
        str[0] =back.getBookname();                                 //将图书名称的值赋予到数组中
        str[1] =back.getBookISBN();                                 //将图书条形码赋予到数组中
❶       str[2]=String.valueOf(MapPz.getMap().get(back.getTypeId()+""));  //将图书类别赋予到数组中
        str[3] =back.getReaderName();                               //将读者姓名赋予到数组中
        str[4] =back.getReaderISBN();                               //将读者条形码赋予到数组中
        str[5] =back.getBorrowDate();                               //将读者借书时间赋予到数组中
        str[6]=back.getBackDate();                                  //将读者还书时间赋予到数组中
        model.addRow(str);                                          //将数组添加到表格模型中
    }
}
```

代码贴士

❶ MapPz.getMap().get(back.getTypeId()+"")：在数据库中取得图书类别编号，可以使用 MapPz.get()方法取得图书类别名称。

（2）在设计窗体时，需要实现用户单击表格中的某一行，在相应文本框中显示此书借阅的罚款信息。可以设置表格的鼠标监听事件，在 mouseClicked()方法中实现上述操作。关键代码如下：

例程 44　代码位置：光盘\TM\05\src\com\wsy\iframe\BookBackIFrame.java

```
class TableListener extends MouseAdapter {
    public void mouseClicked(final MouseEvent e) {
        java.util.Date date=new java.util.Date();
        String fk="";
        String days1="";
        int selRow=table.getSelectedRow();                          //取得当前表格选中的行
        //获取图书类别表中的罚款信息
        List list =Dao.selectBookTypeFk(table.getValueAt(selRow, 2).toString().trim());
```

```
            for(int i=0;i<list.size();i++){                          //循环查询结果集
                BookType booktype=(BookType)list.get(i);             //将查询结果放入 BookType 类中
                fk=booktype.getFk();                                 //取得罚款信息
                days1=booktype.getDays();                            //取得此类书籍允许的借阅天数
            }
            //将借阅日期放入相应的文本框中
            borrowDate.setText(table.getValueAt(selRow, 5).toString().trim());
❶           int days2,days3;
            borrowdays.setText(days1+"");                    //将此类书籍允许的借阅天数放入相应的文本框中
❷           days2=date.getDate()-java.sql.Timestamp.valueOf(table.getValueAt(selRow,  5).toString().trim()).
getDate();
            realdays.setText(days2+"");
❸           days3=days2-Integer.parseInt(days1);
            if(days3>0){                                             //如果读者借书超出规定时间
                ccdays.setText(days3+"");                            //将超出的规定时间放入相应的文本框中
                Double zfk=Double.valueOf(fk)*days3;                 //计算读者应交的罚款金额
                fkmoney.setText(zfk+"元");                           //将罚款金额放入相应的文本框中
            }
            else{                                                    //如果读者借书没有超过规定时间
                ccdays.setText("没有超过规定天数");                  //将指定信息放入到罚款金额文本框中
            }
        }
}
```

代码贴士

❶ days2：实际借阅天数。days3：超出规定天数。

❷ 获取当前时间与读者归还时间的差即为实际借阅天数。

❸ 获取实际借阅天数与规定天数的差即为超出规定天数。

（3）为“图书归还”按钮添加监听事件，图书归还操作主要是将图书借阅表中的“是否归还”字段内容设置为 0，此操作可以在 Dao 类中完成，然后在监听事件的 actionPerformed()方法中调用。关键代码如下：

例程 45 代码位置：光盘\TM\05\src\com\wsy\iframe\BookBackIFrame.java

```
class BookBackActionListener implements ActionListener{
    public void actionPerformed(ActionEvent e) {
        if(bookISBNs.isEmpty()||readerISBNs.isEmpty()){      //如果表格中没有借阅的相关信息
            JOptionPane.showMessageDialog(null, "请选择所要归还的图书！"); //弹出相应对话框
            return;                                                  //返回，不进行以下代码
        }
        int i=Dao.UpdateBookBack(bookISBNs, readerISBNs);        //执行图书归还操作
        if(i==1){                                                    //如果执行成功
            JOptionPane.showMessageDialog(null, "还书操作完成！");  //弹出相应的对话框
            int selectedRow = table.getSelectedRow();                //取表格当前行
            model.removeRow(selectedRow);                            //在模型中将此行删除
        }
    }
}
```

5.9.4　单元测试

在图书借阅模块开发过程中，笔者在“读者条形码”文本框与“图书条形码”文本框中添加的是焦点监听事件，在 focusLost()方法中查询读者相关信息与图书相关信息，然后在“图书条形码”文本框的 focusLost()方法中设置将图书相关信息放入表格中。这样的设置存在一个弊端，当管理员向“图书条形码”文本框中输入字符串后，引发“图书条形码”文本框失去焦点事件，此时会将相关内容放入窗体表格中进行显示，由于管理员并没有向“读者条形码”文本框中输入任何字符串，所以在窗体表格中没有读者相关信息显示。同时，使用焦点监听事件也会误导管理员操作系统。

当管理员直接在“图书条形码”文本框中输入图书条形码后，没有向“读者条形码”文本框输入字符串，而在触发文本框失去焦点事件时，就会出现如图 5.25 所示的异常。

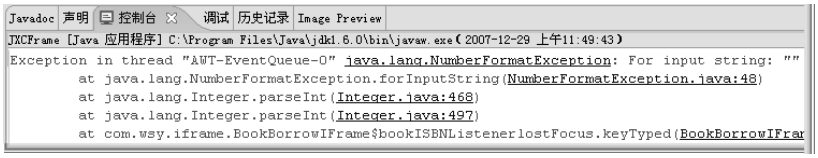


图 5.25　异常提示

在这种情况下，为了避免出现上述问题，需要在触发“图书条形码”文本框失去焦点事件时，首先判断管理员是否在“读者条形码”文本框中输入字符串，如果“读者条形码”文本框为空，则弹出提示对话框。同时为了避免误导管理操作本系统，将失去焦点事件改为键盘按 Enter 键事件。关键代码如下：

例程 46　代码位置：光盘\TM\05\src\com\wsy\iframe\BookBorrowIFrame.java

```
class bookISBNListenerlostFocus extends KeyAdapter {
    public void keyTyped(KeyEvent e) {
        if (e.getKeyChar() == '\n') {                                    //判断是否按 Enter 键
            if (readerISBN.getText().trim().length()!=0 //如果“读者条形码”和“图书条形码”文本框不为空
                    && bookISBN.getText().trim().length()!=0) {
                String ISBNs = bookISBN.getText().trim();
                List list = Dao.selectBookInfo(ISBNs);                   //进行查询操作
                …//省略部分代码
                else                                                     //如果查询为空
                    JOptionPane.showMessageDialog(null, "请输入读者条形码！");
            }
        }
    }
}
```

5.10　图书查询模块设计

5.10.1　图书查询模块概述

图书查询窗体主要包括条件查询与全部查询。窗体整个布局使用 BorderLayout 布局管理器，在窗

体中部放置了 JTabbedPane 组件，分别在 JTabbedPane 组件的两个标签中放置了一个面板，一个面板用于放置条件查询结果集，另一个面板用于放置查询全部图书信息的结果集。在条件查询面板中，用户可以在组合框中选择需要查询的字段，然后在条件文本框中输入需要查询的字符串；在全部查询面板中，用户选择“显示图书全部信息”选项卡，即可查看所有图书相关信息。图书查询模块运行效果如图 5.26 所示。

图 5.26　图书查询窗体

5.10.2　图书查询模块技术分析

图书查询窗体主要包括按条件查询功能与全部图书查询功能，这时需要在窗体中使用选项卡，可以使用 JTabbedPane 组件。

在窗体中使用 JTabbedPane 组件需要初始化 JTabbedPane 组件，可以使用如下代码：

```
final JTabbedPane tabbedPane = new JTabbedPane();
tabbedPane.setPreferredSize(new Dimension(0, 50));
getContentPane().add(tabbedPane);
```

然后在选项卡中添加选项卡名称，例如：

```
tabbedPane.addTab("条件查询", null, panel_1, null);
```

图书查询主要实现将查询结果集放入表格中，所以需要初始化表格。

初始化表格首先是将表头信息放入数组中，分别为图书名称、类别、作者等相关信息。然后初始化表格数据，可以在 Dao 类中将图书相关信息查询出来，放入二维数组中。其中，查询包括全部信息查询与条件查询。在这里条件查询使用模糊查询，在 SQL Server 2000 中实现模糊查询可以使用带有 like 关键字的 SQL 语句，使用 like 的 SQL 语句可以确定给定的字符串是否与指定的模式匹配。模式可以包含常规字符和通配符。模式匹配过程中，常规字符必须与字符串中指定的字符完全匹配。然而，可使用字符串的任意片段匹配通配符。与使用“=”和“！=”字符串比较运算符相比，使用通配符可使 like 运算符更加灵活。可以使用如下 SQL 语句对图书信息表进行模糊查询：

```
select * from tb_bookInfo where bookname like '%computer%'
```

在 Dao 类查询图书相关信息后，可以将数据放入二维数组中，然后使用表头数组与表文数组初始化表格。Swing 中的表格初始化有多种形式。以下是 JTable 类的所有构造函数：

- ☑ JTable()。
- ☑ JTable(int numRows,int numColumns)。
- ☑ JTable(Object[][] rowData,Object[] columnNames)。
- ☑ JTable(TableModel dm)。
- ☑ JTable(TableModel dm,TableColumnModel cm)。
- ☑ JTable(TableModel dm,TableColumnModel cm,ListSelectionModel sm)。

☑ JTable(Vector rowData,Vector columnNames)。

在这里笔者使用第 3 个构造函数创建表格，其中第一个参数是表格内容，第二个参数是表头。

5.10.3　图书查询模块的实现过程

图书查询模块的实现步骤如下：

（1）在 Dao 类中定义两个查询方法，分别为条件查询与全部查询，其中条件查询使用了模糊查询机制，查询完毕后将查询结果放入 JavaBean 中，然后将 JavaBean 对象添加到 list 中。关键代码如下：

例程 47　代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
public static List selectbookserch() {
    List list=new ArrayList();                                  //初始化 list 集合
❶   String sql = "select *    from tb_bookInfo";                //查询全部图书相关信息
❷   ResultSet s = Dao.executeQuery(sql);                        //执行 SQL 语句
    try {
        while (s.next()){                                       //循环结果集
            BookInfo bookinfo=new BookInfo();                   //初始化 BookInfo 对象
            …//省略部分代码
            list.add(bookinfo);                                 //将 BookInfo 对象添加到结果集
        }
    } catch (Exception e) {
        e.printStackTrace();                                    //抛出异常
    }
❸   Dao.close();                                                //关闭数据库连接
    return list;                                                //将 list 集合返回
}
```

代码贴士

❶ sql：查询图书信息表的全部内容。

❷ Dao.executeQuery(sql)：调用 Dao 类中的 executeQuery()方法，执行 SQL 语句，返回结果集。

❸ Dao.close()：调用 Dao 类中的 close()方法，关闭数据库连接。

在 Dao 类中定义条件查询的代码如下：

例程 48　代码位置：光盘\TM\05\src\com\wsy\dao\Dao.java

```
public static List selectbookmohu(String bookname){
    List list=new ArrayList();                                  //初始化 list 集合
    //模糊查询 SQL 语句
❶   String sql="select * from tb_bookInfo where bookname like '%"+bookname+"%'";
❷   ResultSet s=Dao.executeQuery(sql);                         //执行 SQL 语句
    try {
        while(s.next()){                                        //循环结果集
            BookInfo bookinfo=new BookInfo();
            …//省略部分代码
            list.add(bookinfo);                                 //将对象添加到 list 集合中
        }
    } catch (SQLException e) {
```

```
            e.printStackTrace();                                    //捕捉抛出异常
        }
❸       Dao.close();
❹       return list;                                                //将结果集返回
}
```

代码贴士

❶ sql：使用模糊查询语句查询图书信息表的全部内容。

❷ Dao.executeQuery(sql)：调用 Dao 类中的 executeQuery()方法，执行 SQL 语句，返回结果集。

❸ Dao.close()：调用 Dao 类中的 close()方法，关闭数据库连接。

❹ return list：将结果以集合的形式返回。

（2）在 BookSearchIFrame 类中创建表格。首先在一维数组中定义表头，然后在二维数组中定义表格内容，在定义表格内容过程中可以调用步骤（1）中提到的 Dao 类中的查询图书相关信息方法。关键代码如下：

例程 49 代码位置：光盘\TM\05\src\com\wsy\iframe\BookSearchIFrame.java

```
//定义表格头部
String booksearch[] = { "编号", "分类", "名称", "作者", "出版社", "译者", "出版日期", "单价" };
//创建获取表文的方法，返回二维数组
private Object[][] getselect(List list) {
    Object[][] s = new Object[list.size()][8];                      //初始化二维数组
    for (int i = 0; i < list.size(); i++) {                         //循环结果集
        BookInfo book = (BookInfo) list.get(i);                     //将结果集内容放入 BookInfo 对象中
        s[i][0] = book.getISBN();                                   //将图书相关信息放入二维数组中
    …//省略部分代码
    }
    return s;                                                       //将二维数组返回
}
```

初始化表格的表头与表文后，可以在窗体中创建表格。关键代码如下：

例程 50 代码位置：光盘\TM\05\src\com\wsy\iframe\BookSearchIFrame.java

```
table_2 = new JTable(results,booksearch);
```

（3）在“查询”按钮中添加监听事件，重写 actionPerformed()方法，在此方法中调用 Dao 类中的查询方法。关键代码如下：

例程 51 代码位置：光盘\TM\05\src\com\wsy\iframe\BookSearchIFrame.java

```
class SearchListener implements ActionListener {
    public void actionPerformed(ActionEvent arg0) {
        String name=(String)choice.getSelectedItem();               //获取组合框中选择的查询字段
        if(name.equals("图书名称")){                                 //如果管理员选择“图书名称”
            //以图书名称为条件进行查询
            Object[][] results=getselect(Dao.selectbookmohu(textField_1.getText()));
            table_2 = new JTable(results,booksearch);               //初始化表格
            scrollPane_1.setViewportView(table_2);                  //将表格添加到滚动条中
```

```
            }
            else if(name.equals("图书作者")){                    //如果管理员选择“图书作者”
                //以图书作者为条件进行查询
                Object[][] results=getselect(Dao.selectbookmohuwriter(textField_1.getText()));
                table_2 = new JTable(results,booksearch);        //初始化表格
                scrollPane_1.setViewportView(table_2);           //将表格添加到滚动条中
            }
        }
}
```

5.11　开发技巧与难点分析

5.11.1　级联删除

在本系统数据库建模时，将读者信息表与图书借阅表设置了关联关系，读者条形码作为图书借阅表的外键，在读者信息管理模块中设计了读者删除功能，如果此读者借阅图书，系统会将此读者条形码保存在图书借阅表中作为此表的外键，如果此时在读者信息管理模块中删除此读者，将会抛出异常，如图 5.27 所示。

图 5.27　异常提示

这时需要在 SQL Server 2000 中对读者信息表进行级联删除设置。

级联删除设置步骤如下：

（1）打开 SQL Server 2000 的企业资源管理器，右击 db_library 数据库中的 tb_reader 表，在弹出的快捷菜单中选择“设计表”命令，弹出“设计表”窗口，在此窗口的工具栏上单击按钮，弹出“属性”窗口，选中“级联更新相关字段”和“级联删除相关记录”复选框。

（2）对 tb_reader 表设置完级联删除后，可以进行测试。首先设置读者完成借阅图书操作，这时在 tb_borrow 表中相应地插入了此读者借阅图书的相关信息，当在读者信息修改与删除窗体中删除此读者的相关信息，不会再发生异常，同时 tb_borrow 表中的有关此读者借阅图书的相关信息也被删除。

5.11.2　窗体中单选按钮即时显示

在图书验收模块开发中，进入窗体后，首先将图书订购信息显示在表格中，当管理员单击表格中某一行数据后，在窗体相应文本框中即时显示相关内容。这里存在一个难点，就是“是否验收”单选按钮的即时显示。

Swing 中单选按钮组件被选择的语句如下：

```
radioButton.setSelected(true);
```

当从表格中获取“是否验收”字段内容时，在表格鼠标监听事件中根据字段内容判断应该使哪个

单选按钮被选中，其中“是否验收”字段内容为 0 时代表图书已经验收，内容为 1 时代表图书没有验收。可以使用如下代码设置单选按钮的即时显示：

例程 52 代码位置：光盘\TM\05\src\com\wsy\iframe\newBookCheckIFrame.java

```
if(table.getValueAt(selRow, 4).toString().trim().equals("否"))//1 代表没有验收
    radioButton2.setSelected(true);
else
    radioButton1.setSelected(true);
```

5.12 格式化的文本框

在 Swing 中，当限制文本框中只能输入数字，而不是其他字符时，可以为文本框添加键盘监听事件，并且在键盘按下事件方法中销毁所有非法字符。虽然这种方法被广泛使用，但这种方法存在某种弊端，当用户使用 Ctrl+V 组合键将文本复制到文本框中，并没有触发键盘按键事件，这时同样可以输入非法字符而系统并不能进行相应处理。为了处理格式化文本框，Swing 的设计者提供了 JFormattedTextField 类，该类不仅能用于数字的输入，还能用于日期的输入。

5.12.1 使用 JFormattedTextField 限制整型数字输入

初始化一个格式化文本框有几种方式，Java API 中提供了如下 6 个构造函数：

- ☑ JFormattedTextField()。
- ☑ JFormattedTextField(Format format)。
- ☑ JFormattedTextField(JFormattedTextField.AbstractFormatter formatter)。
- ☑ JFormattedTextField(JFormattedTextField.AbstractFormatterFactory factory)。
- ☑ JFormattedTextField(JFormattedTextField.AbstractFormatterFactory factory,Object currentValue)。
- ☑ JFormattedTextField(Object value)。

在这里使用第二种方式，代码如下：

```
JFormattedTextField intField=new JFormattedTextField(NumberFormat.getInstance());
```

其中，NumberFormat 类中的 getInstance()方法返回 Format 对象，实现将字符串格式化为整型，当在窗体中创建了此格式化文本框，如果用户输入非数字型字符，将会返回文本框默认值。

5.12.2 使用 JFormattedTextField 限制日期输入

格式化文本框不仅可以格式化整型数字，还可以格式化日期型数据。Java 中格式日期通常使用 SimpleDateFormat 类，它继承自 java.text.DateFormat 类，而 java.text.DateFormat 类继承自 java.text.Format 类，DateFormat 类中的 getDateInstance()方法返回 DateFormat 对象，所以在 JFormattedTextField 文本框中可以使用 SimpleDateFormat 类格式化日期。

这里使用 SimpleDateFormate(String pattern)方法初始化 SimpleDateFormat 类对象。例如：

```
SimpleDateFormat myfmt=new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
```

其中，yyyy 代表年、MM 代表月、dd 代表天、hh 代表小时、mm 代表分钟、ss 代表秒。

说明 为了区分月份和分钟，使用大写的 M 与小写的 m 进行区分。

格式化完成日期，可以将 Format 对象放入格式化文本框中。例如：

```
pubDate= new JFormattedTextField(myfmt.getDateInstance());
```

这时如果用户在文本框中输入与字符串格式不同的时间，系统会自动将文本框置为默认值。

5.13　本 章 小 结

本章运用软件工程的设计思想，通过一个完整的图书馆管理系统带领读者走完一个系统的开发流程。同时，在程序的开发过程中，采用了 Swing 机制，使整个系统的设计思路更加清晰。通过本章的学习，读者不仅可以了解一般系统的开发流程，而且还应该对 Swing 语言有比较深入的了解，同时读者可以掌握在 Swing 项目中使用实现 Action 接口的开发模式，掌握如何创建菜单栏与工具栏以及如何格式化窗体中的文本框等技术，为今后应用 Swing 语言开发程序奠定了坚实的基础。

第 6 章

企业快信

（Swing+JavaDB 实现）

随着互联网的迅速发展，短信和 E-mail 已经成为人与人之间沟通的桥梁，越来越多的人开始选择通过网络进行即时沟通。为此，越来越多的网站开始提供发送 E-mail 以及收发手机短信的功能。与此同时，短信和 E-mail 也以其快捷、无时空限制、低成本等优势受到众多企业的青睐，成为企业移动商务的主流应用方式。本章中介绍的企业快信就是为企业提供的短信和 E-mail 群发的解决方案。

通过阅读本章，可以学习到：

- 掌握开发企业快信的基本流程
- 掌握分析并设计数据库的方法
- 掌握使用短信猫收发短信的方法
- 掌握通过 Java Mail 进行邮件群发的方法
- 掌握短信猫和 Java Mail 组件的使用方法
- 掌握 JavaDB 数据库在实际开发中的应用

6.1　企业快信概述

在企业信息化的今天，效率决定成败，企业内、外部沟通的及时性将直接影响企业的运作效率。现在多数企业的办公自动化系统（即 OA）的信息传递仅限于计算机内部网络，如果用户不在线，将无法知道是否有新的工作或紧急通知，为了确认是否有待办工作，不得不经常去访问 OA，检索是否有新任务，而事实上这种检索的结果经常是徒劳。这样不仅造成了机器资源的浪费，也造成了人力资源的浪费。因此急需一套成型的企业快信系统解决上述问题。

6.2　系 统 分 析

6.2.1　需求分析

企业快信的作用是帮助企业解决企业内部、企业与外部沟通难、信息不能及时传播等问题。为此，企业快信系统需要提供邮件群发、短信群发等功能。通过对多数企业日常业务的考察、分析，并结合短信及邮件自身的特点，得出本系统要求具有以下功能：

- ☑ 用于管理客户和员工信息的名片夹管理功能。
- ☑ 用于对常用短语及其类别进行管理的信息库管理功能。
- ☑ 短信群发功能。
- ☑ 邮件群发功能。
- ☑ 发送邮件附件的功能。

6.2.2　可行性研究

开发任何一个基于计算机的系统，都会受到时间和资源上的限制。因此，在接受任何一个项目开发任务之前，必须根据客户可能提供的时间和资源条件进行可行性分析，以减少项目开发风险，避免人力、物力和财力的浪费。可行性分析与风险分析在很多方面是相互关联的，项目风险越大，开发高质量的软件的可行性就越小。

1．经济可行性

采用短信作为企业的移动通信手段，将给企业对内、对外进行信息传递与沟通带来革命性的变化，从而使得移动办公、客户服务、员工沟通等运作效率显著提升，而成本则显著下降。值得说明的是，虽然短信有以上诸多优点，但它还是有一定的不足，例如信息内容单一和受到字数限制等。为解决这一问题，在企业快信中提供了邮件群发功能。通过邮件进行沟通也是目前比较流行的方式，它也具备实用、方便和廉价等优点。

2．技术可行性

开发一个企业快信系统，涉及的技术问题不会太多，主要用到的技术就是使用短信猫和 Java Mail

组件来实现收发短信和群发邮件等功能。由于采用北京人大金仓信息技术有限公司开发的短信猫，并且该公司也提供了相应的应用程序开发包，所以为程序的开发提供了便利的条件。同时，Java Mail 组件是 Sun 公司发布的一种用于读取、编写和发送电子邮件的包，利用它可以方便地实现邮件群发。

6.3 系 统 设 计

6.3.1 系统目标

根据前面所作的需求分析及用户的需求可知，企业快信属于小型的企业通信软件，在系统实施后，应达到以下目标：

- ☑ 界面设计友好、美观。
- ☑ 操作灵活、方便。
- ☑ 提供功能强大的信息库管理，方便用户进行短信息的编写。
- ☑ 提供邮件群发功能，提高工作效率。
- ☑ 在发送短信时，可以直接从现有信息库中获取信息内容。
- ☑ 对用户输入的数据，进行严格的数据检验，尽可能地避免人为错误。
- ☑ 数据存储安全、可靠。

6.3.2 系统功能结构

根据企业快信的特点，可以将其分为名片夹管理、信息库管理、短信群发、邮件群发、系统参数设置、系统设置 6 个部分，其中各个部分及其包括的具体功能模块如图 6.1 所示。

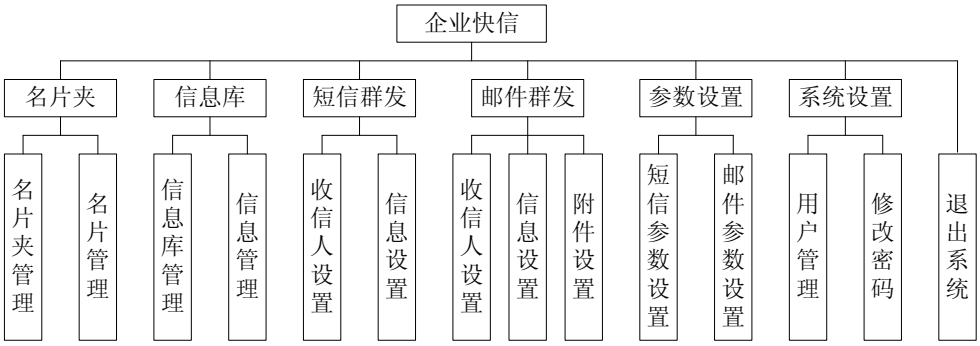


图 6.1 系统功能结构

6.3.3 业务流程图

企业快信的系统流程如图 6.2 所示。

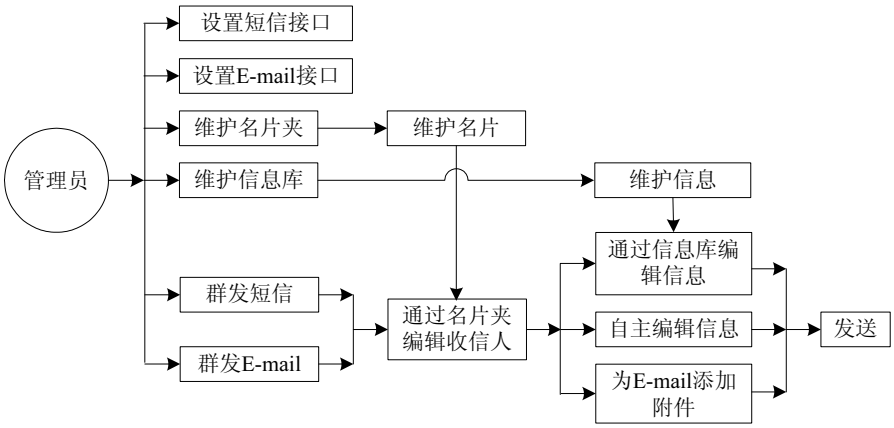


图 6.2　系统流程图

6.3.4　系统预览

企业快信由多个程序界面组成，下面仅列出几个典型界面，其他界面效果参见光盘中的源程序。

群发短信和 E-mail 的功能均在主窗体中完成，分别如图 6.3 和图 6.4 所示。在“收信人列表”中显示的为所有接收信息的人员，需要从“名片夹”中添加；如果要发送手机短信，则选中“信息内容”中的“短信内容”面板，可以直接输入短信内容，也可以在下面的“信息库”中选择已有的信息内容。如果是发送 E-mail，还可以通过单击“添加附件”按钮添加附件。

图 6.3　群发短信界面（光盘\…\TipWizardFrame.java）

单击图 6.3 中右下方的“添加”按钮，将打开如图 6.5 所示的“添加名片”对话框，该对话框用来向名片夹中添加名片。

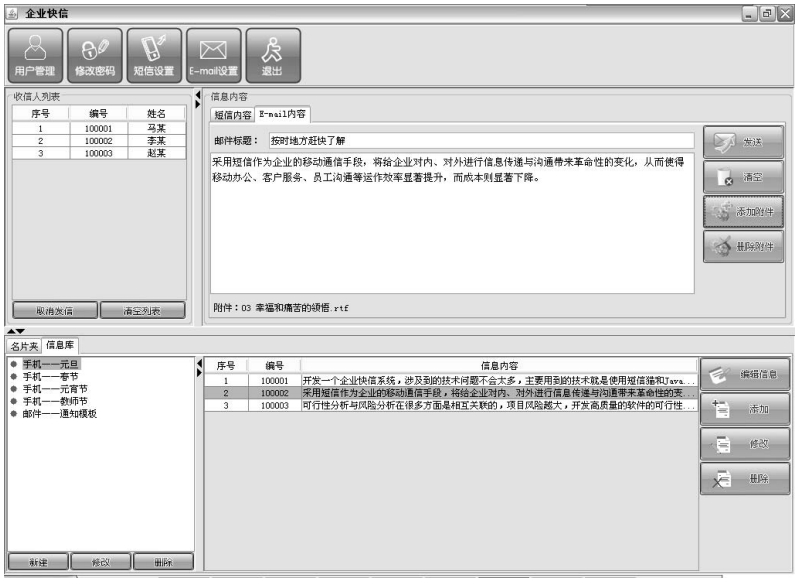


图 6.4　群发 E-mail 界面（光盘\…\TipWizardFrame.java）

图 6.5　添加名片（光盘\…\explorer\PersonnelDialog.java）

单击图 6.4 中右下方的“添加”按钮，将打开如图 6.6 所示的“添加信息”对话框，该对话框用来向信息库中添加信息。

单击“短信设置”按钮，将打开如图 6.7 所示的“短信猫设置”对话框，该对话框用来设置发送短信的接口信息。

单击“E-mail 设置”按钮，将打开如图 6.8 所示的“邮箱设置”对话框，该对话框用来设置发送 E-mail 的接口信息。

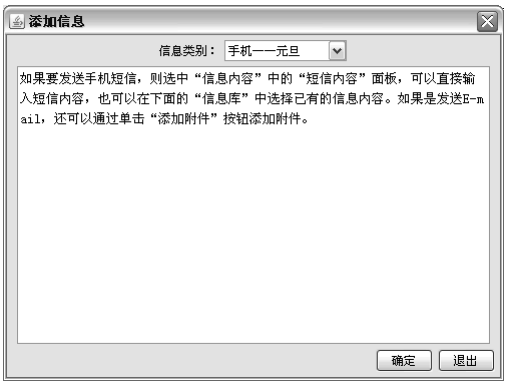


图 6.6　添加信息（光盘\…\explorer\InfoDialog.java）

图 6.7　短信猫设置（光盘\…\explorer\LetterSetDialog.java）

图 6.8　邮箱设置（光盘\…\explorer\MailSetDialog.java）

说明　由于路径太长，因此省略了部分路径，省略的路径是“TM\06\ExpressLetter\src\com\mwq\frame”。

6.3.5　构建开发环境

在开发企业快信时，需要具备下面的软件环境。

☑　操作系统：Windows 2003 以上。

☑　Java 开发包：JDK 1.6 以上。

☑　Java Mail 开发包：Java Mail 1.4。

☑　短信猫：北京人大金仓信息技术有限公司生产的串口短信猫（DG-C1A）。

☑　数据库：JavaDB。

☑　浏览器：IE 6.0 以上。

☑　分辨率：最佳效果为 1024×768 像素。

由于本系统中需要使用短信猫及 Java Mail 组件，下面将详细介绍如何配置短信猫及 Java Mail 的开发环境。

1．建立短信猫的开发环境

在使用短信猫时，首先要将短信猫安装到使用的计算机上，并接通电源，然后将短信猫提供的通信动态库 BestMail.dll 复制到 JDK 安装路径下的 jre\bin 文件夹下（例如 C:\jdk1.6.0_03\jre\bin），最后将封装的 Java 类库 BestMail.jar 添加到工程的构建路径中。

2．建立 Java Mail 的开发环境

由于目前 Java Mail 还没有被加在标准的 Java 开发工具中，所以在使用前必须另外下载 Java Mail API，以及 Sun 公司的 JAF（JavaBeans Activation Framework），Java Mail 的运行必须信赖于 JAF 的支持。

☑　下载并构建 Java Mail API

Java Mail API 是发送和接收 E-mail 的核心 API，可以到网址 http://java.sun.com/products/ javamail/downloads/index.html 上下载，目前最新版本的文件名为 javamail-1_4.zip。下载后解压缩到硬盘上，并在系统的环境变量 CLASSPATH 中指定 activation.jar 文件的放置路径。例如，将 mail.jar 文件复制到 C:\JavaMail 文件夹中，可以在环境变量 CLASSPATH 中添加以下代码：

```
C:\JavaMail\mail.jar;
```

如果不想更改环境变量，也可以把 mail.jar 添加到工程的构建路径中。

☑　下载并构建 JAF

目前 Java Mail API 的所有版本都需要 JAF（JavaBeans Activation Framework）的支持。JAF 为输入的任意数据块提供了支持，并能相应地对其进行处理。

JAF 可以到网址 http://java.sun.com/products/javabeans/jaf/downloads/index.html 上下载，当前最新版本的 JAF 文件名为 jaf-1_1-fr.zip，下载后解压缩到硬盘上，并在系统的环境变量 CLASSPATH 中指定 activation.jar 文件的放置路径。例如，将 activation.jar 文件复制到 C:\JavaMail 文件夹中，可以在环境变量 CLASSPATH 中添加以下代码：

```
C:\JavaMail\activation.jar;
```

如果不想更改环境变量，也可以把 activation.jar 添加到工程的构建路径中。

6.3.6 文件夹组织结构

在编写代码前，可以把系统中可能用到的文件夹先创建出来（例如创建一个名为 img 的文件夹，用来保存程序中使用的效果图片），这样不但方便以后的开发工作，也可以规范软件的整体架构。笔者在开发企业快信时，设计了如图 6.9 所示的文件夹架构。在开发时，只需要将所创建的文件保存在相应的文件夹中即可。

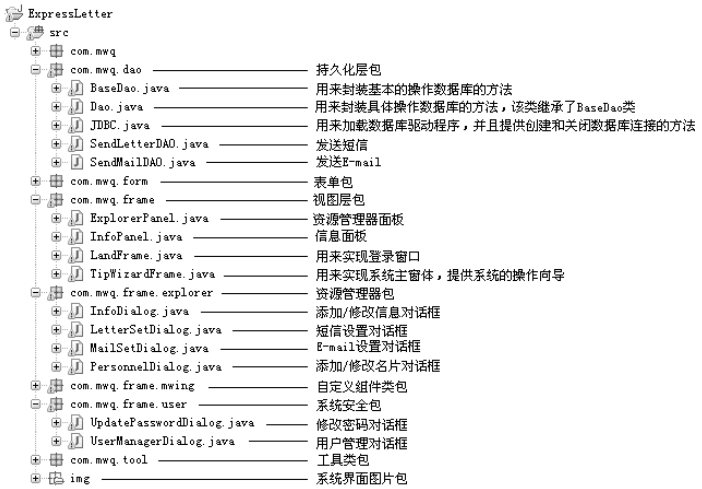


图 6.9 企业快信文件夹组织结构

6.4 数据库设计

6.4.1 数据库分析

由于本系统是一个小型实用的信息群发系统，提供了名片夹和信息库的功能，用户可以将常用的人员保存到名片夹中，将常用的短语保存到信息库中，以方便使用。基于这个特点以及语言特点，本系统将采用 JavaDB 数据库作为底层数据库，以方便用户使用本系统。

6.4.2 数据库概念设计

根据以上对系统所作的需求分析和系统设计，规划出本系统中使用的数据库实体分别为类型实体、档案实体、常用短语实体、管理员实体。下面将给出几个关键实体的 E-R 图。

☑ 档案实体

档案实体包括编号、类型、姓名、性别、出生日期、公司、部门、职务、移动电话和 E-mail，档案实体的 E-R 图如图 6.10 所示。

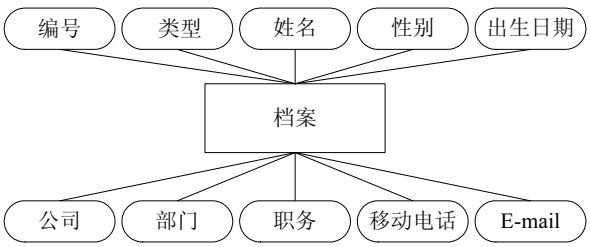


图 6.10 档案实体 E-R 图

☑ 常用短语实体

常用短语实体包括编号、类型和内容，常用短语实体的 E-R 图如图 6.11 所示。

☑ 类型实体

类型实体包括编号、名称和使用者（即用于档案实体还是用于常用短语实体），类型实体的 E-R 图如图 6.12 所示。

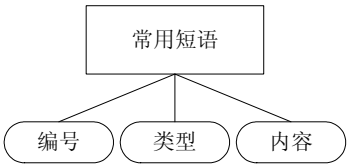


图 6.11　常用短语实体 E-R 图

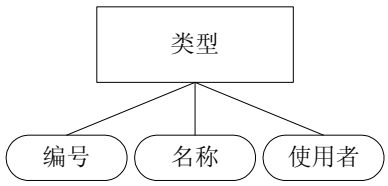


图 6.12　类型实体 E-R 图

6.4.3　数据库逻辑结构设计

在数据库概念设计中已经分析了档案实体、常用短语实体和类型实体，这些实体对象是数据表结构的基本模型，最终的数据模型都要实施到数据库中，形成整体的数据结构。

表 tb_personnel 的表结构如表 6.1 所示。

表 6.1　表 tb_personnel 的表结构

序　号	列　名	类　型	允 许 空	主　键	外　键
1	Num	int	否	是	
2	type_id	int	否		是
3	Name	varchar(8)	否		
4	Sex	char(2)	否		
5	Birthday	date	否		
6	Company	varchar(50)	否		
7	Dept	varchar(40)	否		
8	Duty	varchar(50)	否		
9	Handset	varchar(15)	否		
10	Email	varchar(30)	否		

表 tb_info 的表结构如表 6.2 所示。

表 6.2　表 tb_info 的表结构

序　号	列　名	类　型	允 许 空	主　键	外　键
1	num	int	否	是	
2	type_id	int	否		是
3	name	varchar(500)	否		

表 tb_type 的表结构如表 6.3 所示。

表 6.3　表 tb_type 的表结构

序　号	列　名	类　型	允 许 空	主　键	外　键
1	Id	int	否	是	
2	Name	varchar(20)	否		
3	Used	char(4)	否		

6.4.4 视图设计

完成数据库建模后，还可以根据实际需要，建立一些视图，通过对视图的应用，可以减少在程序中编写复杂的 SQL 语句。在开发企业快信时，经常需要根据类型查询档案或信息，如果不建立视图，就要先查询该类型的主键，然后再根据主键查询档案或信息，而通过视图则可以直接根据类型查询档案或信息。

为表 tb_personnel 和表 tb_type 建立的视图 v_personnel_type 的结构如表 6.4 所示。

表 6.4　视图 v_personnel_type 的结构

序　号	列　名	所 属 表	序　号	列　名	所 属 表
1	Num	tb_personnel	6	Company	tb_personnel
2	type_name	tb_type	7	Dept	tb_personnel
3	Name	tb_personnel	8	Duty	tb_personnel
4	Sex	tb_personnel	9	Handset	tb_personnel
5	Birthday	tb_personnel	10	Email	tb_personnel

为表 tb_info 和表 tb_type 建立的视图 v_ info _type 的结构如表 6.5 所示。

表 6.5　视图 v_info_type 的结构

序　号	列　名	所 属 表	序　号	列　名	所 属 表
1	Num	tb_info	3	Name	tb_info
2	type_name	tb_type			

6.5　主窗体设计

为了增强企业快信系统的可操作性，系统将主窗体划分为了 3 个部分，即工具栏、信息发送区和资源管理区，如图 6.13 所示。

在工具栏中包含“用户管理”、“修改密码”、“短信设置”、“E-mail 设置”和“退出”5 个按钮，其中“短信设置”和“E-mail 设置”按钮用来设置发送短信和 E-mail 必须设置的接口信息。实现工具栏的关键代码如下：

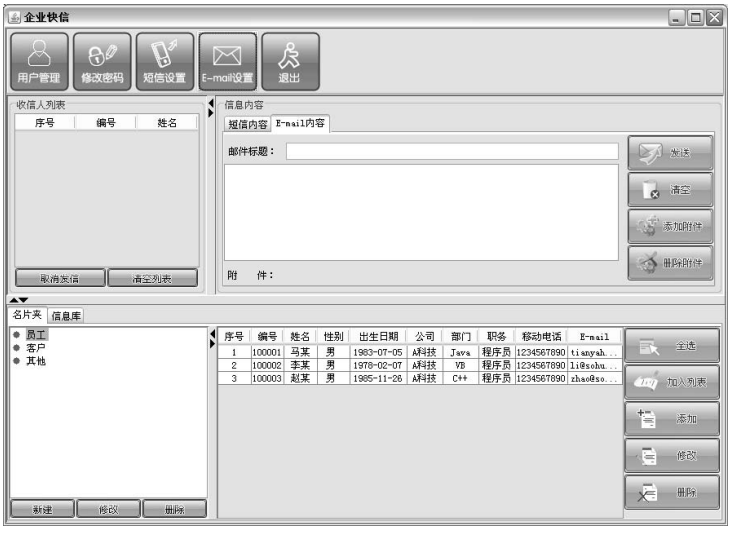


图 6.13　企业快信主窗体界面

例程 01　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ TipWizardFrame.java

```
final JToolBar toolBar = new JToolBar();                                    //创建工具栏对象
toolBar.setBorder(new EtchedBorder(EtchedBorder.LOWERED));                  //设置工具栏的边框样式
❶   toolBar.setFloatable(false);                                            //设置工具栏不可移动
getContentPane().add(toolBar, BorderLayout.NORTH);                          //将工具栏添加到面板中
final JButton userButton = new JButton();                                   //创建用户管理按钮
userButton.addActionListener(new ActionListener() {                         //添加事件监听器
    public void actionPerformed(ActionEvent e) {                            //处理按钮事件
        UserManagerDialog dialog = new UserManagerDialog();                 //创建用户管理对话框对象
        dialog.setVisible(true);                                            //设置用户管理对话框可见
    }
});
userButton.setMargin(new Insets(0, 0, 0, 0));                               //设置按钮的边框
URL userUrl = this.getClass().getResource("/img/user.png");                 //获得按钮默认图片的路径
❷   userButton.setIcon(new ImageIcon(userUrl));                             //设置按钮的默认图片
URL userOverUrl = this.getClass().getResource("/img/user_over.png");        //获得按钮鼠标经过图片的路径
❸   userButton.setRolloverIcon(new ImageIcon(userOverUrl));                 //设置按钮的鼠标经过图片
    toolBar.add(userButton);                                                //将按钮添加到工具栏
```

代码贴士

❶ setFloatable(boolean floatable)：该方法用来设置工具栏是否可以移动，默认为 true，即可以移动。

❷ setIcon(Icon defaultIcon)：该方法用来设置按钮在通常情况下显示的图片。

❸ setRolloverIcon(Icon defaultIcon)：该方法用来设置当鼠标经过按钮时显示的图片。

信息发送区的主要功能是发送信息，包括手机短信和 E-mail。信息发送区的左侧为“收信人列表”，通过列表下方的按钮可以取消部分或全部收信人；右侧为信息内容，如果是发送 E-mail，则既可以设置邮件标题，还可以添加附件；如果是发送手机短信，则只可以编辑信息内容。

在资源管理区又分为名片夹和信息库，它们的左侧均为分类信息，通过其下方的按钮可以维护分类信息，包括添加、修改和删除类别；它们的中间部分均用来显示选中类别包含的名片或信息列表；它们的右侧均为用来操作列表的按钮，包括添加、修改和删除名片或信息，以及将列表中的选中行添加到“收信人列表”或“信息内容”中。

为了适当减小主窗体类 TipWizardFrame 的大小，本系统的主窗体由 3 个类完成，主窗体类 TipWizardFrame 负责绘制工具栏和收信人列表，信息内容由 InfoPanel 类负责绘制，资源管理区由 ExplorerPanel 类负责绘制。

为了增加信息发送区和资源管理区的操作空间，将这两个工作区放在了一个 JSplitPane 面板中，这样就可以通过单击图 6.14 中的▲和▼按钮，调整当前使用工作区的大小。

图 6.14　由 JSplitPane 实现的垂直分割面板

实现垂直分割面板的关键代码如下：

例程 02　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\TipWizardFrame.java

```
final JSplitPane workaroundSplitPane = new JSplitPane();                    //创建分割面板对象
❶   workaroundSplitPane.setDividerSize(12);                                 //设置分割条的宽度
❷   workaroundSplitPane.setOneTouchExpandable(true);                        //设置为支持快速展开/折叠分割条
```

```
❸ workaroundSplitPane.setDividerLocation(310);                    //设置面板默认的分割位置
❹ workaroundSplitPane.setPreferredSize(new Dimension(0, 590));    //设置分割面板的首选高度
❺ workaroundSplitPane.setOrientation(JSplitPane.VERTICAL_SPLIT);  //设置为垂直分割
```

代码贴士

❶ setDividerSize (int width)：该方法用于设置分割条的宽度，默认宽度为 5 像素。

❷ setOneTouchExpandable(boolean expandable)：该方法用于设置是否支持快速展开/折叠的分割条，默认为 false，即不支持。需要注意的是，有些外观可能不支持该功能。

❸ setDividerLocation(int location)：该方法用于设置分割条的显示位置，入口参数为针对分割面板的绝对位置；该方法拥有一个重载方法 setDividerLocation(double proportionalLocation)，用来以百分比的形式设置分割条的显示位置，proportionalLocation 为 0～1.0 的双精度浮点值。

❹ setPreferredSize(Dimension preferredSize)：该方法用于设置分割面板的首选大小，构造方法 Dimension(int width, int height)的第一个入口参数为宽度，第二个入口参数为高度。

❺ setOrientation(int orientation)：该方法用于设置分割面板的分割方向，默认为在水平方向分割，即 HORIZONTAL_SPLIT，如果要用于垂直方向分割，则需要设置为 VERTICAL_SPLIT。

同样，在信息发送区和资源管理区也使用了 JSplitPane 面板，这样就可以通过单击图 6.15 中的◀和▶按钮，调整当前使用工作区的大小。

图 6.15　由 JSplitPane 实现的水平分割面板

实现水平分割面板的关键代码如下：

例程 03　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\TipWizardFrame.java

```
final JSplitPane sendSplitPane = new JSplitPane();                    //创建分割面板对象
sendSplitPane.setOneTouchExpandable(true);                            //设置为支持快速展开/折叠分割条
sendSplitPane.setDividerSize(12);                                     //设置分割条的宽度
sendSplitPane.setDividerLocation(244);                                //设置面板默认的分割位置
sendSplitPane.setOrientation(JSplitPane.HORIZONTAL_SPLIT);            //设置为水平分割
```

6.6　公共模块设计

6.6.1　编写数据库连接类

数据库连接类负责加载数据库驱动程序，以及创建和关闭数据库连接。为了最大程度地应用每个已经创建的数据库连接，这里将其保存到了 ThreadLocal 类的对象中。如果是数据库不存在，还负责建立数据库。

（1）在数据库连接类中定义一些常量，包括连接数据库使用的驱动程序、连接数据库的路径，并且定义一个 ThreadLocal 类的对象，用来保存已经创建的数据库连接。关键代码如下：

例程 04　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\dao\JDBC.java

```
private static final String DRIVERCLASS = "org.apache.derby.jdbc.EmbeddedDriver";  //数据库驱动
private static final String URL = "jdbc:derby:db_ExpressLetter";                   //数据库 URL
```

```
private static final ThreadLocal<Connection> threadLocal = new ThreadLocal<Connection>(); //保存数据库连接线程
private static Connection conn = null;                                                    //数据库连接
```

（2）编写用来加载数据库驱动程序的代码，并且判断数据库文件是否已经存在，如果不存在，还要创建数据库。通常情况下将其放到静态代码块中，这样做的好处是只在该类第一次被加载（即第一次被调用）时执行加载数据库驱动程序以及创建数据库的动作，避免了反复加载数据库驱动程序，或者是反复判断数据库是否已经存在，从而提高软件的性能。关键代码如下：

例程 05　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\dao\JDBC.java

```
static {                    //通过静态方法加载数据库驱动，并且在数据库不存在的情况下创建数据库
    try {
        Class.forName(DRIVERCLASS).newInstance();                          //加载数据库驱动
        File dbExpressLetterFile = new File("db_ExpressLetter");           //创建数据库文件对象
        if (!dbExpressLetterFile.exists()) {                               //判断数据库文件是否存在
            String[] sqls = new String[6];                                 //定义创建数据库的 SQL 语句
            …//由于篇幅有限，此处省略了负责初始化创建数据库的 SQL 语句数组的代码
            conn = DriverManager.getConnection(URL + ";create=true");      //创建数据库连接
            threadLocal.set(conn);                                         //保存数据库连接
            Statement stmt = conn.createStatement();                       //创建数据库连接状态对象
            for (int i = 0; i < sqls.length; i++) {                        //通过执行 SQL 语句创建数据库
                stmt.execute(sqls[i]);                                     //执行 SQL 语句
            }
            stmt.close();                                                  //关闭数据库连接状态对象
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

（3）编写用来创建和关闭数据库连接的方法。这里将这两个方法均定义为静态的，这样通过类名就可以调用方法，方便使用。在这两个方法中首先从 ThreadLocal 类的对象中获得数据库连接，然后判断是否存在可用的数据库连接，如果存在则直接返回或关闭，否则重新创建。关键代码如下：

例程 06　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\dao\JDBC.java

```
public static Connection getConnection() {                                 //创建数据库连接的方法
    conn = (Connection) threadLocal.get();                                 //从线程中获得数据库连接
    if (conn == null) {                                                    //没有可用的数据库连接
        try {
            conn = DriverManager.getConnection(URL);                       //创建新的数据库连接
            threadLocal.set(conn);                                         //将数据库连接保存到线程中
        } catch (Exception e) {
            String[] infos = { "未能成功连接数据库！", "请确认本软件是否已经运行！" };
            JOptionPane.showMessageDialog(null, infos);                    //弹出连接数据库失败的提示
            System.exit(0);                                                //关闭系统
            e.printStackTrace();
        }
    }
    return conn;
```

```
}
public static boolean closeConnection() {                    //关闭数据库连接的方法
    boolean isClosed = true;                                 //默认关闭成功
    conn = (Connection) threadLocal.get();                   //从线程中获得数据库连接
    threadLocal.set(null);                                   //清空线程中的数据库连接
    if (conn != null) {                                      //数据库连接可用
        try {
            conn.close();                                    //关闭数据库连接
        } catch (SQLException e) {
            isClosed = false;                                //关闭失败
            e.printStackTrace();
        }
    }
    return isClosed;
}
```

6.6.2 封装常用的操作数据库的方法

对数据库的操作包括查询、添加、修改和删除，其中查询是通过 executeQuery(String sql)方法执行 SQL 语句的，添加、修改和删除是通过 executeUpdate(String sql)方法执行 SQL 语句的。在本系统中共提供了 4 个用来执行查询的方法，分别用来查询多个记录、查询指定记录、查询多个记录的指定值和查询指定记录的指定值；一个用来添加、修改和删除记录的方法。由于篇幅有限，在这里只介绍用来查询多个记录的方法，以及用来添加、修改和删除记录的方法。

下面的方法用来查询多个记录，为了将检索结果直接应用于表格组件，这里全部用 Vector 向量对象封装查询结果，并且为代表每一行的向量添加了行序号。关键代码如下：

例程 07 代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\dao\BaseDao.java

```
protected Vector selectSomeNote(String sql) {
    Vector<Vector<Object>> vector = new Vector<Vector<Object>>();   //创建结果集向量
    Connection conn = JDBC.getConnection();                         //获得数据库连接
    try {
        Statement stmt = conn.createStatement();                    //创建连接状态对象
        ResultSet rs = stmt.executeQuery(sql);                      //执行 SQL 语句获得查询结果
        int columnCount = rs.getMetaData().getColumnCount();        //获得查询数据表的列数
        int row = 1;                                                //定义行序号
        while (rs.next()) {                                         //遍历结果集
            Vector<Object> rowV = new Vector<Object>();             //创建行向量
            rowV.add(new Integer(row++));                           //添加行序号
            for (int column = 1; column <= columnCount; column++) {
                rowV.add(rs.getObject(column));                     //添加列值
            }
            vector.add(rowV);                                       //将行向量添加到结果集向量中
        }
        rs.close();                                                 //关闭结果集对象
        stmt.close();                                               //关闭连接状态对象
    } catch (SQLException e) {
        e.printStackTrace();
```

```
        }
        return vector;                                          //返回结果集向量
    }
```

下面的方法用来添加、修改和删除记录，这里采用手动提交，目的是捕获持久化异常，并回滚数据库，以保证数据的合法性。关键代码如下：

例程 08　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\dao\BaseDao.java

```
public boolean longHaul(String sql) {
    boolean isLongHaul = true;                                  //默认持久化成功
    Connection conn = JDBC.getConnection();                     //获得数据库连接
    try {
        conn.setAutoCommit(false);                              //设置为手动提交
        Statement stmt = conn.createStatement();                //创建连接状态对象
        stmt.executeUpdate(sql);                                //执行 SQL 语句
        stmt.close();                                           //关闭连接状态对象
        conn.commit();                                          //提交持久化
    } catch (SQLException e) {
        isLongHaul = false;                                     //持久化失败
        try {
            conn.rollback();                                    //回滚
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
        e.printStackTrace();
    }
    return isLongHaul;                                          //返回持久化结果
}
```

6.7　资源管理模块设计

资源管理模块分为名片夹管理和信息库管理两个功能，这两个功能的开发思路和实现方法基本相同，所以用到的技术也基本相同，并且在这两个功能之间有一部分代码还实现了代码重用。本节将介绍名片夹管理功能。

6.7.1　名片夹管理功能概述

名片夹管理模块分为名片夹管理和名片管理两部分。名片夹管理部分包括新建、修改和删除名片夹，界面效果如图 6.16 所示。

名片管理部分包括向名片夹添加名片、修改或删除名片夹中的名片，以及将选中的名片添加到收信人列表中，界面效果如图 6.17 所示。

当向名片夹中添加名片时，默认为向当前选中的名片夹中添加。

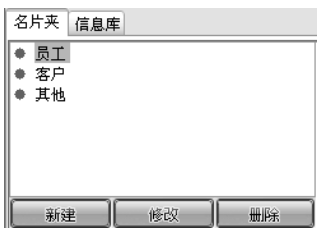


图 6.16　名片夹管理

例如在图 6.16 中选中的是“员工”名片夹，在这种情况下单击图 6.17 中的“添加”按钮，弹出对话框的“类别”选项则默认为选中“员工”选项，如图 6.18 所示。

图 6.17　名片管理

图 6.18　添加名片

如果在没有任何名片夹的情况下单击图 6.17 中的“添加”按钮，将弹出如图 6.19 所示的提示先建立名片夹的提示框。

图 6.19　提示先建立名片夹

6.7.2　名片夹管理功能技术分析

在添加和修改名片夹时，都要验证名片夹名称是否已经存在。为了提高代码的重用性，这里通过一个单独的方法完成验证，该方法的第一个入口参数为添加或修改节点的父节点，第二个入口参数为新添加节点或修改后节点的名称，如果新的节点名称不存在，则返回 false，否则返回 true，并且弹出该名称已经存在的提示信息。关键代码如下：

例程 09　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
private boolean isHad(DefaultMutableTreeNode treeNode, String newChildName) {  //验证该名称是否已经存在
    boolean had = false;                                                        //默认为不存在
❶   int childCount = treeNode.getChildCount();                                  //获得子节点的数量
    for (int i = 0; i < childCount; i++) {                                      //遍历子节点
        DefaultMutableTreeNode childTreeNode = (DefaultMutableTreeNode) treeNode
❷           .getChildAt(i);                                                     //获得子节点对象
❸       if (childTreeNode.getUserObject().toString().equals(newChildName)) {   //判断名称是否相同
            JOptionPane.showMessageDialog(null, "该名称已经存在！", "友情提示",
                    JOptionPane.INFORMATION_MESSAGE);          //弹出该名称已经存在的提示
            had = true;                                                         //该名称已经存在
            break;                                                              //停止遍历子节点
        }
    }
    return had;                                                                 //返回结果
}
```

代码贴士

❶ getChildCount()：该方法用来获得拥有子节点的数量，返回值为 int 型。

❷ getChildAt (int index)：该方法用来获得位于指定索引位置的节点对象，索引位置从 0 开始。

❸ getUserObject()：该方法用来获得节点的内容。

6.7.3　名片夹管理功能的实现过程

名片夹管理使用的主要数据表：tb_type、tb_personnel

名片夹管理功能包括对名片夹的管理和对名片的管理，下面将分别介绍这两个管理功能的实现过程。

1．名片夹管理的实现过程

首先创建一个名片夹树，这里将名片夹树添加到了滚动面板中，避免名片夹过多时不能正常显示。名片夹树的选择模式需要为单选，即一次只能选中一个名片夹。名片夹树的根节点也不需要显示出来，即需要将数的根节点设置为不可见的。关键代码如下：

例程 10　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
final JScrollPane cardTreeScrollPane = new JScrollPane();          //创建显示名片夹树的滚动面板
cardTreePanel.add(cardTreeScrollPane);                             //添加到上级面板中
cardTreeRoot = new DefaultMutableTreeNode("root");                 //创建名片夹树的根节点
initTree(cardTreeRoot, "card");                                    //初始化名片夹树
cardTreeModel = new DefaultTreeModel(cardTreeRoot);                //创建名片夹树模型
cardTree = new JTree(cardTreeModel);                               //创建名片夹树
❶   cardTree.setRootVisible(false);                                //设置名片夹树的根节点不可见
❷   cardTree.getSelectionModel().setSelectionMode(
        TreeSelectionModel.SINGLE_TREE_SELECTION);                 //设置名片夹树的选择模式为单选
if (cardTreeRoot.getChildCount() > 0)
❸   cardTree.setSelectionRow(0);                                   //如果名片夹树存在子节点，则设置选中第一个子节点
cardTree.addTreeSelectionListener(new TreeSelectionListener() {    //为名片夹树添加节点选中事件监听器
    public void valueChanged(TreeSelectionEvent e) {
        initCardListTable();                                       //初始化名片夹列表
    }
});
cardTreeScrollPane.setViewportView(cardTree);                      //将名片夹树添加到滚动面板中
```

代码贴士

❶ setRootVisible(boolean rootVisible)：该方法用来设置树的根节点是否可见，默认为 true，即默认为可见。

❷ setSelectionMode(int mode)：该方法用来设置树的选择模式。静态常量 SINGLE_TREE_SELECTION（常量值为 1）表示只允许选中一个节点；静态常量 CONTIGUOUS_TREE_SELECTION（常量值为 2）表示允许选中多个节点，但是这些节点必须是相临的；静态常量 DISCONTIGUOUS_TREE_SELECTION（常量值为 4）表示允许选中多个节点，并且这些节点不必是相临的；默认的选择模式为 DISCONTIGUOUS_TREE_SELECTION。

❸ setSelectionRow(int rowIndex)：该方法用来设置选中的行，树的根节点为第 0 行。例如，A 为树的根节点，a1 和 a2 为 A 的子节点，s 为 a1 的子节点，如果 a1 节点处于合并状态，则 a2 为第 2 行；如果 a1 节点处于展开状态，则 a2 为第 3 行。

在上面的代码中用到了两个可重用的方法，一个是用来初始化树的 initTree(DefaultMutableTreeNode treeRoot, String used)方法，第一个入口参数为欲初始化树的根节点对象，第二个入口参数为树的类型，即为名片夹树还是信息库树。该方法的关键代码如下：

例程 11 代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
private void initTree(DefaultMutableTreeNode treeRoot, String used) {     //初始化树的方法
    Vector typeV = dao.sTypeByUsed(used);                                 //查询用于指定树的类型
    for (int i = 0; i < typeV.size(); i++) {                              //遍历向量
        Vector type = (Vector) typeV.get(i);                              //获得类型向量
        treeRoot.add(new DefaultMutableTreeNode(type.get(2)));            //将类型添加到树中
    }
}
```

另一个是用来初始化选中名片夹的名片列表的 initCardListTable()方法，首先要清空所有名片，然后再根据选中的名片夹添加名片，最后刷新名片列表表格模型。关键代码如下：

例程 12 代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
private void initCardListTable() {
    cardListTableValueV.removeAllElements();                              //清空名片列表
    DefaultMutableTreeNode cardTreeNode = (DefaultMutableTreeNode) cardTree
            .getLastSelectedPathComponent();                              //获得名片夹树的选中节点对象
    if (cardTreeNode != null) {                                           //判断是否存在选中的节点
        String cardName = cardTreeNode.getUserObject().toString();        //获得选中名片夹的名称
        cardListTableValueV.addAll(dao.sPersonnelVByTypeName(cardName)); //检索名片夹包含的名片
    }
    cardListTableModel.setDataVector(cardListTableValueV,
            cardListTableColumnV);                                        //刷新名片列表表格模型
}
```

下面实现新建名片夹功能。首先需要调用 addTreeNode()方法获得名片夹名称，如果名称为空即用户取消了新建操作，如果不为空则将新的名片夹添加到名片夹树的最后。关键代码如下：

例程 13 代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
addCardTypeButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String name = addTreeNode(cardTreeRoot, "名片夹");                //获得新名片夹名称
        if (name != null) {                                               //当用户取消新建时名称为空
            int childCount = cardTreeRoot.getChildCount();                //获得当前拥有名片夹的数量
            cardTreeModel.insertNodeInto(new DefaultMutableTreeNode(
                    name), cardTreeRoot, childCount);                     //在名片夹树的最后创建新的名片夹
            cardTreeModel.reload();                                       //刷新名片夹树模型
            cardTree.setSelectionRow(childCount);                         //设置新建名片夹为选中状态
            dao.iType(name, "card");                                      //将新建名片夹保存到数据库中
        }
    }
});
```

addTreeNode(DefaultMutableTreeNode treeNode, String typeName)方法用来弹出输入框令用户输入名片夹名称，并且调用 isHad()方法判断输入的名称是否已经存在，该方法的第一个入口参数为欲添加节点的父节点，第二个入口参数为新添加节点的类型，即为“名片夹”还是“信息库”。关键代码如下：

例程 14　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
private String addTreeNode(DefaultMutableTreeNode treeNode, String typeName) {
    String nodeName = "";                                          //创建节点名称为空字符串
    while (nodeName.length() == 0) {                               //判断节点名称的长度是否为 0
        nodeName = JOptionPane.showInputDialog(null, "请输入" + typeName
                + "名称：", "新建" + typeName, JOptionPane.INFORMATION_MESSAGE);//接受用户输入名称
        if (nodeName == null) {                                    //判断节点名称是否为空值
            break;                                                 //为空值即用户取消了新建，则跳出循环
        } else {
            nodeName = nodeName.trim();                            //去掉首尾空格
            if (nodeName.length() > 0) {                           //判断节点名称的长度是否为 0
                if (isHad(treeNode, nodeName))                     //如果不为 0 则判断该名称是否已经存在
                    nodeName = "";                                 //如果存在则设置节点名称为空字符串
            }
        }
    }
    return nodeName;                                               //返回节点名称
}
```

下面实现修改名片夹功能。首先要判断是否存在被选中的节点，如果存在则获得选中的节点对象，并弹出确认修改的提示框，如果确认修改则调用 updateTreeNode()方法获得修改后的名片夹名称，如果名称为空即用户取消了修改操作，如果不为空则修改名片夹的名称。关键代码如下：

例程 15　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
updCardTypeButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        TreePath selectionPath = cardTree.getSelectionPath();              //获得选中节点的路径对象
        if (selectionPath == null) {                                       //判断路径是否为空
            JOptionPane.showMessageDialog(null, "请选择要修改的名片夹！", "友情提示",
                    JOptionPane.INFORMATION_MESSAGE);                      //如果为空则弹出提示
        } else {
            DefaultMutableTreeNode treeNode = (DefaultMutableTreeNode) selectionPath
                    .getLastPathComponent();                               //获得选中节点对象
            String nowName = treeNode.getUserObject().toString();          //获得选中节点的名称
            int i = JOptionPane.showConfirmDialog(null, "确定要修改名片夹"" + nowName + ""的名称？",
                    "友情提示", JOptionPane.YES_NO_OPTION);                //弹出确认修改提示框
            if (i == 0) {                                                  //如果为 0 则修改
                String newName = updateTreeNode(treeNode);                 //获得修改后的名称
                if (newName != null) {          //判断修改后的名称是否为空，如果为空则用户取消了修改
                    treeNode.setUserObject(newName);                       //修改节点名称
                    cardTreeModel.reload();                                //刷新树
                    cardTree.setSelectionPath(selectionPath);              //设置修改的节点为选中节点
                    dao.uTypeNameByName("card", nowName, newName);//将修改后的名称保存到数据库
                }
            }
        }
    }
});
```

updateTreeNode(DefaultMutableTreeNode treeNode)方法用来弹出输入框令用户输入修改后的名片夹名称，并且调用 isHad()方法判断输入的名称是否已经存在，该方法的入口参数为欲修改的节点对象。关键代码如下：

例程 16 代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
private String updateTreeNode(DefaultMutableTreeNode treeNode) {
    DefaultMutableTreeNode parentNode = (DefaultMutableTreeNode) treeNode
            .getParent();                                          //获得欲修改节点的父节点
    String newNodeName = "";                                       //创建节点名称为空字符串
    while (newNodeName.length() == 0) {                            //判断节点名称的长度是否为 0
        newNodeName = JOptionPane.showInputDialog(null, "请输入新名称：", "修改名称",
                JOptionPane.INFORMATION_MESSAGE);                  //接受用户输入名称
        if (newNodeName == null) {                                 //判断节点名称是否为空值
            break;                                                 //为空值即用户取消了修改，则跳出循环
        } else {
            newNodeName = newNodeName.trim();                      //去掉首尾空格
            if (newNodeName.length() > 0) {                        //判断节点名称的长度是否为 0
                if (isHad(parentNode, newNodeName))                //如果不为 0 则判断该名称是否已经存在
                    newNodeName = "";                              //如果存在则设置节点名称为空字符串
            }
        }
    }
    return newNodeName;                                            //返回节点名称
}
```

下面实现删除名片夹功能。首先判断是否存在被选中的名片夹，如果存在则弹出提示框确认是否真的要删除。在确认删除的情况下，进一步判断该名片夹是否包含名片，如果包含名片，则弹出如图 6.20 所示的提示对话框，令用户选择对包含名片的处理方式。如果单击“取消”按钮，即取消删除名片夹的操作；如果单击“删除”按钮，即删除其包含的名片；如果单击“移入其他名片夹”按钮，即将其包含的名片移入到其他名片夹中，此时会弹出一个如图 6.21 所示的提示对话框，令用户选择移入的名片夹，可移入的名片夹为除了欲删除名片夹的所有名片夹。

图 6.20 询问对包含名片的处理方式

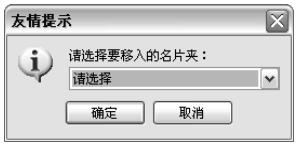


图 6.21 询问要移入的名片夹

实现删除名片夹功能的关键代码如下：

例程 17 代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
delCardTypeButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        DefaultMutableTreeNode treeNode = (DefaultMutableTreeNode) cardTree
                .getLastSelectedPathComponent();                   //获得选中的名片夹对象
        if (treeNode == null) {                                    //未选择要删除的名片夹
```

```
❶               JOptionPane.showMessageDialog(null, "请选择要删除的名片夹！", "友情提示",
                        JOptionPane.INFORMATION_MESSAGE);          //弹出提示信息
                return;                                            //并直接返回
            }
            String name = treeNode.getUserObject().toString();     //获得欲删除名片夹的名称
❷           int i = JOptionPane.showConfirmDialog(null, "确定要删除名片夹"" + name
                    + ""？ ", "友情提示", JOptionPane.YES_NO_OPTION);  //弹出删除的确认提示
            if (i != 0)                                            //用户取消了删除操作
                return;                                            //直接返回
            if (dao.sPersonnelVByTypeName(name).size() > 0) {      //该名片夹中包含名片
                String options[] = { "取消", "移入其他名片夹", "删除" };  //定义对其包含名片的处理方式
❸               int optionIndex = JOptionPane.showOptionDialog(null, "请选择对该名片夹下人员的处理方式：",
                        "友情提示", JOptionPane.YES_NO_CANCEL_OPTION,
                        JOptionPane.QUESTION_MESSAGE, null, options,
                        options[0]);                               //弹出提示信息，令用户选择处理方式
                if (optionIndex == 0)                              //用户取消了删除操作
                    return;                                        //直接返回
                int typeId = dao.sTypeIdByUsedAndName("card", name);   //获得名片夹的主键 ID
                if (optionIndex == 1) {                            //用户选择移入其他名片夹
                    Vector<Vector> cardV = dao.sTypeByUsedExcept("card", typeId);//查询可移入的名片夹
                    String[] cards = new String[cardV.size() + 1];     //创建一个选择项数组
                    cards[0] = "请选择";                            //添加一个提示选择项
                    for (int j = 0; j < cardV.size(); j++) {           //初始化选择项数组
                        cards[j + 1] = cardV.get(j).get(2).toString();  //添加可移入的名片夹
                    }
                    Object card = "请选择";                         //默认为选中“请选择”
                    while (card.equals("请选择")) {                 //当选中的为“请选择”时执行循环
❹                   card = JOptionPane.showInputDialog(null, "请选择要移入的名片夹：", "友情提示",
                                JOptionPane.INFORMATION_MESSAGE, null,
                                cards, cards[0]);                  //弹出对话框令用户选择欲移入的名片夹
                        if (card == null)                          //用户取消了删除操作
                            return;                                //直接返回
                    }
                    int newTypeId = dao.sTypeIdByUsedAndName("card", card
                            .toString());                          //获得欲移入名片夹的主键 ID
                    dao.uPersonnelTypeIdByTypeId(typeId, newTypeId);   //修改名片的外键
                }
                if (optionIndex == 2) {                            //用户选择删除其包含的名片
                    dao.dPersonnelByTypeId(typeId);                //从数据库删除其包含的名片
                }
            }
            cardTreeModel.removeNodeFromParent(treeNode);          //从名片夹树中删除名片夹
            dao.dTypeByName("card", name);                         //从数据库中删除名片夹
        }
});
```

代码贴士

❶ showMessageDialog()：该静态方法用来弹出提示某些消息的对话框，消息的类型可以为错误（ERROR_MESSAGE）、消息（INFORMATION_MESSAGE）、警告（WARNING_MESSAGE）、问题（QUESTION_MESSAGE）

或普通（PLAIN_MESSAGE）。

❷ showConfirmDialog()：该静态方法用来弹出提示某些消息的对话框，并且要求用户进行确认。

❸ showOptionDialog()：该静态方法用来弹出提示某些消息的对话框，并且要求用户进行确认。它的优点是开发人员可以自行定义用来确认的按钮。

❹ showInputDialog()：该静态方法用来弹出提示某些输入的对话框，可以接受用户的输入信息，也可以提供几个备选项供用户选择。

2．名片管理的实现过程

名片管理功能由 5 个按钮实现，即“全选”、“加入列表”、“添加”、“修改”和“删除”，下面将依次讲解这 5 个按钮的功能及具体实现代码。

“全选”按钮用来快速选中名片列表中的所有行。该按钮的实现代码很简单，只是用到了 JTable 类的 selectAll()方法。提供该按钮的主要目的是为用户提供方便。关键代码如下：

例程 18 代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
selAllButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        cardListTable.selectAll();                          //选中表格中的所有行
    }
});
```

“加入列表”按钮用来将选中的名片添加到收信人列表中。在添加到收信人列表前，首先要判断该名片是否已经被添加到收信人列表中，重复添加是没有意义的。关键代码如下：

例程 19 代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
addToSendListButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int rowCount = sendListTableModel.getRowCount();        //获得当前收信人列表中的收信人个数
        int[] selectedRows = cardListTable.getSelectedRows();   //获得名片列表中的选中行
        int index = rowCount + 1;//初始化收信人列表的序号
        for (int selectedRow = 0; selectedRow < selectedRows.length; selectedRow++) {     //遍历选中行
            int newNum = (Integer) cardListTable.getValueAt(selectedRows[selectedRow], 1);//获得名片编号
            boolean had = false;                                //默认为未加入收信人列表
            for (int row = 0; row < rowCount; row++) {          //遍历收信人列表
                int nowNum = (Integer) sendListTableModel.getValueAt(row, 1);      //获得收信人的编号
                if (newNum == nowNum) {                         //判断名片编号和收信人编号是否相同
                    had = true;                                 //已经加入收信人列表
                    break;                                      //跳出循环
                }
            }
            if (!had) {                                         //未加入收信人列表
                Vector rowV = new Vector();                     //创建一个代表收信人的向量
                rowV.add(index++);                              //添加序号
                rowV.add(newNum);                               //添加编号
                rowV.add(cardListTable.getValueAt(selectedRows[selectedRow], 2));      //添加姓名
                sendListTableModel.addRow(rowV);                //加入收信人列表
```

```
                }
            }
            cardListTable.clearSelection();                    //取消名片列表中的选中行
        }
});
```

“添加”按钮用来向名片夹中添加名片。默认为向当前选中的名片夹中添加，如果需要添加到其他名片夹中，用户也可以进行修改。此处将创建 PersonnelDialog 类的对象，该类的构造方法有 3 个入口参数，第一个入口参数为对话框的名称，第二个入口参数为当前选中名片夹的名称，第三个入口参数为名片编号，如果是添加名片，则将名片编号设置为“-1”。关键代码如下：

例程 20　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
addCardButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (cardTreeRoot.getChildCount() == 0) {                    //判断是否存在名片夹
            JOptionPane.showMessageDialog(null, "请先建立名片夹！", "友情提示",
                    JOptionPane.INFORMATION_MESSAGE);               //弹出建立名片夹的提示
        } else {
            DefaultMutableTreeNode treeNode = (DefaultMutableTreeNode) cardTree
                    .getLastSelectedPathComponent();                //获得当前选中的名片夹
            String cardName = treeNode.getUserObject().toString();  //获得名片夹名称
            PersonnelDialog personnelDialog = new PersonnelDialog(
                    "添加名片", cardName, -1);                      //创建添加名片的对话框对象
            personnelDialog.setVisible(true);                       //设置添加名片的对话框为可见
            initCardListTable();                                    //刷新名片列表
        }
    }
});
```

“修改”按钮用来修改名片信息，此时只能有一个被选中的名片。此处也将创建 PersonnelDialog 类的对象。当修改名片信息时，可以不设置当前选中的名片夹名称。关键代码如下：

例程 21　代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
updCardButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int[] selectedRows = cardListTable.getSelectedRows();       //获得名片列表中的选中行
        if (selectedRows.length == 1) {                             //仅选中了一个名片
            int num = (Integer) cardListTable.getValueAt(selectedRows[0], 1);   //获得选中名片的编号
            PersonnelDialog personnelDialog = new PersonnelDialog(
                    "修改名片", "", num);                           //创建修改名片的对话框对象
            personnelDialog.setVisible(true);                       //设置修改名片的对话框为可见
            initCardListTable();
        } else {
            if (selectedRows.length == 0) {                         //未选中要修改的名片
                JOptionPane.showMessageDialog(null, "请选择要修改的人员！",
                        "友情提示", JOptionPane.INFORMATION_MESSAGE);   //弹出提示信息
            } else {                                                //选中了多个名片
                JOptionPane.showMessageDialog(null, "一次只能修改一个人员！",
```

```
                                "友情提示", JOptionPane.INFORMATION_MESSAGE);    //弹出提示信息
                }
            }
        }
});
```

“删除”按钮用来删除选中的名片。在执行删除操作前，首先要弹出如图 6.22 所示的提示对话框，询问是否确定要删除选中的名片，目的是增加软件的人性化特点，避免用户错误删除名片信息。

图 6.22 删除名片提示对话框

实现“删除”按钮的关键代码如下：

例程 22 代码位置：光盘\mr\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
delCardButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int[] selectedRows = cardListTable.getSelectedRows();    //获得名片列表中的选中行
        if (selectedRows.length == 0) {                          //未选中要删除的名片
            JOptionPane.showMessageDialog(null, "请选择要删除的人员！", "友情提示",
                    JOptionPane.INFORMATION_MESSAGE);//弹出提示信息
        } else {
            String[] infos = new String[selectedRows.length + 1]; //组织提示信息
            infos[0] = "确定要删除以下人员：";                     //添加提示信息
            for (int i = 0; i < selectedRows.length; i++) {      //遍历选中的名片
                infos[i + 1] = "      " + cardListTable.getValueAt(selectedRows[i], 1)  //获得名片编号
                        + "   " + cardListTable.getValueAt(selectedRows[i], 2);     //获得名片名称
            }
            int i = JOptionPane.showConfirmDialog(null, infos, "友情提示",
                    JOptionPane.YES_NO_OPTION);                  //弹出提示信息
            if (i == 0) {                                        //确定删除
                for (int j = 0; j < selectedRows.length; j++) {  //遍历选中的名片
                    int num = (Integer) cardListTable.getValueAt(selectedRows[j], 1);     //获得名片编号
                    dao.dPersonnelByNum(num);                    //从数据库删除
                }
                initCardListTable();                             //刷新名片列表
            }
        }
    }
});
```

6.8 发送短信模块设计

6.8.1 发送短信模块功能概述

发送短信模块是企业快信系统的核心模块之一，操作员可以通过该模块对企业内部的所有或者部分员工，以短信的形式发送企业通知、工资条、具体技术、开会等信息，并且这个信息是群体发送的。

该模块的运行效果如图 6.23 所示。

图 6.23　企业快信的短信发送界面

6.8.2　发送短信模块技术分析

发送短信模块必须创建 smssend 类的实例对象，该类包含在短信猫设备光盘中提供的 JAR 包中。在使用短信猫设备前，必须调用 smssend 类的 GSMModem InitNew()方法初始化设备，然后调用 GSMModemSMSsend()方法实现短信的发送。

1．GSMModemInitNew()方法

该方法用于初始化短信猫设备，它接收 5 个参数。其语法格式如下：

```
GSMModemInitNew(String device, String baudrate, String initstring, String charset, boolean swHandshake, String sn)
```

该方法的参数说明如表 6.6 所示。

表 6.6　GSMModemInitNew()方法的参数说明

参　数　名	说　　明
device	标识通信端口，如果为 NULL，系统会自动检测
baudrate	标识通信波特率，如果为 NULL，系统会自动检测
initstring	标识初始化命令，为 NULL 即可
charset	标识通信字符集，为 NULL 即可
swHandshake	标识是否进行软件握手，为 FALSE 即可
sn	标识短信猫的授权号，需要根据实际情况填写

2．GSMModemSMSsend()方法

该方法用于发送手机短信。其语法格式如下：

```
GSMModemSMSsend(String serviceCenterAddress, int encodeval, byte[] text, String phonenumber, boolean flag)
```

该方法的参数说明如表 6.7 所示。

表 6.7　GSMModemSMSsend()方法的参数说明

参　数　名	说　　明
serviceCenterAddress	标识短信中心号码，为 NULL 即可
encodeval	标识短信息编码格式，如果为 8，表示中文短信编码
text	标识短信内容
phonenumber	标识接收短信的电话号码
flag	标识状态报告

6.8.3 发送短信模块的实现过程

发送短信模块的开发步骤如下：

（1）创建 InfoPanel 类，该类是发送短信模块和发送邮件模块的面板类，这里主要以介绍发送短信模块的界面为主。在该类的构造方法中需要初始化发送短信模块的界面控件，其中包括发送信息的文本框、“发送”和“清空”按钮。关键代码如下：

例程 23 代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\InfoPanel.java

```
public InfoPanel(MTable sendListTable) {
    super();
    this.sendListTable = sendListTable;                                          //初始化发送列表
    setBorder(new TitledBorder("信息内容"));                                      //设置面板边框
    setLayout(new BorderLayout());                                               //设置布局管理器
    tabbedPane = new JTabbedPane();
    add(tabbedPane, BorderLayout.CENTER);
    final JPanel infoPanel = new JPanel();                                       //初始化短信发送面板
    infoPanel.setLayout(new BorderLayout());                                     //设置面板的布局管理器
    tabbedPane.addTab("短信内容", null, infoPanel, null);                         //设置短信选项卡界面
    final JScrollPane infoScrollPane = new JScrollPane();
    infoPanel.add(infoScrollPane, BorderLayout.CENTER);
    infoTextArea = new JTextArea();                                              //初始化短信编辑文本框
    infoTextArea.setLineWrap(true);                                              //使该文本框支持换行
    infoScrollPane.setViewportView(infoTextArea);                                //为文本框提供滚动面板
    final JPanel infoButtonPanel = new JPanel();                                 //初始化存放按钮的面板
    infoButtonPanel.setLayout(new BoxLayout(infoButtonPanel, BoxLayout.Y_AXIS));//设置面板的布局管理器
    infoPanel.add(infoButtonPanel, BorderLayout.EAST);
    final JButton infoSendButton = new JButton();                                //创建短信发送按钮
    infoSendButton.addActionListener(new InfoSendButtonActionListener());       //为按钮添加事件监听器
❶   infoSendButton.setMargin(new Insets(0, 0, 0, 0));                           //取消按钮的四周边界
    URL sendUrl = this.getClass().getResource("/img/send.png");
❷   infoSendButton.setIcon(new ImageIcon(sendUrl));                             //设置按钮的默认图标
    URL sendOverUrl = this.getClass().getResource("/img/send_over.png");
❸   infoSendButton.setRolloverIcon(new ImageIcon(sendOverUrl));                 //设置鼠标经过时按钮的图标
    infoButtonPanel.add(infoSendButton);                                         //添加发送按钮到面板
    final JButton infoCancelButton = new JButton();                              //创建“清空”按钮
    infoCancelButton.addActionListener(new InfoCancelButtonActionListener());//设置按钮的事件监听器
    infoCancelButton.setMargin(new Insets(0, 0, 0, 0));                          //取消按钮的四周边界
    URL cancelUrl = this.getClass().getResource("/img/cancel.png");
    infoCancelButton.setIcon(new ImageIcon(cancelUrl));                          //设置按钮的默认图标
    URL cancelOverUrl = this.getClass().getResource("/img/cancel_over.png");
    infoCancelButton.setRolloverIcon(new ImageIcon(cancelOverUrl));              //设置鼠标经过时按钮的图标
    infoButtonPanel.add(infoCancelButton);                                       //添加“清空”按钮到面板
…//省略部分代码
}
```

代码贴士

❶ setMargin()：该方法用于设置按钮的四周边界，其 4 个参数分别对应 4 个边界值。

❷ setIcon()：该方法用于设置按钮的图标。

❸ setRolloverIcon(): 该方法用于设置鼠标悬浮于按钮之上时显示的图标。

（2）编写“清空”按钮的事件监听器 InfoCancelButtonActionListener 类，该类实现了 ActionListener 接口和接口中的 actionPerformed()方法，拥有处理按钮事件的能力，在重写的 actionPerformed()方法中完成了短信编辑文本框的清空任务。关键代码如下：

例程 24　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\InfoPanel.java

```
private class InfoCancelButtonActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {                    //处理事件的方法
        infoTextArea.setText(null);                                 //设置文本框内容为 NULL
    }
}
```

（3）编写“发送”按钮的事件监听器 InfoSendButtonActionListener 类，该类也实现了 ActionListener 接口和接口中的 actionPerformed()方法，拥有处理按钮事件的能力。当用户单击“发送”按钮时，该类的 actionPerformed()方法首先遍历 sendListTable 发送列表中的所有用户，并调用 Dao 数据库操作类的 sPersonnelByNum()方法获取用户信息，然后将用户信息封装到 SendLetterForm 类中，最后调用 SendLetterDAO 短信发送类的 sendLetter()方法，为 sendListTable 发送列表中的每个用户分别发送短信信息。关键代码如下：

例程 25　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\InfoPanel.java

```
private class InfoSendButtonActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        int sendCount = sendListTable.getRowCount();                //获取发送列表的用户数量
        for (int i = 0; i < sendCount; i++) {                       //遍历发送列表
            Integer id = (Integer) sendListTable.getValueAt(i, 1);  //获取用户的 ID 编号
            Vector person = dao.sPersonnelByNum(id);                //从数据库获取该用户信息
            String phone = person.get(8).toString();                //获取用户的电话号码
            String info = infoTextArea.getText();                   //获取短信信息
            SendLetterForm form = new SendLetterForm();             //定义短信 Form
            form.setContent(info);                                  //初始化短信内容
            form.setToMan(phone);                                   //初始化短信发送目标
            SendLetterDAO sendDao = new SendLetterDAO();            //初始化短信发送类
            String message = sendDao.sendLetter(form);              //发送短信
            JOptionPane.showMessageDialog(InfoPanel.this, message); //以信息对话框提示发送结果
        }
    }
}
```

（4）创建 SendLetterForm 类。它是一个 JavaBean，是短信群发界面的数据封装类。在该类中分别定义 content 和 toMan 属性，并提供访问这两个属性的方法。关键代码如下：

例程 26　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\form\SendLetterForm.java

```
public class SendLetterForm{
    private String content;                                         //短信内容属性
    private String toMan;                                           //接收短信的手机号
    public String getContent() {                                    //访问短信内容的方法
        return content;
```

```
    }
    public void setContent(String content) {                    //设置短信内容的方法
        this.content = content;
    }
    public String getToMan() {                                  //访问接收短信的手机号
        return toMan;
    }
    public void setToMan(String toMan) {                        //设置接收短信的手机号
        this.toMan = toMan;
    }
}
```

（5）创建 SendLetterDAO 类。它是短信发送类，该类定义了控制短信猫设备的多个方法，其中包括 sendLetter()、getConnectionModem()、mySend()和 closeConnection()方法，分别用于发送短信、获得设备连接、控制短信猫设备和关闭设备连接。除这些方法外，该类还定义了 Preferences 类的实例对象，它可以实现首选项设置的保存，短信猫设备的端口号、波特率和注册码等信息都保存在首选项中。关键代码如下：

例程 27 代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\dao\SendLetterDAO.java

```
public class SendLetterDAO {
    private smssend smssender = null;
❶   private final Preferences perf=Preferences.userRoot();      //首选项类
❷   private String device = perf.get("device","COM1");           //初始化设备端口
    private String baud = perf.get("baud","9600");              //初始化设备波特率
    private String sn = perf.get("sn","YIWU-IJDD-CDQW-JDWG");   //初始化设备的注册码
…//省略部分代码
}
```

代码贴士

❶ userRoot()：该方法用于创建用户根节点的首选项对象。

❷ get()：该方法用于获取首选项中的指定参数。

（6）编写 SendLetterDAO 类的 getConnectionModem()方法，用于初始化短信猫设备，它在短信发送等其他方法之前被调用，如果该方法返回错误信息，那么其他方法将无法执行。关键代码如下：

例程 28 代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\dao\SendLetterDAO.java

```
//初始化 GSM Modem 设备
public boolean getConnectionModem(String device, String baud, String sn) {
    smssender = new smssend();                                  //初始化设备的 session 类
    boolean connection = true;                                  //初始化连接状态
    if (!smssender.GSMModemInitNew(device, baud, null, "GSM", false, sn)) {     //如果初始化设备不成功
        JOptionPane.showMessageDialog(null, "初始化 GSM Modem  设备失败："//在对话框中输出错误信息
                + smssender.GSMModemGetErrorMsg());
        connection = false;                                     //设置初始化设备连接状态
    }
    return connection;                                          //返回设备连接状态
}
```

（7）编写 SendLetterDAO 类的 sendLetter()方法，用于发送短信到目标手机中。该方法接收的参数

是 SendLetterForm 类的实例，并从该实例中获取短信发送信息和目标手机号码，然后调用 mySend()方法发送手机短信到目标手机号中。关键代码如下：

例程 29　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\dao\SendLetterDAO.java

```
public String sendLetter(SendLetterForm form) {
    String ret = "";                                                    //发送结果
    String info = "";                                                   //短信信息
    String sendnum = "";                                                //目标手机号码
    String flag = "";                                                   //标识变量
    try {
        info = form.getContent();                                       //获取短信内容
        sendnum = form.getToMan();                                      //初始化目标手机号码
        flag = mySend(device, baud, sn, info, sendnum);                 //调用 mySend() 方法发送短信
        if (flag.equals("ok")) {
        } else {                                                        //如果返回结果不是“ok”信息
            ret = flag;                                                 //初始化返回结果
        }
    } catch (Exception e) {                                             //如果出现异常
        System.out.println("发送短信产生的错误：" + e.getMessage());     //输出错误信息
        ret = "发送短信失败！";                                          //初始化发送结果为失败信息
    }
    return ret;
}
```

（8）编写 SendLetterDAO 类的 mySend()方法，它也实现发送短信到目标手机中，但是该方法是被 sendLetter()方法调用的私有方法，方法的主体首先调用 getConnectionModem()方法初始化短信猫设备，如果初始化成功，则实现短信群发。关键代码如下：

例程 30　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\dao\SendLetterDAO.java

```
private String mySend(String device, String baud, String sn, String info,
        String sendnum) {
    boolean flag = false;                                               //初始化标识变量
    String rtn = "";
    flag = this.getConnectionModem(device, baud, sn);                   //初始化设备
    if (flag) {
❶       byte[] sendtest = smssender.getUNIByteArray(info);              //转化为 UNICOCE
        String[] arrSendnum = sendnum.split(",");
        for (int i = 0; i < arrSendnum.length; i++) {                   //实现群发
❷           if (!smssender.GSMModemSMSsend(null, 8, sendtest,
                    arrSendnum[i], false)) {                            //发送短信
                System.out.println("发送短信失败："
❸                       + smssender.GSMModemGetErrorMsg());            //输出发送错误信息
                rtn = rtn + "向" + arrSendnum[i] + "发送短信失败!<br>原因是："
                        + smssender.GSMModemGetErrorMsg() + "<br>";
            }
        }
    } else {
        rtn = "初始化 GSM Modem 设备失败！";
    }
```

```
        if (rtn.equals("")) {
            rtn = "ok";
        }
        closeConnection();                                          //关闭连接
        return rtn;
    }
```

代码贴士

❶ getUNIByteArray()：该方法用于将文本字符串转换为 Java 的 UNICODE 字节数组。

❷ GSMModemSMSsend()：该方法用于执行发送短信任务。

❸ GSMModemGetErrorMsg()：该方法用于获取错误信息。

（9）编写 SendLetterDAO 类的 closeConnection()方法，用于关闭短信设备的连接。当短信发送业务执行完毕，必须显示释放连接短信设备所占用的系统资源，减少资源浪费，以提高系统运行速度和稳定性。关键代码如下：

例程 31 代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\dao\SendLetterDAO.java

```
//关闭连接的方法
public void closeConnection() {
    if (smssender != null) {                                        //如果 smssender 对象仍在连接
        smssender.GSMModemRelease();                                //就释放占用的资源
        System.out.println("关闭成功！！！");
    }
}
```

6.9 发送邮件模块设计

6.9.1 发送邮件模块功能概述

发送邮件模块也是企业快信系统的核心模块之一，操作员可以通过该模块对企业内部的所有或者部分员工，以 E-mail 电子邮件的形式发送企业通知、工资条等。另外，由于 E-mail 有信息容量大、包含附件的两大优点，使它支持更多的信息内容，可以携带更丰富的资料，比较适合发送技术信息、会议主题、工作内容等信息，并且这个信息是群体发送的。该模块的运行效果如图 6.24 所示。

图 6.24 企业快信的邮件发送界面

6.9.2　发送邮件模块技术分析

发送邮件模块和发送短信模块的界面部分都定义在 InfoPanel 类中。这两个模块都需要读取发送信息的用户列表，该列表是自定义的 MTable 类的实例，它定义在 TipWizardFrame 类中，该类负责定义程序的主界面，为获得 MTable 类的实例 sendListTable 对象，必须为 InfoPanel 类的构造方法提供该对象的参数引用，然后 InfoPanel 类在发送邮件或者短信时，从该对象中获取接收信息的用户列表。关键代码如下：

```
public InfoPanel(MTable sendListTable)
```

6.9.3　发送邮件模块的实现过程

发送邮件模块的开发步骤如下：

（1）在 InfoPanel 类的构造方法中初始化邮件群发界面中的控件，这些控件包括邮件标题文本框、邮件内容文本框、“发送”按钮、“清空”按钮、“添加附件”按钮、“删除附件”按钮等。关键代码如下：

例程 32　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\InfoPanel.java

```
public InfoPanel(MTable sendListTable) {
…//省略部分代码
    final JPanel emailPanel = new JPanel();                                    //创建界面面板
    emailPanel.setLayout(new BorderLayout());                                  //设置布局管理器
    tabbedPane.addTab("E-mail 内容", null, emailPanel, null);
    final JPanel emailInfoPanel = new JPanel();
    emailInfoPanel.setLayout(new BorderLayout());
    emailPanel.add(emailInfoPanel);
    final JScrollPane emailScrollPane = new JScrollPane();
    emailInfoPanel.add(emailScrollPane);
    emailTextArea = new JTextArea();                                           //创建邮件编辑文本框
    emailTextArea.setLineWrap(true);                                           //设置文本框支持换行
    emailScrollPane.setViewportView(emailTextArea);
    panel.setLayout(new BorderLayout());
    emailInfoPanel.add(panel, BorderLayout.NORTH);
    panel.add(new JLabel("邮件标题："), BorderLayout.WEST);                    //添加邮件标题标签
    panel.add(titleField);
    final JPanel annexPanel = new JPanel();                                    //定义附件面板
    annexPanel.setLayout(new FlowLayout());
    emailInfoPanel.add(annexPanel, BorderLayout.SOUTH);
    annexLabel = new JLabel("附件：");                                         //初始化附件标签
    annexPanel.add(annexLabel);
    final JPanel emailButtonPanel = new JPanel();                              //创建按钮面板
    emailButtonPanel.setLayout(new BoxLayout(emailButtonPanel, BoxLayout.Y_AXIS));
    emailPanel.add(emailButtonPanel, BorderLayout.EAST);
    final JButton emailSendButton = new JButton();                             //创建“发送”按钮
    emailSendButton.addActionListener(new EmailSendButtonActionListener());    //添加按钮事件监听器
```

```
        emailSendButton.setMargin(new Insets(0, 0, 0, 0));                    //取消按钮的四周边界
        URL emailSendUrl = this.getClass().getResource("/img/send.png");      //获取按钮图标路径
        emailSendButton.setIcon(new ImageIcon(emailSendUrl));                 //设置按钮默认图标
        URL emailSendOverUrl = this.getClass().getResource("/img/send_over.png"); //获取按钮图标路径
        emailSendButton.setRolloverIcon(new ImageIcon(emailSendOverUrl));     //设置鼠标经过按钮时的图标
        emailButtonPanel.add(emailSendButton);
        final JButton emailCancelButton = new JButton();                      //创建“清空”按钮
        emailCancelButton.setMargin(new Insets(0, 0, 0, 0));                  //取消按钮的四周边界
        URL emailCancelUrl = this.getClass().getResource("/img/cancel.png");  //获取按钮图标路径
        emailCancelButton.setIcon(new ImageIcon(emailCancelUrl));             //设置按钮默认图标
        URL emailCancelOverUrl = this.getClass().getResource("/img/cancel_over.png");//获取按钮图标路径
        emailCancelButton.setRolloverIcon(new ImageIcon(emailCancelOverUrl));//设置鼠标经过按钮时的图标
        emailButtonPanel.add(emailCancelButton);
        final JButton addAnnexButton = new JButton();                         //创建添加附件按钮
        addAnnexButton.addActionListener(new AddAnnexButtonActionListener());//添加附件的事件监听器
        addAnnexButton.setMargin(new Insets(0, 0, 0, 0));                     //取消按钮的四周边界
        URL addAnnexUrl = this.getClass().getResource("/img/add_annex.png");  //获取按钮图标路径
        addAnnexButton.setIcon(new ImageIcon(addAnnexUrl));                   //设置按钮默认图标
        URL addAnnexOverUrl = this.getClass().getResource("/img/add_annex_over.png");
        addAnnexButton.setRolloverIcon(new ImageIcon(addAnnexOverUrl));       //设置鼠标经过按钮的图标
        emailButtonPanel.add(addAnnexButton);
        final JButton cancelAnnexButton = new JButton();
        cancelAnnexButton.addActionListener(new CancelAnnexButtonActionListener());//删除附件的事件监听器
        cancelAnnexButton.setMargin(new Insets(0, 0, 0, 0));
        URL delAnnexUrl = this.getClass().getResource("/img/del_annex.png");
        cancelAnnexButton.setIcon(new ImageIcon(delAnnexUrl));
        URL delAnnexOverUrl = this.getClass().getResource("/img/del_annex_over.png");
        cancelAnnexButton.setRolloverIcon(new ImageIcon(delAnnexOverUrl));
        emailButtonPanel.add(cancelAnnexButton);
    }
```

（2）创建“发送”按钮的事件监听器 EmailSendButtonActionListener 类，该类实现了 ActionListener 接口和接口中的 actionPerformed()方法，拥有处理按钮事件的能力。当用户单击“E-mail 内容”选项卡上的“发送”按钮时，将调用该类的 actionPerformed()方法处理邮件发送业务。该方法首先获取用户输入的邮件标题、邮件内容，然后遍历发送列表中的所有用户，连接这些用户的 E-mail 地址，E-mail 地址之间以“,”符号分隔。最后调用 sendMail()方法完成邮件群发业务。关键代码如下：

例程 33 代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\InfoPanel.java

```
private class EmailSendButtonActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String title = titleField.getText();                                  //获取邮件标题
        String content = emailTextArea.getText();                             //获取邮件内容
        if (title.isEmpty() || content.isEmpty()) {                           //验证邮件标题和内容
            JOptionPane.showMessageDialog(InfoPanel.this, "请填写邮件标题和内容");
            return;
        }
        int sendCount = sendListTable.getRowCount();                          //获取邮件群发总数
        for (int i = 0; i < sendCount; i++) {                                 //遍历发送列表
```

```
                Integer id = (Integer) sendListTable.getValueAt(i, 1);          //获取用户 ID
                Vector person = dao.sPersonnelByNum(id);                        //从数据库获取该 ID 用户
                String addressee = person.get(9).toString();                    //获取用户的 E-mail 地址
                mailForm.setTitle(title);                                       //初始化 mailForm 实例
                mailForm.setContent(content);
                if (mailForm.getAddressee() == null
                        || mailForm.getAddressee().isEmpty())
                    mailForm.setAddressee(addressee);
                else
                    mailForm.setAddressee(mailForm.getAddressee() + ','
                            + addressee);
            }
            int res = mailDao.sendMail(mailForm);                               //发送邮件
            String message = null;
            if (res == 1)                                                       //判断发送结果
                message += "E-mail 群体发送成功";
            else
                message += "E-mail 发送失败";
            JOptionPane.showMessageDialog(InfoPanel.this, message);             //提示发送结果
            mailForm = new SendMailForm();                                      //初始化 mailForm 对象
        }
    }
```

（3）创建“添加附件”按钮的事件监听器 AddAnnexButtonActionListener 类，该类实现了 ActionListener 接口和接口中的 actionPerformed()方法。当用户单击“添加附件”按钮时，将调用 actionPerformed()方法实现相应的业务逻辑。该方法首先创建文件选择对话框，并获取用户选择的文件，然后判断用户是否选择了文件，如果没选择，则终止该方法，不做附件添加处理。最后，如果用户选择了作为附件的文件，就把该文件的路径添加到 mailForm 对象中，该对象将被传送到邮件发送方法中。关键代码如下：

例程 34　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\InfoPanel.java

```
private class AddAnnexButtonActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        JFileChooser fc = new JFileChooser();                                   //创建文件选择对话框
        fc.showOpenDialog(InfoPanel.this);                                      //打开对话框
        File sFile = fc.getSelectedFile();                                      //获取用户选择的文件
        if(sFile==null)                                                         //判断是否选择文件
            return;
        mailForm.setAdjunct(sFile.getAbsolutePath());                           //保存文件路径
        annexLabel.setText("附件：" + sFile.getName());                          //设置界面的附件信息
    }
}
```

（4）创建“删除附件”按钮的事件监听器 CancelAnnexButtonActionListener 类，该类实现了 ActionListener 接口和接口中的 actionPerformed()方法。当用户单击“删除附件”按钮时，将调用 actionPerformed()方法去处理删除附件的业务逻辑，附件由“添加附件”按钮添加文件路径到 mailForm

对象中，actionPerformed()方法所要实现的就是把附件的文件路径从 mailForm 对象中删除，同时更新界面上的附件信息。关键代码如下：

例程 35 代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\InfoPanel.java

```
private class CancelAnnexButtonActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        mailForm.setAdjunct("");                                    //清除附件路径
        annexLabel.setText("附件：");                               //初始化附件标签
    }
}
```

（5）创建 SendMailForm 类。它是一个 JavaBean，是邮件群发界面的数据封装类。在该类中分别定义 content、title、addressee 和 adjunct 属性，并提供访问这 4 个属性的方法。关键代码如下：

例程 36 代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\form\SendMailForm.java

```
public class SendMailForm {
    private String content;                                         //邮件内容
    private String title;                                           //邮件标题
    private String addressee;                                       //目标 E-mail 地址
    private String adjunct;                                         //附件路径
    public String getContent() {                                    //访问 content 属性的方法
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public String getTitle() {                                      //访问 title 属性的方法
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getAddressee() {                                  //访问 addressee 属性的方法
        return addressee;
    }
    public void setAddressee(String addressee) {
        this.addressee = addressee;
    }
    public String getAdjunct() {                                    //访问 adjunct 属性的方法
        return adjunct;
    }
    public void setAdjunct(String adjunct) {
        this.adjunct = adjunct;
    }
}
```

（6）编写 SendMailDAO 类，用于实现邮件群发业务。该类定义了一个 sendMail()方法，这个方法首先获取发送邮件的设置信息，这些信息包括服务器、用户、密码等，另外该方法从 SendMailForm 类

中获取邮件发送信息，包括邮件标题、邮件内容、附件等信息，然后使用 JavaMail 技术实现邮件的发送业务。关键代码如下：

例程 37　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\dao\SendMailDAO.java

```
public class SendMailDAO {
    private final Preferences perf=Preferences.userRoot();
    public int sendMail(SendMailForm form) {
        int ret = 0;
        String from =perf.get("mailName","");                          //获取邮件服务器
        String to = form.getAddressee();                               //获取目标邮件地址
        String subject = form.getTitle();                              //获取邮件标题
        String content = form.getContent();                            //获取邮件内容
        String password = perf.get("password","");                     //获取密码
        String path = form.getAdjunct();                               //获取附件
        try {
            String mailserver = perf.get("SMTPserver", "LZW");         //在局域网内发送邮件的代码
            Properties prop = new Properties();
            prop.put("mail.smtp.host", mailserver);
            prop.put("mail.smtp.auth", "true");
❶           Session sess = Session.getDefaultInstance(prop);
❷           sess.setDebug(true);                                       //开启调试模式
❸           MimeMessage message = new MimeMessage(sess);
            message.setFrom(new InternetAddress(from));                //给消息对象设置发件人
            String toArr[]=to.split(",");                              //设置收件人
            InternetAddress[] to_mail=new InternetAddress[toArr.length];
            for(int i=0;i<toArr.length;i++){
                to_mail[i]=new InternetAddress(toArr[i]);
            }
           message.setRecipients(Message.RecipientType.BCC,to_mail);
            message.setSubject(subject);                               //设置主题
            Multipart mul = new MimeMultipart();                       //新建 MimeMultipart 对象
            BodyPart mdp = new MimeBodyPart();                         //新建存放信件内容对象
            mdp.setContent(content, "text/html;charset=gb2312");
            mul.addBodyPart(mdp);                                      //将信件内容加入到 MimeMulitipart 对象中
            if (path != null && !path.isEmpty() ){                     //当存在附件时
                mdp = new MimeBodyPart();                              //新建存放附件的 BodyPart
                String adjunctname = new String(path.getBytes("GBK"), "ISO-8859-1"); //将路径转码
                FileDataSource fds = new FileDataSource(path);
                DataHandler handler = new DataHandler(fds);
                mdp.setFileName(adjunctname);
                mdp.setDataHandler(handler);
                mul.addBodyPart(mdp);
            }
            message.setContent(mul);                                   //把 mul 作为消息对象内容
            message.saveChanges();
❹           Transport transport = sess.getTransport("smtp");           //以 smtp 方式登录邮箱
            transport.connect(mailserver, from, password);
            transport.sendMessage(message, message.getAllRecipients());
            transport.close();
            ret = 1;
```

```
            } catch (Exception e) {
                System.out.println("发送邮件产生的错误：" + e.getMessage());
                ret = 0;
            }
            return ret;
        }
}
```

代码贴士

❶ Session：该类是定义保存诸如 SMTP 主机和认证信息的基本邮件会话。

❷ setDebug(true)：该方法将设置程序为调试模式，在控制台将输出调试信息。

❸ MimeMessage：Message 类的子类，它是电子邮件系统的核心类，用于存储实际发送的电子邮件信息。

❹ Transport：该类将使用指定的协议（通常是 SMTP）发送电子邮件。

6.10 系统设置模块设计

6.10.1 系统设置模块功能概述

企业快信的系统设置模块包括短信设置和邮件设置两部分，本节将介绍这两部分功能对本系统的意义和实现的业务逻辑。

1．短信设置

本系统的发送手机短信模块需要使用短信猫设备，该设备必须设置相应的参数信息，包括设备的通信端口、波特率和短信猫设备的注册编码，它们将被 SendLetterDAO 类调用，以完成短信猫的初始化和短信发送功能。短信猫设置界面如图 6.25 所示。

2．邮箱设置

本系统的邮件发送模块也需要相应的参数设置才能实现邮件的发送，这些参数包括 SMTP 服务器的名称或者 IP 地址、登录服务器的用户名、登录服务器的密码，它们将被 SendMailDAO 类调用，以完成邮件群发的功能。邮箱设置界面如图 6.26 所示。

图 6.25 短信猫设置界面

图 6.26 邮箱设置界面

6.10.2 系统设置模块技术分析

虽然系统设置模块用于保存企业快信系统的参数设置，但是本模块并没有使用任何数据表，而是

采用 Preferences 类保存相应的参数信息。该类由 Java API 提供，其常用方法如表 6.8 所示。

表 6.8　Preferences 类的常用方法

方　法	说　明
systemRoot()	返回系统的根首选项节点
userRoot()	返回调用用户的根首选项节点
put(String key, String value)	以 Key 名称保存 value 值到首选项中
get(String key, String def)	获取首选项中 key 名称的存储值，如果没有该值，则采用默认的 def 值

6.10.3　短信设置的实现过程

短信设置的开发步骤如下：

（1）创建 LetterSetDialog 类，该类需要继承 JDialog 类成为一个对话框窗体，在该窗体中创建设置短信猫参数的各个文本框控件和相应的标签以及按钮控件，另外还需要创建 Preferences 首选项类的实例。关键代码如下：

例程 38　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\explorer\LetterSetDialog.java

```
public class LetterSetDialog extends JDialog {
    private final Dao dao = Dao.getInstance();                          //访问数据库的 Dao 实例
    private final JTextField deviceField = new JTextField();            //通信端口
    private final JTextField baudField = new JTextField();              //波特率
    private final JTextField snField = new JTextField();                //注册编码
    private final Preferences perf=Preferences.userRoot();              //首选项
…//省略部分代码
}
```

（2）在该类的构造方法中初始化窗体上所有控件，并布局这些控件的摆放位置，然后获取 Preferences 首选项中的参数值，使用该值初始化窗体文本框控件的初始值。关键代码如下：

例程 39　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\explorer\LetterSetDialog.java

```
public LetterSetDialog() {
    setTitle("短信猫设置");                                              //设置窗体标题
    setBounds(100, 100, 347, 204);                                      //设置窗体位置和大小
    final JPanel setPanel = new JPanel();
    setPanel.setBorder(new EmptyBorder(20, 0, 10, 10));
    final GridLayout gridLayout = new GridLayout(0, 2)                  //设置布局管理器
    setPanel.setLayout(gridLayout);
    getContentPane().add(setPanel, BorderLayout.CENTER);
    JLabel label = new JLabel("通讯端口：");                             //界面的标签控件
    JLabel label_2 = new JLabel("波特率 ：");
    JLabel label_3 = new JLabel("注册码：");
    String device = perf.get("device","COM1");                          //获取首选项的通信端口
    String baud = perf.get("baud","9600");                              //获取波特率
    String sn = perf.get("sn","YIWU-IJDD-****-****");                   //获取注册编码
    setPanel.add(label);                                                //添加标签到窗体
```

```
        label.setHorizontalAlignment(SwingConstants.CENTER);                    //设置标签居中
        setPanel.add(deviceField);                                              //添加“通讯端口”文本框到窗体
        deviceField.setText(device);                                            //初始化文本框内容
        setPanel.add(label_2);                                                  //添加标签到窗体
        label_2.setHorizontalAlignment(SwingConstants.CENTER);                  //设置标签居中
        setPanel.add(baudField);                                                //添加“波特率”文本框到窗体
        baudField.setText(baud);                                                //初始化文本框内容
        setPanel.add(label_3);                                                  //添加标签到窗体
        label_3.setHorizontalAlignment(SwingConstants.CENTER);                  //设置标签居中
        setPanel.add(snField);                                                  //添加“注册码”文本框到窗体
        snField.setText(sn);                                                    //初始化文本框内容
        final JPanel buttonP anel = new JPanel();                               //添加标签到窗体
        final FlowLayout flo wLayout = new FlowLayout();                        //设置标签居中
        flowLayout.setAlignment(FlowLayout.RIGHT);
        buttonPanel.setLayout(flowLayout);
        getContentPane().add(buttonPanel, BorderLayout.SOUTH);
        final JButton submitButton = new JButton("确定");                       //创建“确定”按钮
        submitButton.addActionListener(new OKActionListener());                 //添加按钮监听器
        buttonPanel.add(submitButton);                                          //添加“确定”按钮到窗体
        final JButton exitButton = new JButton("退出");                         //创建“退出”按钮
        exitButton.addActionListener(new ExitActionListener());                 //添加按钮监听器
        buttonPanel.add(exitButton);                                            //添加按钮到窗体
        final JLabel leftLabel = new JLabel();
        leftLabel.setPreferredSize(new Dimension(10, 0));
        getContentPane().add(leftLabel, BorderLayout.WEST);
        final JLabel rightLabel = new JLabel();
        rightLabel.setPreferredSize(new Dimension(10, 0));
        getContentPane().add(rightLabel, BorderLayout.EAST);
}
```

（3）编写“确定”按钮的事件监听器 OKActionListener 类，它实现了 ActionListener 接口和接口中的 actionPerformed()方法。当用户单击“确定”按钮时，将激活 actionPerformed()方法去处理相应的业务逻辑。该方法首先获取界面中各文本框的内容，然后调用 put()方法，将所有设置保存，最后以对话框提示用户“保存完毕”，并隐藏该对话框。关键代码如下：

例程 40 代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\explorer\LetterSetDialog.java

```
class OKActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String device = deviceField.getText();                                  //获取通信端口
        String baud = baudField.getText();                                      //获取波特率
        String sn = snField.getText();                                          //获取注册编码
        perf.put("device",device);                                              //保存通信端口
        perf.put("baud",baud);                                                  //保存波特率
        perf.put("sn",sn);                                                      //保存注册编码
        JOptionPane.showMessageDialog(LetterSetDialog.this, "保存完毕");//提示信息
        setVisible(false);                                                      //隐藏窗体
    }
}
```

6.10.4　邮箱设置的实现过程

邮箱设置的开发步骤如下：

（1）创建 MailSetDialog 类，它必须继承 JDialog 类成为一个对话框窗体，在该窗体中定义 Preferences 首选项和设置邮箱信息的各文本框，这些文本框分别对应 SMTP 服务器、登录服务器的用户名和密码。关键代码如下：

例程 41　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\explorer\MailSetDialog

```
public class MailSetDialog extends JDialog {
    private final Dao dao = Dao.getInstance();                          //获取 dao 实例
    private final JTextField mailAddress = new JTextField();            //创建登录 SMTP 的用户文本框
    private final JPasswordField mailPassword = new JPasswordField();   //创建“密码”文本框
    private final Preferences perf = Preferences.userRoot();            //创建首选项
    private final JLabel label_2 = new JLabel();
    private final JTextField SMTPAddress = new JTextField();            //创建“SMTP 服务器”文本框
…//省略部分代码
}
```

（2）在该类的构造方法中初始化窗体上所有控件，并布局这些控件的摆放位置，然后获取 Preferences 首选项中的参数值，使用该值初始化窗体文本框控件的初始值。关键代码如下：

例程 42　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\explorer\MailSetDialog

```
public MailSetDialog() {
    setTitle("邮箱设置");                                               //设置窗体标题
    setBounds(100, 100, 347, 213);                                      //设置窗体大小和位置
    final JPanel setPanel = new JPanel();
    setPanel.setBorder(new EmptyBorder(30, 0, 20, 10));
    final GridLayout gridLayout = new GridLayout(0, 2);
    setPanel.setLayout(gridLayout);                                     //设置布局管理器
    getContentPane().add(setPanel, BorderLayout.CENTER);
    String mailName = perf.get("mailName", "mybook@***.com");           //获取首选项中的用户名信息
    String password = perf.get("password", "111");                      //密码信息
    String SMTPserver = perf.get("SMTPserver", "LZW");                  //SMTP 服务器信息
    setPanel.add(label_2);
    label_2.setHorizontalAlignment(SwingConstants.CENTER);
    label_2.setText("SMTP 服务器：");
    setPanel.add(SMTPAddress);                                          //添加“SMTP 服务器”文本框到窗体
    SMTPAddress.setText(SMTPserver);
    JLabel label = new JLabel();
    JLabel label_3 = new JLabel();
    setPanel.add(label);
    label.setHorizontalAlignment(SwingConstants.CENTER);
    label.setText("用户名：");
    setPanel.add(mailAddress);                                          //添加“用户名”文本框到窗体中
    mailAddress.setText(mailName);
    setPanel.add(label_3);
```

```
        label_3.setHorizontalAlignment(SwingConstants.CENTER);
        label_3.setText("密码：");
        setPanel.add(mailPassword);                                    //添加“密码”文本框到窗体中
        mailPassword.setText(password);
        final JPanel buttonPanel = new JPanel();
        final FlowLayout flowLayout = new FlowLayout();
        flowLayout.setAlignment(FlowLayout.RIGHT);
        buttonPanel.setLayout(flowLayout);
        getContentPane().add(buttonPanel, BorderLayout.SOUTH);
        final JButton submitButton = new JButton("确定");              //创建“确定”按钮
        submitButton.addActionListener(new OKActionListener());        //添加按钮事件监听器
        buttonPanel.add(submitButton);                                 //添加按钮到窗体
        final JButton exitButton = new JButton("退出");                //创建“退出”按钮
        exitButton.addActionListener(new ExitActionListener());        //添加按钮事件监听器
        buttonPanel.add(exitButton);                                   //添加按钮到窗体
        final JLabel leftLabel = new JLabel();
        leftLabel.setPreferredSize(new Dimension(10, 0));
        getContentPane().add(leftLabel, BorderLayout.WEST);
        final JLabel rightLabel = new JLabel();
        rightLabel.setPreferredSize(new Dimension(10, 0));
        getContentPane().add(rightLabel, BorderLayout.EAST);
    }
```

（3）创建“确定”按钮的事件监听器 OKActionListener 类，它实现了 ActionListener 接口和接口中的 actionPerformed()方法。当用户单击“确定”按钮时，将激活 actionPerformed()方法去处理相应的业务逻辑。该方法首先获取界面中各文本框的内容，然后调用 put()方法，将所有设置保存，最后以对话框提示用户“保存完毕”，并隐藏该对话框。关键代码如下：

例程 43 代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\explorer\MailSetDialog

```
class OKActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String maild = mailAddress.getText();                          //获取用户名
        String password = new String(mailPassword.getPassword());      //获取密码
        String SMTPserver = SMTPAddress.getText();                     //获取 SMTP 服务器
        perf.put("mailName", maild);                                   //保存所有信息
        perf.put("password", password);
        perf.put("SMTPserver", SMTPserver);
        JOptionPane.showMessageDialog(MailSetDialog.this, "保存完毕"); //提示信息
        setVisible(false);                                             //隐藏窗体
    }
}
```

6.11 开发技巧与难点分析

在将信息库中选中的信息添加到信息内容中时，需要判断当前要发送的是手机短信还是 E-mail。判断的方法是通过 JTabbedPane 类的 getSelectedIndex()方法，该方法将返回当前被选中的选项卡的索引

值，位于最左侧的选项卡的索引值为 0，选项卡的索引值向右依次加 1。如图 6.27 所示，“短信内容”选项卡的索引值为 0，“E-mail 内容”选项卡的索引值为 1。

图 6.27　选项卡

实现将信息库中选中的信息添加到信息内容中的关键代码如下：

例程 44　代码位置：光盘\TM\06\ExpressLetter\src\com\mwq\frame\ExplorerPanel.java

```
addToSendInfoButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int selectedRow = infoListTable.getSelectedRow();                    //获得信息列表的选中行
        if (selectedRow < 0) {                                               //未选择任何行
            JOptionPane.showMessageDialog(null, "请选择要编辑的信息！", "友情提示",
                    JOptionPane.INFORMATION_MESSAGE);                        //弹出提示信息
        } else {
            String info = infoListTable.getValueAt(selectedRow, 2).toString();     //获得信息内容
            if (infoTabbedPane.getSelectedIndex() == 0)                      //当前被选中的是“短信内容”面板
                infoTextArea.setText(info);                                  //添加信息到短信内容文本区域
            else                                                             //当前被选中的是“E-mail 内容”面板
                emailTextArea.setText(info);                                 //添加信息到 E-mail 内容文本区域
        }
    }
});
```

6.12　使用短信猫和 Java Mail 组件

6.12.1　使用短信猫

短信猫又名 GSM Modem，专门针对短信应用设计，内含工业级短信发送模块，简化了通信接口，性能稳定可靠，符合各种商业和工业级短信应用要求，支持向移动、联通以及小灵通用户收发短信，适用于各行各业多领域的无线数据通信、短信息通告、短信息查询等应用。

本书中使用的是北京人大金仓信息技术有限公司的串口短信猫。在购买短信猫时会附带包含 SDK 的开发包，其中提供了操作短信猫的方法。下面介绍操作短信猫的主要方法。

（1）获取短信猫注册时所需信息的方法 GSMModemGetSnInfoNew()。其语法格式如下：

```
public native String GSMModemGetSnInfoNew(String device,String baudrate)
```

其参数及返回值说明如表 6.9 所示。

表 6.9　GSMModemGetSnInfoNew()方法参数与返回值说明

参 数 名 称	参 数 类 型	参 数 说 明
device	String 型	用于指定通信端口号，为 null 时系统会自动检测
baudrate	String 型	用于指定通信波特率，为 null 时系统会自动检测
返回值	String 型	短信标识码，将此号码发送给厂商即可获得正式的授权码

（2）初始化 GSM Modem 设备的方法 GSMModemInitNew()。其语法格式如下：

```
public native boolean GSMModemInitNew(String device,String baudrate,String initstring,String charset,boolean swHandshake,String sn)
```

其参数及返回值说明如表 6.10 所示。

表 6.10　GSMModemInitNew()方法参数与返回值说明

参 数 名 称	参 数 类 型	参 数 说 明
device	String 型	用于指定通信端口号，为 null 时系统会自动检测
baudrate	String 型	用于指定通信波特率，为 null 时系统会自动检测
initstring	String 型	用于指定 at 初始化命令，设为 null，系统默认即可
charset	String 型	用于指定通信字符集，设为 null，系统默认即可
swHandshake	boolean 型	用于指定是否进行软件握手，设为 false 即可
Sn	String 型	用于指定通信许可证书，区分大小写。例如 REEE-IVKD-VKTZ-VDZB
返回值	boolean 型	true 为成功，false 连接失败

（3）发送短信的方法 GSMModemSMSsend()。其语法格式如下：

```
public native boolean GSMModemSMSsend(String serviceCenterAddress,int codeval,byte [] text,String phonenumber, boolean requestStatusReport)
```

其参数及返回值说明如表 6.11 所示。

表 6.11　GSMModemSMSsend()方法参数与返回值说明

参 数 名 称	参 数 类 型	参 数 说 明
serviceCenterAddress	String 型	用于指定短信中心号码
codeval	int 型	用于指定文本编码格式，0-7bit，4-8bit，8-16bit
text	byte[]型	用于指定短信内容的 UNICODE 字节数组
phonenumber	String 型	用于指定接收电话号码
requestStatusReport	boolean 型	用于指定状态报告，一般不进行状态报告
返回值	boolean 型	True 代表发送成功，false 代表发送失败

（4）获取当前的通信端口的方法 GSMModemGetDevice()。其语法格式如下：

```
public native String GSMModemGetDevice();
```

返回值：String 型，返回端口名称，如 COM1。

（5）获取当前的通信波特率的方法 GSMModemGetBaudrate()。其语法格式如下：

```
public native String GSMModemGetBaudrate();
```

返回值：String 型，返回波特率，如 9600。

（6）获取当前的连接状态的方法 GSMModemIsConn()。其语法格式如下：

```
public native boolean GSMModemIsConn();
```

返回值：boolean 型，返回系统是否连接，true 为正在连接，false 为未连接。

（7）断开连接，并释放资源的方法 GSMModemRelease()。其语法格式如下：

```
public native void GSMModemRelease();
```

（8）将文本字符串转换为 Java 的 UNICODE 字节数组的方法 getUNIByteArray()。其语法格式如下：

```
public static byte[] getUNIByteArray(String instr)
```

☑ instr：String 型，用于指定短信文本字符串。
☑ 返回值：byte[]型，转换后的 Java 的 UNICODE 字节数组。

（9）将 Java 的字符串转换为十六进制的字符串的方法 bufToHex()。其语法格式如下：

```
public static String bufToHex(byte[] temp);
```

☑ temp：byte[]型，用于指定 UNICODE 字节数组。
☑ 返回值：String 型，转换后的十六进制的短信文本字符串。

（10）将十六进制字符串转换为 Java 的字符串的方法 HexToBuf()。其语法格式如下：

```
public static String HexToBuf(String text) ;
```

☑ text：String 型，用于指定十六进制字符串。
☑ 返回值：String 型，转换后的 Java 短信文本字符串。

（11）从手机中读出错误信息的方法 GSMModemGetErrorMsg()。其语法格式如下：

```
public native String GSMModemGetErrorMsg();
```

返回值：错误说明文字。

6.12.2　使用 Java Mail 组件

Java Mail 是 Sun 公司发布用来处理 E-mail 的 API，是一种可选的、用于读取、编写和发送电子消息的包（标准扩展）。使用 Java Mail 可以创建 MUA（Mail User Agent，邮件用户代理的简称）类型的程序，它类似于 Eudora、Pine 及 Microsoft Outlook 等邮件程序。其主要目的不是像发送邮件或提供 MTA（Mail Transfer Agent，邮件传输代理的简称）类型程序那样用于传输、发送和转发消息，而是可以与 MUA 类型的程序交互，以阅读和撰写电子邮件。MUA 依靠 MTA 处理实际的发送任务。

Java Mail API 中提供很多用于处理 E-mail 的类，其中比较常用的有 Session（会话）、Message（消息）、Address（地址）、Authenticator（认证方式）、Transport（传输）、Store（存储）和 Folder（文件夹）7 个类。这 7 个类都可以在 Java Mail API 的核心包 mail.jar 中找到，下面进行详细介绍。

1．Session 类

Java Mail API 中提供了 Session 类，用于定义保存诸如 SMTP 主机和认证的信息的基本邮件会话。通过 Session 可以阻止恶意代码窃取其他用户在会话中的信息（包括用户名和密码等认证信息），从而让其他工作顺利执行。

每个基于 Java Mail 的程序都需要创建一个 Session 或多个 Session 对象。由于 Session 对象利用 java.util.Properties 对象获取诸如邮件服务器、用户名、密码等信息，以及其他可在整个应用程序中共享的信息，所以在创建 Session 对象前，需要先创建 java.util.Properties 对象的实例。创建 java.util.Properties 对象的实例的代码如下：

```
Properties props=new Properties();
```

创建 Session 对象可以通过以下两种方法，不过通常情况下会使用第二种方法创建共享会话。

（1）使用静态方法创建 Session 的语句如下：

```
Session session = Session.getInstance(props, authenticator);
```

说明 props 为 java.util. Properties 类的对象；authenticator 为 Authenticator 对象，用于指定认证方式。

（2）创建默认的共享 Session 的语句如下：

```
Session defaultSession = Session.getDefaultInstance(props, authenticator);
```

如果在进行邮件发送时，不需要指定认证方式，可以使用空值（null）作为参数 authenticator 的值。例如创建一个不需要指定认证方式的 Session 对象的代码如下：

```
Session mailSession=Session.getDefaultInstance(props,null);
```

2．Message 类

Message 类是电子邮件系统的核心类，用于存储实际发送的电子邮件信息。Message 类是一个抽象类，要使用该抽象类可以使用其子类 MimeMessage，该类保存在 javax.mail.internet 包中，可以存储 MIME 类型和报头（在不同的 RFC 文档中均有定义）消息，并且将消息的报头限制成只能使用 US-ASCII 字符，尽管非 ASCII 字符可以被编码到某些报头字段中。

如果想对 MimeMessage 类进行操作，首先要实例化该类的一个对象。在实例化该类的对象时，需要指定一个 Session 对象，这可以通过将 Session 对象传递给 MimeMessage 的构造方法来实现。例如实例化 MimeMessage 类的对象 message 的代码如下：

```
MimeMessage msg = new MimeMessage(mailSession);
```

实例化 MimeMessage 类的对象 message 后，就可以通过该类的相关方法设置电子邮件信息的详细信息。MimeMessage 类中常用的方法包括以下几个：

（1）setText()方法

该方法用于指定纯文本信息的邮件内容。该方法只有一个参数，用于指定邮件内容。其语法格式如下：

```
setText(String content)
```

content：纯文本的邮件内容。

（2）setContent()方法

该方法用于设置电子邮件内容的基本机制，多数用于发送 HTML 等纯文本以外的信息。该方法包括两个参数，分别用于指定邮件内容和 MIME 类型。其语法格式如下：

```
setContent(Object content, String type)
```

☑ content：用于指定邮件内容。

☑ type：用于指定邮件内容类型。

例如，指定邮件内容为“简单快乐”，类型为普通的文本。代码如下：

```
message.setContent("简单快乐", "text/plain");
```

（3）setSubject()方法

该方法用于设置邮件的主题。该方法只有一个参数，用于指定主题内容。其语法格式如下：

```
setSubject(String subject)
```

subject：用于指定邮件的主题。

（4）saveChanges()方法

该方法能够保证报头域同会话内容保持一致。其语法格式如下：

```
msg.saveChanges();
```

（5）setFrom()方法

该方法用于设置发件人地址。该方法只有一个参数，用于指定发件人地址，该地址为 InternetAddress 类的一个对象。其语法格式如下：

```
msg.setFrom(new InternetAddress(from));
```

说明 创建 InternetAddress 类的对象的方法可参见下面的“Address 类”部分。

（6）setRecipients()方法

该方法用于设置收件人地址。该方法有两个参数，分别用于指定收件人类型和收件人地址。其语法格式如下：

```
setRecipients(RecipientType type, InternetAddress address);
```

☑ type：收件人类型。可以使用以下 3 个常量来区分收件人的类型。

- Message.RecipientType.TO　　//发送
- Message.RecipientType.CC　　//抄送
- Message.RecipientType.BCC　　//暗送

☑ address：收件人地址。可以为 InternetAddress 类的一个对象或多个对象组成的数组。

例如设置收件人的地址为 wgh8007@163.com 的代码如下：

```
address=InternetAddress.parse("wgh8007@163.com",false);
msg.setRecipients(Message.RecipientType.TO, toAddrs);
```

（7）setSentDate()方法

该方法用于设置发送邮件的时间。该方法只有一个参数，用于指定发送邮件的时间。其语法格式如下：

```
setSentDate(Date date);
```

date：用于指定发送邮件的时间。

（8）getContent()方法

该方法用于获取消息内容，该方法无参数。其语法格式如下：

```
getContent()
```

（9）writeTo()方法

该方法用于获取消息内容（包括报头信息），并将其内容写到一个输出流中。该方法只有一个参数，用于指定输出流。其语法格式如下：

```
writeTo(OutputStream os)
```

os：用于指定输出流。

3．Address 类

Address 类用于设置电子邮件的响应地址。Address 类是一个抽象类，要使用该抽象类可以使用其子类 InternetAddress，该类保存在 javax.mail.internet 包中，可以按照指定的内容设置电子邮件的地址。

如果想对 InternetAddress 类进行操作，首先要实例化该类的一个对象。在实例化该类的对象时，有以下两种方法：

（1）创建只带有电子邮件地址的地址，可以把电子邮件地址传递给 InternetAddress 类的构造方法。代码如下：

```
InternetAddress address = new InternetAddress("wgh717@sohu.com");
```

（2）创建带有电子邮件地址并显示其他标识信息的地址，可以将电子邮件地址和附加信息同时传递给 InternetAddress 类的构造方法。代码如下：

```
InternetAddress address = new InternetAddress("wgh717@sohu.com","Wang GuoHui");
```

说明 Java Mail API 没有提供检查电子邮件地址有效性的机制。如果需要，读者可以自己编写检查电子邮件地址是否有效的方法。

4．Authenticator 类

Authenticator 类通过用户名和密码来访问受保护的资源。Authenticator 类是一个抽象类，要使用该抽

象类首先需要创建一个 Authenticator 的子类，并重载 getPasswordAuthentication()方法。关键代码如下：

```
class WghAuthenticator extends Authenticator {
    public PasswordAuthentication getPasswordAuthentication() {
        String username = "wgh";                                    //邮箱登录账号
        String pwd = "111";                                         //登录密码
        return new PasswordAuthentication(username, pwd);
    }
}
```

然后再通过以下代码实例化新创建的 Authenticator 的子类，并将其与 Session 对象绑定。

```
Authenticator auth = new WghAuthenticator ();
Session session = Session.getDefaultInstance(props, auth);
```

5．Transport 类

Transport 类用于使用指定的协议（通常是 SMTP）发送电子邮件。Transport 类提供了以下两种发送电子邮件的方法。

（1）只调用其静态方法 send()，按照默认协议发送电子邮件。代码如下：

```
Transport.send(message);
```

（2）首先从指定协议的会话中获取一个特定的实例，然后传递用户名和密码，再发送信息，最后关闭连接。代码如下：

```
Transport transport =sess.getTransport("smtp");
transport.connect(servername,from,password);
transport.sendMessage(message,message.getAllRecipients());
transport.close();
```

在发送多个消息时，建议采用第二种方法，因为它将保持消息间活动服务器的连接，而使用第一种方法时，系统将为每一个方法的调用建立一条独立的连接。

注意　如果想要查看经过邮件服务器发送邮件的具体命令，可以用 session.setDebug(true)方法设置调试标志。

6．Store 类

Store 类定义了用于保存文件夹间层级关系的数据库，以及包含在文件夹之中的信息，该类也可以定义存取协议的类型，以便存取文件夹与信息。

在获取会话后，就可以使用用户名和密码或 Authenticator 类来连接 Store 类。与 Transport 类一样，首先要告诉 Store 类将使用什么协议。

☑　使用 POP3 协议连接 Stroe 类。代码如下：

```
Store store = session.getStore("pop3");
store.connect(host, username, password);
```

☑ 使用 IMAP 协议连接 Stroe 类。代码如下：

```
Store store = session.getStore("imap");
store.connect(host, username, password);
```

如果使用 POP3 协议，只能使用 INBOX 文件夹，但是使用 IMAP 协议，则可以使用其他的文件夹。

在使用 Store 类读取完邮件信息后，需要及时关闭连接。关闭 Store 类的连接可以使用以下代码：

```
store.close();
```

7．Folder 类

Folder 类定义获取（fetch）、备份（copy）、附加（append）及删除（delete）信息等的方法。

在连接 Store 类后，就可以打开并获取 Folder 类中的消息。打开并获取 Folder 类中的信息的代码如下：

```
Folder folder = store.getFolder("INBOX");
folder.open(Folder.READ_ONLY);
Message message[] = folder.getMessages();
```

在使用 Folder 类读取完邮件信息后，需要及时关闭对文件夹存储的连接。关闭 Folder 类的连接的语法格式如下：

```
folder.close(Boolean boolean);
```

boolean：用于指定是否通过清除已删除的消息来更新文件夹。

6.13 本 章 小 结

本章通过一个典型实用的企业快信应用软件，系统地向读者介绍了如何利用 Java 应用程序，实现群发手机短信和群发电子邮件的方法。在实现该企业快信应用软件时，还添加了几个方便用户使用的功能，例如利用名片夹管理名片信息，以及利用信息库管理信息模板，这样用户就不用反复地填写收信人列表以及编写重复的信息了。希望读者能掌握实现这些功能的方法，并做到灵活运用，以便实现更加人性化的企业快信程序。

第 7 章

欣想电子商城

（Swing+Hibernate+SQL Server 实现）

电子商务网站是信息时代企业生存的基础，同时也是企业对外展示信息、从事商务活动的窗口和平台。如何设计、建立一个经济、实用、安全、高效、稳定的网站是每个电子商务网站必须考虑的问题。本章将带领读者实现一个电子商城网站，使用该网站能够提高企业内部管理效率，充分利用企业内部资源，从整体上降低成本，加快对市场的响应速度，提高服务质量，进而全面提高企业的竞争力。

通过阅读本章，可以学习到：

- 如何进行项目的需求分析和可行性分析
- 如何进行系统设计
- 如何进行数据库分析和数据库建模
- 如何配置 Spring 的数据源
- 电子商城主要功能模块的实现方法
- 网站发布与运行
- 使用 MyEclipse 生成 Hibernate 实体类

7.1 开 发 背 景

欣想商城是一家以商品销售为主体的购物商城。公司为了扩大规模，增强企业的竞争力，决定向多元化发展，借助 Internet 在国内的快速发展，将部分资金投入网站建设，为顾客提供更加方便、快捷的电子购物方式，即电子商城。

笔者受该商城的委托，开发欣想电子商城系统，它必须具有使用方便、用户界面友好、运行速度快、系统稳定可靠、远程维护方便等特点。

7.2 系 统 分 析

7.2.1 需求分析

在线购物已经成为一种时尚，人们足不出户就可以购买所需商品，因其具有方便、安全、友好的交互等特性，顾客群体也逐渐庞大，尤其是网络时代中成长的年轻人。现在流行的电子商务有 B2B、B2C、C2C、G2C 等类型。欣想电子商城要采用的是 B2B 类型，它可以使顾客通过网络购物、浏览商品、查询订单、查看公告和销售排行等。通过对一些典型电子商城网站的考察、分析，并结合企业要求以及实际的市场调查，要求本系统具有以下功能：

- ☑ 美观、友好的操作界面，能保证系统的易用性。
- ☑ 规范、完善的基础信息设置。
- ☑ 商品分类详尽，可按不同类别查看商品信息。
- ☑ 按商品大类及商品名称进行模糊查询。
- ☑ 实现网上购物。
- ☑ 新品及特价商品展示。
- ☑ 商品销售排行。

7.2.2 可行性分析

企业在运营过程中，经常会受到以下条件的限制：

- ☑ 产品的宣传受到限制，采购商或顾客只能通过上门咨询、电话沟通等方式进行各种信息的获取，受一定的时间与物理空间的局限并且成本较高。
- ☑ 庞大的商业经济周转。
- ☑ 复杂的产品周转渠道。从看样品、谈价格到支付货款等一系列的产品周转渠道过于复杂，企业与顾客之间缺乏全面的沟通与快捷运营的平台。
- ☑ 商业企业中根据季节的变化，热销商品在销售高峰到来时货源紧张，企业需要实时了解商品的销售情况，保证热销商品的要货满足率。

因此，企业需要重新认识市场、消费者以及自身市场定位，正确认识电子商务技术在企业中的重要地位，以少量的时间和资金建立企业信息门户网站并架设一定范围的商务网络，以此来制定长远发

展战略，使企业与顾客间的经济活动变得更灵活、更主动。

7.3　系 统 设 计

7.3.1　系统目标

建设欣想电子商城的最终目的是发展业务和提高业绩。目前 Internet 网上商家不少，但由于缺乏相应的安全保障、管理机制和可维护性，造成重复建设和资源浪费。一个网上购物网站，尤其是数据流量比较大的网络管理系统，必须要满足使用方便、操作灵活等设计需求。本系统在设计时应该满足以下几个目标：

- ☑ 灵活的信息查询，界面设计要美观友好，方便、快捷、准确，数据存储安全可靠。
- ☑ 全面展示商城内所有商品，并可以展示最新商品。
- ☑ 实现网上购物。
- ☑ 商品销售排行，方便顾客了解本商城内的热销商品，同时辅助企业作出相应的策略调整。
- ☑ 查看商城内的公告信息。
- ☑ 系统最大限度地实现易维护性和易操作性。
- ☑ 系统运行稳定、安全可靠。

7.3.2　系统功能结构

电子商务系统是一个典型的 Java Web 应用程序，它由系统前台和后台管理两部分组成。

☑ 系统前台

该部分主要包括商品展台、商品购物、会员管理、商城公告及订单查询、商品查询等。

☑ 后台管理

该部分主要对商城内的一些基础数据进行有效管理，包括商品管理、会员管理、订单管理、公告管理等。

电子商务系统的前台功能结构如图 7.1 所示。

电子商务系统的后台功能结构如图 7.2 所示。

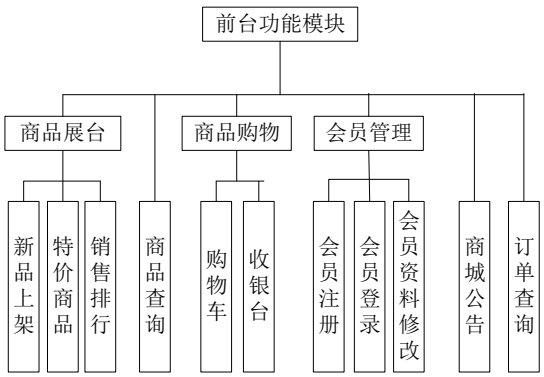


图 7.1　系统前台功能结构

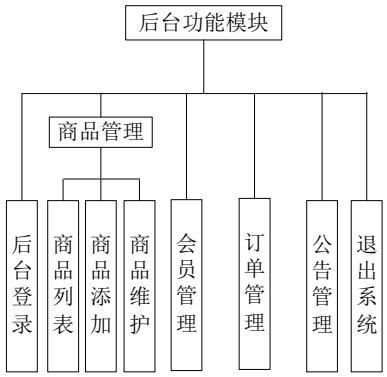


图 7.2　系统后台功能结构

7.3.3 购物流程图

在超市选购商品时，首先应该将商品放到购物车中，待挑选好所有商品后就可以到收银台去开收货单，根据收货单据付款。制作电子商务系统的原理与超市购物的原理是一样的，首先客户应该在网页中选购自己需要的商品并将商品放入购物车中，当然也可以改变购买商品的数量或清空购物车中的商品。选购好商品后就可以到收银台，在收银台填写并提交收货人信息。欣想电子商城的购物流程如图 7.3 所示。

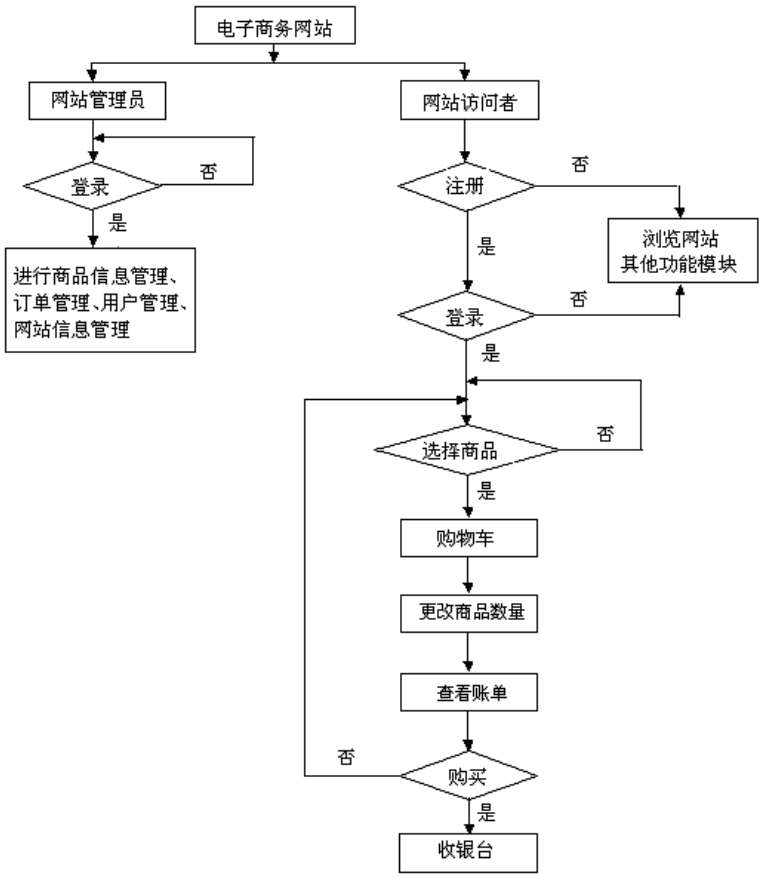


图 7.3　电子商城业务流程图

7.3.4 系统预览

欣想电子商城由多个程序页面组成，下面仅列出几个典型页面的预览，其他页面参见光盘中的源程序。

前台首页如图 7.4 所示，该页面用于实现商品信息展示、用户登录、公告信息、特价商品、商品信息查询等功能。前台新品页面如图 7.5 所示，该页面为顾客展示所有新上架的商品信息，其中包括商品名称、单价、简介等信息。

图 7.4　前台首页（光盘\…\jsp\template\indexTemplate.jsp）

图 7.5　前台新品页面（光盘\…\jsp\NewGoods.jsp）

后台订单管理的订单详细信息页面如图 7.6 所示，该页面用于查看订单的详细信息，当管理员核对该信息之后才到管理页面执行该订单。后台公告列表页面如图 7.7 所示，该页面用于实现公告添加、公告删除等功能。

图 7.6　后台订单详细信息页面
（光盘\…\manage\order_detail.jsp）

图 7.7　后台公告列表页面
（光盘\…\manage\placardmanage.jsp）

说明　由于路径太长，因此省略了部分路径，省略的路径是“TM\07\XinXiangShop\WebRoot\WEB-INF”。

7.3.5　构建开发环境

本系统的开发环境是 MyEclipse 9，具体的 IDE 下载和安装与本章内容无关。下面仅介绍系统开发中需要的各种框架和 JDBC 驱动类包在 Eclipse 构建路径中的设置。

1．添加 Spring 环境

本系统的主体框架采用 Spring 实现网站的 MVC 模式。为提供对 Spring 框架的支持，必须在 Eclipse 的构建路径中添加 Spring 框架的类库。具体添加步骤如下：

（1）在工程名称上右击，在弹出的快捷菜单中选择“构建路径”/“添加库”命令，在弹出的“添加库”对话框中选择 MyEclipse Libraries 列表项，单击“下一步”按钮。

（2）在弹出的“添加库”对话框中分别选中 Spring 1.2 Core Libraries、Spring 1.2 ORM/DAO/Hibernate3 Libraries 和 Spring 1.2 Web Libraries 复选框，如图 7.8 所示，然后单击“完成”按钮。

图 7.8　添加 MyEclipse 自带的 Spring 类库

注意　低版本的 MyEclipse 自带的 Spring 框架的类库对中文处理不是很好，读者可以参考下面的“3.添加 JDBC 环境”的步骤添加外部高版本的 Spring 类库（如 Spring 2.0）。

2．添加 Hibernate 环境

Hibernate 框架类库也是 MyEclipse 自带的，其添加方法与 Spring 相同，不过在“添加库”对话框中选择类库时，应该选中 Hibernate 3.1 Core Libraries 和 Hibernate 3.1 Advanced Support Libraries 复选框，如图 7.9 所示，然后单击“完成”按钮。

注意 低版本的 MyEclipse 自带的 Hibernate 框架的类库对中文处理不是很好，读者可以参考下面的“3.添加 JDBC 环境”的步骤添加外部高版本的 Hibernate 类库（如 Hibernate 3.2）。

3．添加 JDBC 环境

欣想电子商城采用了 SQL Server 2000 数据库系统，所以在项目中需要添加相应的 JDBC 驱动包。笔者的 JDBC 驱动类库存放在 D:\EclipseLib\JDBCforMSSQL 文件夹中。在此基础上添加 JDBC 驱动类库的步骤如下：

（1）在 Eclipse 中选择“窗口”/“首选项”命令，将弹出“首选项”对话框，然后依次展开左侧菜单树的 Java/“构建路径”/“用户库”节点项，或者直接在左上角的文本框中输入“用户库”来激活设置界面，如图 7.10 所示。

图 7.9　添加 MyEclipse 自带的 Hibernate 类库

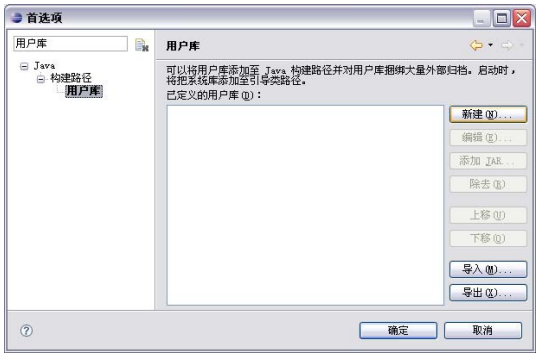


图 7.10　添加用户库

（2）单击图 7.10 所示对话框中的“新建”按钮，在弹出的“新建用户库”对话框中输入新建立的类库名称 MsSqlJDBC，如图 7.11 所示，单击“确定”按钮。

（3）在图 7.10 所示的对话框中选择新建立的 MsSqlJDBC 用户库，单击右侧的“添加 JAR”按钮，在弹出的“选择 JAR”对话框中，选择连接 SQL Server 2000 数据库的 3 个 JDBC 驱动包，然后关闭图 7.10 所示的对话框。

图 7.11　“新建用户库”对话框

（4）在工程名称上右击，在弹出的快捷菜单中选择“构建路径”/“添加库”命令，在弹出的“添加库”对话框中选择“用户库”列表项，单击“下一步”按钮。

（5）在弹出的“添加库”对话框中选择新建立的 MsSqlJDBC 类库，然后单击“完成”按钮。

4．添加 Struts 环境

欣想电子商城使用了 Tiles 布局框架，而这个框架是包含在 Struts 类库中的，所以需要为项目提供

对 Struts 框架的支持。实现步骤如下：

（1）在用户库中创建 Struts 类库，命名为 struts，然后添加 Struts 的 JAR 文件，笔者将其存放在“D:\EclipseLib\jakarta-struts-1.1”文件夹中。

（2）在项目名称上右击，在弹出的快捷菜单中选择“构建路径”/“添加库”命令，在弹出的“添加库”对话框中选择“用户库”列表项，单击“下一步”按钮。

（3）在弹出的“添加库”对话框中选择新建立的 struts 类库，然后单击“完成”按钮。

5．为 MyEclipse 配置 Tomcat 服务器

MyEclipse 提供了各种 Java Web 服务器的连接方式，经过设置后，可以直接在 Eclipse 中启动 Web 服务器，这样可以更加方便地控制服务器的启动和停止。另外，服务器的所有日志信息将输出到 Eclipse 的控制台中，将更加方便程序的调试。本系统的运行环境采用的是 Tomcat 5.5，这里以该服务器的连接方式为例，介绍具体的实现步骤。

（1）在 Eclipse 中选择“窗口”/“首选项”命令，将弹出“首选项”对话框，然后依次展开左侧菜单树中的 MyEclipse/Application Servers/Tomcat 5 节点项，或者直接在左上角的文本框中输入 Tomcat 来激活设置界面，如图 7.12 所示。

（2）在图 7.12 所示的对话框中单击 Tomcat Home Directory 文本框右侧的“浏览”按钮，在弹出的对话框中选择 Tomcat 服务器的安装位置。在笔者的机器中，Tomcat 被安装在 D:\Tomcat 5.5 文件夹中。

（3）设置了 Tomcat 的安装位置后，Tomcat Base Directory 和 Tomcat Temp Directory 文本框中的内容会自动设置。选中 Tomcat Server 栏中的 Enable 单选按钮，并单击“应用”和“确定”按钮。然后就可以通过工具栏中的按钮发布项目到服务器中，再通过工具栏上的按钮启动和停止 Tomcat 服务器。

7.3.6　文件夹组织结构

在进行网站开发前，还要规划网站的架构。也就是说，建立多个文件夹，对各个功能模块进行划分，实现统一管理。这样做的好处是易于开发、易于管理、易于维护。本系统的文件夹组织结构如图 7.13 所示。

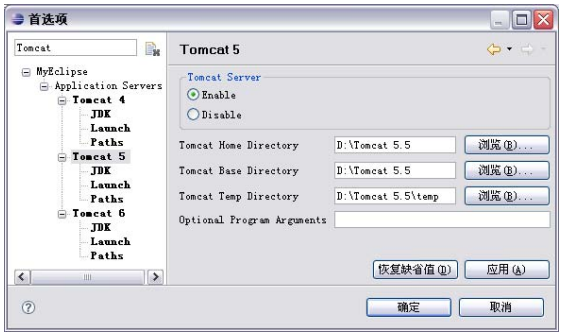


图 7.12　MyEclipse 的 Tomcat 服务器设置对话框

图 7.13　文件夹组织结构

7.4 数据库设计

数据库是欣想电子商城的信息基地，其中包含商品信息、会员信息、销售信息、订单信息、商品分类以及公告信息等，这些数据之间有各种直接或间接的对应关系。本节将介绍欣想电子商城的数据库分析与设计过程。

7.4.1 数据库分析

为防止数据访问量增加使系统资源不足而导致的系统崩溃，欣想电子商城的数据库采用了独立的 SQL Server 2000 数据服务器，将数据库单独放在一个服务器中。这样即使系统崩溃了，数据库服务器也不会受到影响；另外一个好处就是能够更快、更好地处理更多的数据。其数据库运行环境如下。

- ☑ 硬件平台
  - ➢ CPU：P4 3.2GHz。
  - ➢ 内存：4GB 以上。
  - ➢ 硬盘空间：200GB。
- ☑ 软件平台
  - ➢ 操作系统：Windows 2003。
  - ➢ 数据库：SQL Server 2000。

7.4.2 数据库概念设计

分析系统功能结构图，每个功能模块都需要操作一个或多个数据实体，如商品实体对象、订单实体对象和会员实体对象等。本节将介绍系统中比较重要的几个数据实体，最终这些数据实体对象将创建成对应的数据表结构。

1．商品实体对象

商品实体包括商品编号、名称、类型、单价、进货时间等属性。商品编号是识别不同商品实体的唯一标识，其数据类型是 bigint，并且是数据库自增的（它随数据库记录的增加而增加）。其余的属性都是商品通用的特性，例如商品名称、分类、单价、进货时间和商品简介等。另外还有商品的标识属性，例如是否特价、是否新品或者是否热卖商品等。商品信息的实体对象如图 7.14 所示。

2．订单实体对象

订单实体对象对应着顾客购买商品的订单信息，它包括订单编号、商品品种数量、顾客信息、订单执行状态、付款方式、送货方式、折扣、订购日期和备注信息等。其中顾客信息包括姓名、用户名、地址、邮编和电话。订单的实体对象如图 7.15 所示。

3．会员实体对象

会员实体对象拥有会员的基本属性，这些属性包括会员编号、用户名、真实姓名、密码、城市、

地址、E-mail、邮编、证件类型、证件编号、电话等。另外，会员根据消费金额的累计可以分为不同的等级，享受更加优惠的折扣，同时会员如果有任何丧失信誉或违背协议的行为，其用户将被冻结，所以必须提供会员等级、冻结状态和消费金额等标识属性。会员的实体对象如图 7.16 所示。

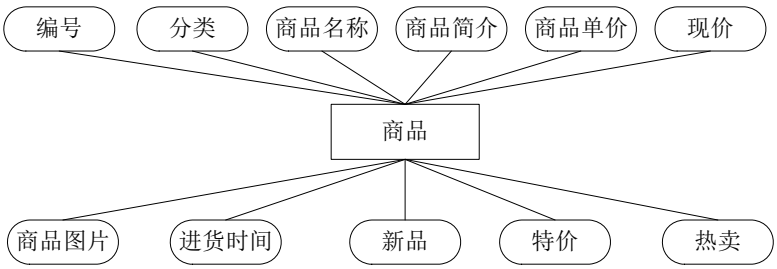


图 7.14　商品实体 E-R 图

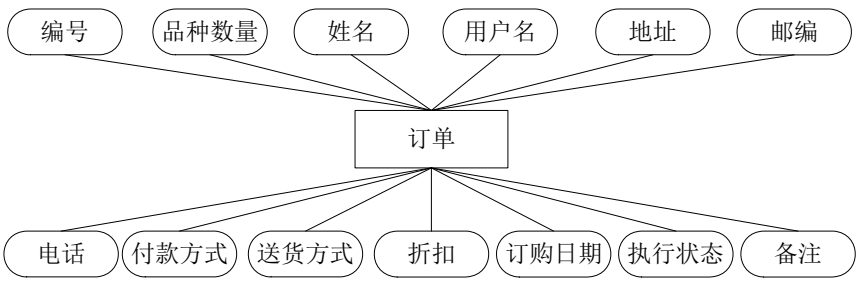


图 7.15　订单实体 E-R 图

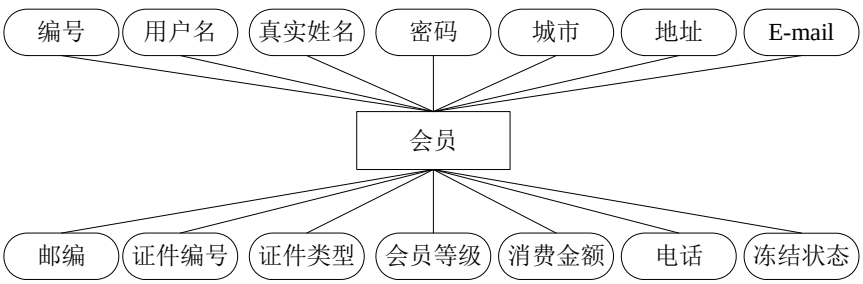


图 7.16　会员实体 E-R 图

7.4.3　PowerDesigner 数据库建模

数据库概念设计中已经分析了商品、订单和会员等主要的数据实体对象。另外，为方便数据查询，欣想电子商城的数据库中还创建了 4 个数据视图的实体，即商品信息视图（V_goods）、商品分类视图（V_Type）、会员视图（V_Member）和订单视图（V_order_detail）。这些实体对象是数据表结构的基本模型，最终的数据模型都要实施到数据库中，形成整体的数据结构。可以使用 PowerDesigner 工具完成这个数据库的建模。其模型结构如图 7.17 所示。

这个数据模型包含了欣想电子商城的所有数据实体和属性，它是对数据库的抽象模型，如果正确地构建了所有实体对象，就可以直接应用该模型构建数据库结构。

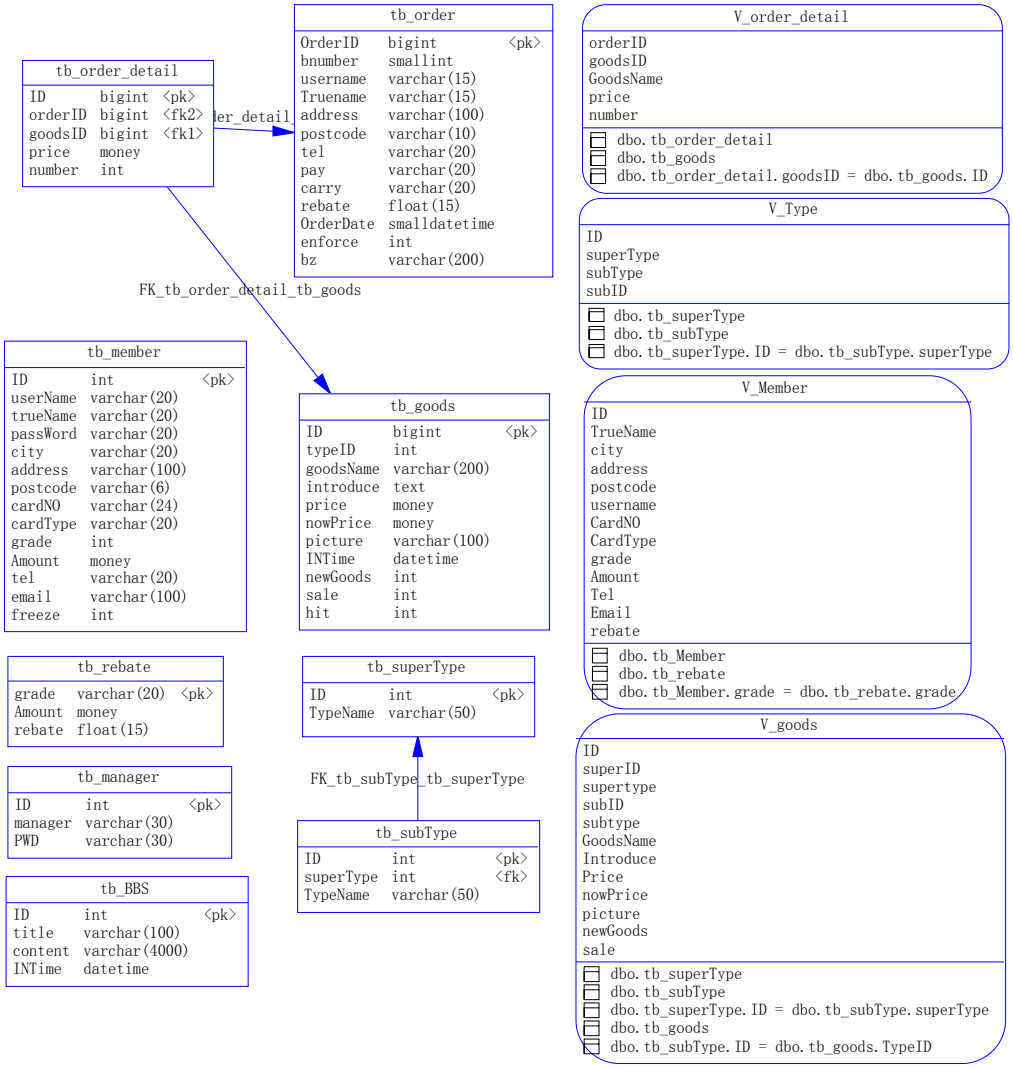


图 7.17　数据库模型图

7.4.4　数据库创建

通过对数据库的建模，可以使用数据库建模工具为欣想电子商城生成创建数据库的脚本，也可以使用该数据模型通过 ODBC 数据源直接在数据库中建立表结构。但是，在此之前，必须在数据库服务器中建立电子商城的数据库 db_shop；如果要使用 ODBC 直接创建数据结构，还必须创建该数据库的 ODBC 数据源连接。本实例的数据源名称为 db_shopODBC，具体实现步骤可参见第 1 章。

7.5　网站首页设计

在无数个相互竞争的网页中，特别是对电子商务网站来说，首页极为重要，它必须展现网站的特

色，并积极地加以表现，首页设计的好坏将直接影响顾客的购买欲望和情绪。在欣想电子商城的首页设计中，首先必须把商城推出的特价商品、最新商品、最新公告等商城的特色和动态信息展现给顾客，然后再提供查看销售排行、查看订单、购物车、商品分类查询等业务。

图 7.18　网站前台首页的运行效果

7.5.1　首页布局

本网站使用 Tiles 模板，布局网站的所有页面，每个单独的 Tiles 组件包含一个页面资源或者其他的 Tiles 组件。在网站的首页中，可以使用<tiles:insert/>标签导入不同的 Tile 组件来组成网页内容。分析欣想电子商城的首页效果图，它分为网站导航、搜索栏、版权信息、左侧分栏、内容分栏和右侧分栏共 6 个部分。欣想电子商城前台首页的运行效果如图 7.18 所示。

布局网站首页的步骤如下：

（1）网站的每个部分都使用不同的 Tiles 组件定义，这些 Tiles 组件的组合就构成了网站首页的模板。组合这些 Tile 组件由 indexTemplate.jsp 页面负责。关键代码如下：

例程 01　代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\jsp\template\indexTemplate.jsp

```
<table align="center" width="792" border="0" cellspacing="0"
    cellpadding="0">
    <tr>
        <td class="tableBorder">
❶           <tiles:insert attribute="navigation" />
            <table width="100%" height="500" border="0" cellpadding="0"
                cellspacing="0">
                <tr>
                    <td valign="top" width="185" align="right"
                        background="images/bg_3.jpg">
❷                       <tiles:insert attribute="left" /></td>
                    <td valign="top">
                        <table width="100%" cellpadding="0" cellspacing="0">
                            <tr>
```

```
                                        <td width="607" colspan="2" valign="top">
❸                                           <tiles:insert attribute="search" /></td>
                                    </tr>
                                    <tr>
                                        <td valign="top" width="69%">
❹                                           <tiles:insert attribute="content" /></td>
                                        <td valign="top" width="31%">
❺                                           <tiles:insert attribute="right" /></td>
                                    </tr>
                                </table>
                            </td>
                        </tr>
                    </table>
                    <table width="790" border="0" align="center" cellpadding="0"
                        cellspacing="0">
❻                       <tr><td><tiles:insert attribute="footer" /></td></tr>
                    </table>
                </td>
            </tr>
        </table>
```

代码贴士

❶ navigation：网站的导航 Tiles 组件，它包括网站的导航菜单和 LOGO 标志。

❷ left：网站左侧分栏的 Tiles 组件，它包括用户登录、商城公告和商品销售排行版块。

❸ search：网站搜索栏的 Tiles 组件，它还包括一个 Banner 广告栏。

❹ content：网站首页内容的 Tiles 组件，它由特价商品和新品上架版块组成。

❺ right：网站右侧分栏的 Tiles 组件，它包含商品分类和广告招商版块。

❻ footer：网站版权信息的 Tiles 组件，它包含网站的所有者、联系电话和 E-mail 等信息。

（2）创建 tiles-template.xml 文件，用于定义前台页面的 Tiles 布局组件，其中包括出错页面的布局模板、首页的布局模板和次级页面的布局模板。这些模板由标题信息、导航栏、左菜单、右菜单、搜索栏、版权信息以及页面的内容分栏 7 个 Tiles 组件组成。模板中定义了 placardTilesController、sellSortTilesController、SearchTileController 和 TypeListTilesController 共 4 个 Tiles 控制器，这 4 个控制器分别为公告页面、销售排行、搜索栏和商品分类提供数据。关键代码如下：

例程 02 代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\tiles-template.xml

```
<!--模板-->
<definition name="Error" path="/WEB-INF/jsp/template/error.jsp" />
<!--首页模板-->
<definition name=".indexLayout"
    path="/WEB-INF/jsp/template/indexTemplate.jsp">
    <put name="title" value="欣想电子商城" />                          <!--网站标题-->
    <put name="navigation" value="/WEB-INF/jsp/navigation.jsp" />     <!--导航栏-->
    <put name="left" value=".left" />                                 <!--左菜单-->
    <put name="search" value=".search" />                             <!--搜索栏-->
    <put name="content" value="/WEB-INF/jsp/productInfo.jsp" />       <!--内容分栏-->
    <put name="right" value=".right" />                               <!--右菜单-->
    <put name="footer" value="/WEB-INF/jsp/footer.jsp" />             <!--版权信息-->
```

```
</definition>
<!--次页模板-->
<definition name=".myLayout"
        path="/WEB-INF/jsp/template/pageTemplate.jsp">
        <put name="title" value="欣想电子商城" />                          <!--网站标题-->
        <put name="navigation" value="/WEB-INF/jsp/navigation.jsp" />      <!--导航栏-->
        <put name="left" value=".left" />                                  <!--左菜单-->
        <put name="search" value=".search" />                              <!--搜索栏-->
        <put name="content" value="/WEB-INF/jsp/productInfo.jsp" />        <!--内容分栏-->
        <put name="footer" value="/WEB-INF/jsp/footer.jsp" />              <!--版权信息-->
</definition>
<!-- 其他 Tiles 组件 -->
<definition name=".search" path="/WEB-INF/jsp/search.jsp"
        controllerClass="com.lzw.tiles.SearchTileController" />            <!--搜索栏控制器-->
<definition name=".placard" path="/WEB-INF/jsp/placard.jsp"
        controllerClass="com.lzw.tiles.placardTilesController" />          <!--公告栏控制器-->
<definition name=".sellSortLeft"
        path="/WEB-INF/jsp/sellSortLeft.jsp"
        controllerClass="com.lzw.tiles.sellSortTilesController" />         <!--销售排行控制器-->
<definition name=".left"
        page="/WEB-INF/jsp/template/leftTemplate.jsp">
        <put name="login" value="/WEB-INF/jsp/login.jsp" />
        <put name="placard" value=".placard" />
        <put name="sellSortLeft" value=".sellSortLeft" />
</definition>
<definition name=".typeList"
        controllerClass="com.lzw.tiles.TypeListTilesController"            <!--商品分类控制器-->
        path="/WEB-INF/jsp/productClass.jsp" />
<definition name=".right"
        path="/WEB-INF/jsp/template/rightTemplate.jsp">
        <put name="productClass" value=".typeList" />
        <put name="guanggao" value="/WEB-INF/jsp/guanggao.jsp" />
</definition>
```

（3）创建搜索栏的 Tiles 组件控制器，其名称为 SearchTileController。它从 Spring 容器中的 daoProxyFactory 代理中获取了 Dao 数据库操作类的实例对象，然后通过该对象获取页面需要的数据，并将数据放到 ComponentContext 上下文中。关键代码如下：

例程 03　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\tiles\SearchTileController.java

```
public class SearchTileController extends ComponentControllerSupport {
    protected void doPerform(ComponentContext componentContext,
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {
        Dao dao=(Dao)getApplicationContext().getBean("daoProxyFactory");//获取 Dao 数据库操作对象
        List list=dao.getSearchClassify();                              //获取搜索栏分类数据
        componentContext.putAttribute("searchClassify", list);          //将数据存入 Tiles 上下文中
    }
}
```

（4）创建公告栏的 Tiles 控制器，其名称为 placardTilesController。该类的实现方法和 SearchTileController

类相同，它们都继承了 Spring 的 ComponentControllerSupport 类，然后获取容器的 Dao 操作类，再通过该类获取页面数据并存放到 Tiles 的上下文中。关键代码如下：

例程 04 代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\tiles\placardTilesController.java

```
public class placardTilesController extends ComponentControllerSupport {
    protected void doPerform(ComponentContext componentContext,
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {
        Dao dao=(Dao)getApplicationContext().getBean("daoProxyFactory");//获取 Dao 数据库操作对象
        List list=dao.getPlacard();                                    //获取公告栏页面数据
        componentContext.putAttribute("placard", list);                //将数据存入 Tiles 上下文中
    }
}
```

（5）在 tiles-defs.xml 文件中定义首页的视图“index”，它也是 Tiles 组件，该组件继承首页模板的 Tiles 组件“.index”（这里以“.”作前缀来区分模板和 Tiles 视图），它不需要做任何修改，在模板中的 Tiles 组件中已经具备了主页的所有视图。关键代码如下：

例程 05 代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\tiles-defs.xml

```
<!-- 首页-->
<definition name="index" extends=".indexLayout" />
```

7.5.2 创建首页控制器

Spring 的控制器有很多种，例如表单控制器、命令控制器、向导控制器等。它们都用于处理客户的请求（即用户通过浏览器浏览并提交的页面请求），并根据客户的请求完成不同的业务逻辑（即使是简单的首页浏览也是一个请求），然后将数据封装到数据模型中并返回视图信息，Spring 的视图解析器会解析该视图信息，指定一个视图来显示控制器返回的数据模型。

首页显示的主要内容是展示商城的商品信息，其数据由 goodsController 控制器提供（该名称是 Spring 容器定义的，具体的实现类是 NewGoodsController），这个控制器就是首页控制器。它是 Spring 的 MultiActionController 控制器（即多动作控制器）的子类，除了为主页控制器提供数据模型外，它还负责处理其他商品视图的业务逻辑。

继承 Spring 的 MultiActionController 控制器创建 NewGoodsController 类，在该类中定义操作数据库的 dao 属性，它是 Dao 类的实例对象，由 Spring 在容器中实例化并注入到该类中。另外还需要定义 SaleGoodsLine 和 newGoodsLine 属性，它们分别用于定义特价商品和新进商品在视图中显示的行数。最后创建 goodsShow()方法处理首页请求的业务逻辑，该方法将返回 index 视图和相应的数据模型。关键代码如下：

例程 06 代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\NewGoodsController.java

```
private Dao dao;                                    //定义 dao 属性
private int saleGoodsLine = 1;                      //定义控制特价商品行数的属性
private int newGoodsLine = 4;                       //定义控制新商品行数的属性
```

```
public ModelAndView goodsShow(HttpServletRequest request,
        HttpServletResponse response) {
    List list1 = dao.getShowSaleGoodsList();                    //调用获取特价商品数据的方法
    List list2 = dao.getShowNewGoodsList();                     //调用获取新商品数据的方法
    Map model = new HashMap();                                  //创建数据模型
    model.put("saleGoods", list1);                              //添加特价商品数据到数据模型
    model.put("newGoods", list2);                               //添加新商品数据到数据模型
    model.put("saleGoodsLine", saleGoodsLine);                  //添加特价商品显示行数到数据模型
    model.put("newGoodsLine", newGoodsLine);                    //添加新商品显示行数到数据模型
    return new ModelAndView("index", model);                    //返回数据模型和视图
}
```

7.5.3　配置控制器

首页控制器已经实现了简单的获取数据的业务逻辑，但是它现在无法处理首页的请求。在 Spring 应用中，控制器必须在配置文件中定义名称并为其注入依赖的属性，然后定义处理器映射，才能处理指定的客户请求。配置该控制器的步骤如下：

（1）创建 controller-config.xml 文件，它是项目的前台控制器配置文件。由于 NewGoodsController 类是 Spring 的多动作控制器的实现类，多动作控制器必须指定方法名称解析器来指定控制器中的每个方法所处理的客户请求，所以必须创建该控制器的方法名称解析器。该解析器的名称可以任意，建议使用有意义的名称，例如本系统采用的 goodsMethodResolver。另外在这个配置文件中也同时定义了 NewGoodsController 控制器，并将方法名称解析器注入到它的 methodNameResolver 属性中。配置文件的关键代码如下：

例程 07　代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\controller-config.xml

```
<!--  商品控制器-->
❶   <bean id="goodsMethodResolver"                              <!--方法名称解析器-->
    class="org.springframework.web.servlet.mvc.multiaction.PropertiesMethodNameResolver">
    <property name="mappings">
        <props>
            <prop key="/index.lzw">goodsShow</prop>             <!--映射处理首页请求的方法-->
            <prop key="/goodsNewGoods.lzw">newGoods</prop>
            <prop key="/goodsSale.lzw">saleGoods</prop>
        </props>
    </property>
</bean>
❷   <bean id="goodsController" class="com.lzw.NewGoodsController">   <!--首页控制器-->
    <property name="methodNameResolver">
        <ref local="goodsMethodResolver" />                     <!--首页控制器的方法名称解析器属性-->
    </property>
❸   <property name="dao">
❹       <ref bean="daoProxyFactory" />
    </property>
    <property name="saleGoodsLine">
❺       <value>2</value>
```

```
        </property>
        <property name="newGoodsLine">
            <value>3</value>
        </property>
</bean>
```

代码贴士

❶ goodsMethodResolver：使用<bean>标签定义多动作控制器的方法名称解析器。

❷ goodsController：使用<bean>标签定义多动作控制器的 JavaBean 实例。

❸ <property >：该标签用于指定 JavaBean 的属性。

❹ <ref>：该标签用于引用其他 JavaBean 的实例。

❺ <value>：该标签用于定义 JavaBean 的属性值。

（2）创建 view-config.xml 文件，它是本系统的处理器映射配置文件，其中包括了前台和后台所有控制器的处理器映射信息，首页控制器必须在此处定义 URL 映射信息，才能被客户访问，这个处理器映射的定义把客户请求的路径信息绑定到指定的控制器中。映射首页控制器的关键代码如下：

例程 08 代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\view-config.xml

```
<bean id="urlHandlerMapping"
      class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
      <property name="mappings">
            <props>
                  <!--前台-->
                  <prop key="/index.lzw">goodsController</prop>
                  …//省略其他控制器映射
            </props>
      </property>
</bean>
```

（3）在 view-config.xml 文件中对 NewGoodsController 控制器定义了处理器映射，当访问“/index.lzw”URL 路径时，将显示首页信息。但是 JSP 网站的默认首页是 index.jsp 视图文件，要使默认路径能够浏览首页信息，必须定义 index.jsp 文件，在文件中将请求转发到 NewGoodsController 控制器中。关键代码如下：

例程 09 代码位置：光盘\TM\07\XinXiangShop\WebRoot\index.jsp

```
<%@ page contentType="text/html; charset=gb2312" language="java"%>
<jsp:forward page="/index.lzw"/>
```

7.6 公共模块设计

在本系统的项目空间中，有部分模块是公用的，或者是多个模块甚至整个系统的配置信息，它们被多个模块重复调用完成指定的业务逻辑，本节将这些公共的模块提出来进行单独介绍。

7.6.1　编写 Dao 公共类

Dao 类主要负责有关数据库的操作，该类继承了 Spring 的 HibernateDaoSupport 类，通过它的 getHibernateTemplate()方法获取 Hibernate 的模板类操作数据库。本节将介绍 Dao 类中关键的数据库操作方法。Dao 类的定义代码如下：

例程 10　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\dao\Dao.java

```
package com.lzw.dao;
import java.util.*;
import org.springframework.dao.DataAccessException;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import com.lzw.form.ModifyMemberForm;
import com.lzw.model.*;
public class Dao extends HibernateDaoSupport {
    …//这里定义各种操作数据库的方法
}
```

1．getGoods(Long id)方法

该方法用于获取指定 ID 编号的商品实体对象，它调用 Spring 的 HibernateTemplate 模板类的 get()方法获取指定 ID 的实体类对象。关键代码如下：

例程 11　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\dao\Dao.java

```
//获取单个商品信息
public TbGoods getGoods(Long id) {
    return (TbGoods) getHibernateTemplate().get(TbGoods.class, id);          //获取商品的实体类对象
}
```

2．insertObject(Object obj)方法

该方法主要负责将实体对象添加到数据库中，实体对象会保存到它所对应的数据表中。调用 HibernateTemplate 模板类的 save()方法可以直接保存指定的实体对象。关键代码如下：

例程 12　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\dao\Dao.java

```
public boolean insertObject(Object obj) {
    try {
        getHibernateTemplate().save(obj);                                    //调用 save()方法
        return true;                                                         //保存成功则返回 true
    } catch (DataAccessException e) {
        e.printStackTrace();
        return false;                                                        //如果抛出异常则返回 false
    }
}
```

3．getTypeList()方法

该方法用于获取所有商品类别。它调用 HibernateTemplate 模板类的 find()方法执行 HQL 语句查询，查询结果将被添加到 List 集合类中并返回给调用者，但是 getTypeList()方法将查询结果以 Map 集合类

型返回。关键代码如下：

例程 13 代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\dao\Dao.java

```
public Map getTypeList() {                                          //获取商品类别列表
    Map model = new HashMap();
    List types = getHibernateTemplate()                             //获取大分类列表
            .find("select id.id,id.superType from VType group by id.id,id.superType");
    Iterator iterator = types.iterator();
    while (iterator.hasNext()) {
        Object[] type = (Object[]) iterator.next();
        List subList = getHibernateTemplate().find(                 //迭代查询每个大分类的子分类列表
            "from VType where id.id='" + type[0] + "'");
        model.put(type[1], subList);                                //将结果添加到 Map 集合中
    }
    return model;
}
```

4．getUser(String username, String pwd)方法

该方法可以通过指定的用户名和密码从数据库中获取相应的会员实体对象。为防止 SQL 注入，该方法将以用户名为依据，获取实体对象，然后再将该实体对象的密码属性与 pwd 参数进行比较，如果相同则返回该实体对象。关键代码如下：

例程 14 代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\dao\Dao.java

```
public TbMember getUser(String username, String pwd) {              //获取会员对象
    List list = null;
    TbMember user = null;
    list = getHibernateTemplate().find(                             //获取指定名称的会员实体对象
            "from TbMember where userName='" + username + "' and freeze=0");
    if (list.size() > 0) {
        user = (TbMember) list.get(0);
        if (pwd != null && !user.getPassWord().equals(pwd)) {       //比较会员实体对象的密码
            user = null;
        }
    }
    return user;                                                    //返回会员实体对象
}
```

5．getGrade(double amount)方法

该方法可以根据会员的消费总金额从折扣表中获取对应的会员级别，该级别以整数区分大小。关键代码如下：

例程 15 代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\dao\Dao.java

```
public int getGrade(double amount) {                                //获得 TbRebate 表数据等级为 ID 的记录
    int grade = 0;
    List list = getHibernateTemplate().find(                        //获取消费总金额对应的会员等级
            "select grade from TbRebate where amount<" + amount);
```

```
    if (list.size() > 0) {
        grade = Integer.valueOf(list.get(0) + "");
    }
    return grade;                                          //返回会员等级
}
```

6．getUserRebate(String username)方法

该方法根据指定的会员名称获取该会员的购物折扣信息，获取的折扣信息以字符串形式保存，本方法必须将它转换成浮点数字类型。关键代码如下：

例程 16　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\dao\Dao.java

```
public float getUserRebate(String username) {                  //获取会员折扣
    float rebate = 1;
    List list = getHibernateTemplate().find(                   //获取折扣信息
            "select id.rebate from VMember where username='" + username
                    + "'");
    if (list.size() > 0) {
        rebate = Float.valueOf(list.get(0) + "").floatValue();  //将折扣信息转换成浮点型
    }
    return rebate;
}
```

7.6.2　配置数据库连接和事务管理器

本系统的 dao-config.xml 文件用于设置 Spring 的数据库连接、配置 Hibernate 环境和事务代理工厂。其中事务代理工厂 daoProxyFactory 在 Dao 数据库操作类的基础上添加了事务处理的能力，本系统 Dao 类的实例对象就是使用该事务代理工厂生成的。关键代码如下：

例程 17　代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\dao-config.xml

```
<beans>
❶   <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.microsoft.jdbc.sqlserver.SQLServerDriver" />
        <property name="url" value="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_shop" />
        <property name="username" value="sa"/>
    </bean>
❷   <bean id="sessionFactory"
        class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
        <property name="dataSource"><ref local="dataSource" /></property>
        <property name="mappingDirectoryLocations">
            <list><value>classpath:com/lzw/model</value></list>
        </property>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">org.hibernate.dialect.SQLServerDialect</prop>
                <prop key="hibernate.show_sql">false</prop>
            </props>
```

```
        </property>
    </bean>
❸   <bean id="dao" class="com.lzw.dao.Dao">
        <property name="sessionFactory"><ref local="sessionFactory" /></property>
    </bean>
❹   <bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactionManager">
        <property name="sessionFactory"><ref local="sessionFactory" /></property>
    </bean>
❺   <bean id="daoProxyFactory"
        class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
        <property name="proxyTargetClass" value="true" />
        <property name="target"><ref local="dao" /></property>
        <property name="transactionManager"><ref local="transactionManager" /></property>
        <property name="transactionAttributes">
            <props>
                <prop key="insert*">PROPAGATION_REQUIRED</prop>
                <prop key="get*">PROPAGATION_REQUIRED,readOnly</prop>
            </props>
        </property>
    </bean>
</beans>
```

代码贴士

❶ dataSource：配置连接数据库的数据源。

❷ sessionFactory：配置 Hibernate 的 sessionFactory。

❸ dao：配置 Dao 数据库操作类，该类继承了 Spring 的 HibernateDaoSupport 类，需要为它注入 sessionFactory 属性。

❹ transactionManager：配置 Spring 的 Hibernate 事务管理器，它同样需要注入 sessionFactory 属性。

❺ daoProxyFactory：配置 Spring 的事务代理工厂，这个代理工厂所产生的 Dao 类的对象在原有 Dao 类的基础上添加了处理事务的能力。

7.6.3 配置 Spring 控制器的请求映射

实例中的 view-config.xml 文件定义了 Spring 框架的视图解析器、Tiles 配置器、处理器映射和异常解析器。该文件是 Spring 的配置文件，使用和 dao-config.xml 文件相同的创建方法。下面分别是 view-config.xml 文件对处理器映射、Tiles 配置器和 Spring 视图解析器的相关配置。

（1）处理器映射的关键代码如下：

例程 18 代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\view-config.xml

```
<beans>
    <bean id="urlHandlerMapping"
        class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
        <property name="mappings">
            <props>
                <!--前台-->
                <prop key="/index.lzw">goodsController</prop>          <!--首页控制器映射-->
                <prop key="/goods*.lzw">goodsController</prop>         <!--商品控制器映射-->
```

```
                <prop key="/register.lzw">registerController</prop>          <!--会员注册控制器映射-->
                <prop key="/login.lzw">loginController</prop>                <!--登录控制器映射-->
                <prop key="/modifyMember.lzw">modifyMemberController</prop><!--会员修改控制器映射-->
                <prop key="/order.lzw">orderController</prop>                <!--订单控制器映射-->
                <prop key="/sellSort.lzw">sellSortController</prop>          <!--销售排行控制器映射-->
                <prop key="/placardDetails.lzw">placardDetailsController</prop><!--公告控制器映射-->
                <prop key="/goodsDetails.lzw">goodsDetailsController</prop> <!--商品信息控制器映射-->
                <prop key="/type.lzw">typeController</prop>                  <!--商品分类列表控制器映射-->
                <prop key="/cart*.lzw">cartController</prop>                 <!--购物车控制器映射-->
                <prop key="/cartSee.lzw">cartSeeController</prop>            <!--购物车查看控制器映射-->
                <prop key="/cartCheckout.lzw">cartCheckoutController</prop><!--收银结账控制器映射-->
                <prop key="/orderDetails.lzw">orderDetailsController</prop>    <!--详细订单控制器映射-->
                <prop key="/searchController.lzw">searchController</prop>      <!--商品搜索控制器映射-->
                <!--后台-->
                <prop key="/loginM.lzw">loginMController</prop>              <!--登录控制器映射-->
                <prop key="/indexM.lzw">indexMController</prop>              <!--后台首页控制器映射-->
                <prop key="/superType.lzw">superTypeController</prop><!--商品大分类控制器映射-->
                <prop key="/superAdd.lzw">superAddController</prop>    <!--添加大分类控制器映射-->
                <prop key="/subType.lzw">subTypeController</prop>        <!--商品小分类控制器映射-->
                <prop key="/subTypeAdd.lzw">subTypeAddController</prop><!--添加小分类控制器映射-->
                <prop key="/goodsAdd.lzw">goodsAddController</prop>  <!--商品添加控制器映射-->
                <prop  key="/goodsDetailM.lzw">goodsDetailsMController</prop><!--后台详细商品控制器
映射-->
                <prop key="/goodsModify.lzw">goodsModifyController</prop>  <!--商品修改控制器映射-->
                <prop key="/goodsDel.lzw">goodsDelController</prop>    <!--商品删除控制器映射-->
                <prop key="/member*.lzw">memManagerController</prop><!--后台会员管理控制器映射-->
                <prop key="/orderM*.lzw">orderManagerController</prop><!--后台订单管理控制器映射-->
                <prop key="/placardManage.lzw">placardManageController</prop><!--公告管理控制器映射-->
                <prop key="/placardAdd.lzw">placardAddController</prop><!--添加公告控制器映射-->
            </props>
        </property>
    </bean>
```

（2）设置 Tiles 配置器的关键代码如下：

例程 19　代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\view-config.xml

```
<bean id="tilesConfiger"
    class="org.springframework.web.servlet.view.tiles.TilesConfigurer">
    <property name="definitions">
        <list>
            <value>WEB-INF/config/tiles-template.xml</value>                  <!--模板定义-->
            <value>WEB-INF/config/tiles-defs.xml</value>                      <!--视图定义-->
            <value>WEB-INF/config/tiles-managerDefs.xml</value>               <!--后台视图定义-->
            <value>WEB-INF/config/tiles-manageTemplate.xml</value>            <!--后台模板定义-->
        </list>
    </property>
</bean>
```

（3）配置 Spring 视图解析器的关键代码如下：

例程 20 代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\view-config.xml

```
    <bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass">
            <value>
                org.springframework.web.servlet.view.tiles.TilesJstlView  <!--使用 Tiles 和 JSTL 视图-->
            </value>
        </property>
    </bean>
</beans>
```

7.7 会员管理模块设计

7.7.1 会员管理模块概述

欣想电子商城前台中的会员管理主要包括会员注册、会员资料修改和会员登录 3 个功能。本节将介绍这 3 个功能对本系统的意义和实现的业务逻辑。

1．会员注册

在欣想电子商城网站中，顾客只有通过注册成为本系统的会员，才能进行商品购买的业务流程，否则只能做一个看客。会员注册功能的页面效果如图 7.19 所示。

图 7.19 会员注册页面

2．会员资料修改

该功能主要负责修改会员的注册资料，例如更改会员的密码、住址和联系方式等信息。会员资料修改页面效果和实现过程与会员注册功能基本相同，在此不再重复介绍。

3．会员登录

会员登录功能是系统识别不同会员的一种手段，系统必须能够清晰地辨认不同的会员，才能提供针对该会员的折扣和其他优惠信息，最主要的是系统需要记录会员的购物信息，并为会员的购物行为生成订单。

会员登录可以使系统识别不同等级的会员，并分别为这些会员提供不同的折扣和其他优惠活动。会员登录后，商品信息后会显示“购买”按钮，可以单击该按钮进行购物。另外，还可以使用系统提供的购物车、订单查看等其他服务。会员登录页面如图 7.20 所示。

图 7.20　会员登录页面

7.7.2　会员管理模块技术分析

会员管理模块的会员注册、会员资料修改和会员登录分别使用了 RegisterController、LoginController 和 ModifyMemberController 控制器处理相应的业务逻辑。

虽然会员管理模块使用了多个控制器，但是它们都属于 Spring 表单控制器，实现该控制器必须继承 Spring 的 SimpleFormController 类，它会自动将页面所提交的表单内容绑定到一个 JavaBean 中，这个 JavaBean 是对表单数据的封装，类似于 Struts 的 ActionForm 类。

Spring 的表单控制器能够分别实现表单页面的显示和表单的业务处理，当它接收到页面 GET 类型的请求时，将显示控制器的 formView 属性所指定的表单页面；在接收到页面 POST 类型的请求时，它会接收表单内容并实现相应的业务逻辑，然后返回 successView 属性指定的视图，但是会员注册功能并没有使用这一特性，因为它必须返回注册结果。

另外，表单控制器还可以设置一个表单验证器来验证用户输入的表单数据是否符合要求。

7.7.3　会员注册的实现过程

会员注册使用的数据表：tb_member

实现会员的注册功能需要创建会员注册页面、编写控制器、编写验证器、配置控制器和请求映射等步骤。其中控制器的请求映射已经在 7.6 节公共模块设计中介绍过，会员注册控制器被映射为处理“/register.lzw”请求，本节将详细介绍其他步骤的实现方法。

（1）创建会员注册页面，在该页面中创建一个表单，表单的 Action 属性（即请求路径）设置为 register.lzw，表单中包含用户名、真实姓名、所在城市、联系地址等会员信息的字段，这些字段都使用了 Spring 的<spring:bind>标签和表单类的指定属性进行了绑定。由于表单页面的字段和程序代码较多，本章将以表单的用户名字段为例，介绍 Spring 表单页面的定义，至于表单中其他字段的定义将提取到表 7.1 中（表中的“—”符号代表默认设置）。关键代码如下：

例程 21　代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\jsp\register.jsp

```
<form action="register.lzw" method="post" name="myform">
  <table width="100%"   border="0" cellspacing="-2" cellpadding="-2">
    <tr>
      <td width="18%" height="30" align="center">用 户 名：</td>
      <td width="82%" class="word_grey">
❶   <spring:bind path="command.username">
❷       <input name="${status.expression }" type="text" maxlength="20"
❸                   value="${status.value }">
```

```
❹          <span class="word_orange">* ${status.errorMessage }</span></spring:bind>
      </tr>
      …//省略其他表单字段的定义
    </table>
</form>
```

代码贴士

❶ command.username：表单类的属性，它将与表单字段进行绑定，该字段名称将随字段而改变，例如联系电话字段将绑定表单类的 command.tel 属性。表 7.1 中的字段绑定是不同表单字段所绑定的表单类的属性。

❷ ${status.expression }：这是 EL 表达式，它是绑定的表单类的属性名称，在页面的表单中使用该名称作为表单字段的名称。

❸ ${status.value }：该表达式的结果是绑定的表单类的属性值，在页面的表单中使用该属性值作为表单字段的值。

❹ ${status.errorMessage }：该表达式的结果是表单控制器绑定的验证器的错误信息，当用户输入的值不符合验证器的输入格式时，验证器将返回该错误信息。

表 7.1　表单字段定义

字段描述	字段绑定	字段类型	字段长度	字段描述	字段绑定	字段类型	字段长度
用户名	command.username	text	20	联系地址	command.address	text	50
真实姓名	command.truename	text	10	邮政编码	command.postcode	text	20
密码	command.pwd	text	20	证件号码	command.cardno	text	20
确认密码	command.pwd1	text	20	联系电话	command.tel	text	—
所在城市	command.city	text	—	E-mail	command.email	text	50

说明 表中的“—”符号代表默认设置。

（2）创建 RegisterController 类，该类是处理会员注册业务的控制器，它继承了 Spring 的 SimpleFormController 类，并重写了该类的部分方法来实现业务处理。首先，在 RegisterController 类的构造方法中调用 setCommandClass()方法设置表单类，这个表单类将用来封装表单中的字段信息。然后，在 onSubmit()方法中处理页面请求的业务逻辑，并返回数据模型和视图对象。最后，在 onBindAndValidate()方法中验证数据库中是否存在用户要注册的用户名，至于其他详细注册字段的验证由验证器去实现。RegisterController 类的关键代码如下：

例程 22　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\ModifyMemberController.java

```
public class RegisterController extends SimpleFormController {          //继承 Spring 的表单控制器
    private Dao dao;                                                     //定义 Dao 属性
    public RegisterController() {
        setCommandClass(ModifyMemberForm.class);                         //设置表单类
    }
    protected ModelAndView onSubmit(Object command, BindException errors)
            throws Exception {
        ModifyMemberForm form = (ModifyMemberForm) command;
        TbMember user = new TbMember();                                  //创建会员实体对象
        user.setAddress(form.getAddress());                              //使用表单数据初始化会员实体对象
        user.setCardNo(form.getCardno());
```

```
            user.setCardType(form.getCardtype());
            user.setCity(form.getCity());
            user.setEmail(form.getEmail());
            user.setPassWord(form.getPwd());
            user.setPostcode(form.getPostcode());
            user.setTel(form.getTel());
            user.setTrueName(form.getTruename());
            user.setUserName(form.getUsername());
            user.setFreeze(0);                                          //设置会员的默认冻结状态为解冻
            dao.insertObject(user);
            return new ModelAndView("register", "info", "注册成功。");      //返回注册成功信息和视图对象
        }
        protected ModelAndView handleRequestInternal(HttpServletRequest request,
                HttpServletResponse response) throws Exception {
            request.setCharacterEncoding("GBK");                        //设置编码格式
            return super.handleRequestInternal(request, response);
        }
        protected void onBindAndValidate(HttpServletRequest request,
                Object command, BindException errors) throws Exception {
            ModifyMemberForm form = (ModifyMemberForm) command;
            TbMember user = dao.getUser(form.getUsername(), null);      //如果数据库中存在指定的用户名
            if (user != null) {
                errors.rejectValue("username", null, "用户已存在。");      //返回错误信息
            }
        }
}
```

（3）创建 ModifyMemberValidate 类，该类实现了 Validator 接口成为 Spring 的验证器。处理会员注册请求的控制器在 onBindAndValidate()方法中做了简单的用户验证，ModifyMemberValidate 类主要负责更细化的验证工作，例如关键信息的空信息验证、密码验证和输入格式验证等。

这个验证器必须实现 Validator 接口中定义的 supports()和 validate()方法，它们分别用于判断验证器是否支持指定表单类的验证和验证表单类的属性，其中的验证错误信息将显示在注册页面相应的字段上。验证器的关键代码如下：

例程 23　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\ModifyMemberValidate.java

```
public class ModifyMemberValidate implements Validator {
    private Dao dao;
    public boolean supports(Class) {
        return clazz.equals(ModifyMemberForm.class);        //判断是否支持验证 clazz 参数指定的表单类
    }
    public void validate(Object target, Errors errors) {
        ModifyMemberForm form = (ModifyMemberForm) target;
        if (form.getUsername().trim().equals("")) {                 //判断是否输入了用户名信息
            errors.rejectValue("username", null, "请输入用户名");
        }
        if (form.getTruename().trim().equals("")) {                 //判断是否输入了真实姓名
            errors.rejectValue("truename", null, "请输入真实姓名");
        }
```

```
            if (form.getPwd().equals("")) {                          //判断是否输入了密码
                errors.rejectValue("pwd", null, "请输入密码");
            }
            if (form.getPwd1().equals("")) {                         //判断是否输入了确认密码
                errors.rejectValue("pwd1", null, "请输入密码");
            }
            //用户会员资料修改功能的验证，验证是否输入原密码
            if (form.getOldpwd() != null && form.getOldpwd().trim().equals("")) {
                errors.rejectValue("oldpwd", null, "请输入原密码");
            }
            if (form.getCardno().trim().equals("")) {                //判断是否输入了证件号码
                errors.rejectValue("cardno", null, "请输入证件号码");
            } else {
                String str = "1234567890";
                byte[] bytes = form.getCardno().trim().getBytes();
                for (byte ch : bytes) {                              //判断输入的证件号码是否是数字格式
                    if (str.indexOf((char) ch) < 0) {
                        errors.rejectValue("cardno", null, "以数字格式输入证件号码");
                        break;
                    }
                }
            }
            if (form.getEmail().trim().equals("")) {                 //判断是否输入了 E-mail 信息
                errors.rejectValue("email", null, "请输入 Email");
            } else if (form.getEmail().indexOf("@") < 3
                    || form.getEmail().indexOf(".") == -1) {         //判断是否输入了正确的 E-mail 格式
                errors.rejectValue("email", null, "请输入正确的 E-mail 格式");
            }
            if (!form.getPwd().equals(form.getPwd1())) {             //判断两次输入的密码是否一致
                errors.rejectValue("pwd", "ddd", "两次密码不一致，请重新输入。");
                errors.rejectValue("pwd1", "ddd", "两次密码不一致，请重新输入。");
            }
        }
    }
```

（4）会员注册控制器和表单验证器必须在 controller-config.xml 文件（即 Spring 的配置文件）中配置定义信息，才能被 Spring 框架加载并处理会员注册请求。关键代码如下：

例程 24 代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\controller-config.xml

```
<!--  会员注册-->
<bean id="registerController" class="com.lzw.RegisterController">
    <property name="dao">
        <ref bean="daoProxyFactory" />
    </property>
    <property name="formView">
        <value>register</value>
    </property>
    <property name="validator">                                      <!--设置表单验证器-->
        <bean class="com.lzw.ModifyMemberValidate" />
```

```
    </property>
</bean>
```

7.7.4　会员登录的实现过程

会员登录使用的数据表：tb_member

实现会员登录功能需要经过创建登录页面、编写登录控制器、配置控制器和请求映射等步骤。其中 7.6 节公共模块设计中已经介绍过，会员登录控制器被映射为处理“/login.lzw”请求，本节将详细介绍其他步骤的实现方法。

（1）创建会员登录页面，在该页面中创建一个表单，表单的 Action 属性（即请求路径）设置为 login.lzw，表单中包含“会员名”文本框、“密码”文本框和会员注册的超链接。关键代码如下：

例程 25　代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\jsp\login.jsp

```
<form name="form1" method="post" action="login.lzw">
<table width="100%" border="0" cellpadding="0" cellspacing="0">
<tr>
    <td width="10%"> </td>
    <td width="90%" valign="top" background="Images/index_09.gif">
    <c:if test="${param.loger=='error'}">                    <!--如果返回 error 信息，提示错误-->
        <script type="text/javascript"><!--alert("用户名和密码错误，或者该用户已被冻结。");//--></script>
    </c:if>
    <c:if test="${sessionScope.username==null }">            <!--如果未登录就显示登录页面-->
    <table width="100%" border="0" cellpadding="0" cellspacing="0">
        <tr><td width="94%" height="24">会员名
                <input name="username" type="text" class="txt_grey"
                    id="username" size="16"></td>
        </tr>
        <tr><td height="24">密  码
                <input name="password" type="password" class="txt_grey" id="PWD"
                    size="16"></td>
        </tr>
        <tr><td height="31"> <a href="register.lzw">注册</a>
                <input name="Submit2" type="submit" class="btn_grey"
                    value="登录">
                <input name="Submit3" type="reset" class="btn_grey" value="重置">
                </td></tr>
    </table>
    </c:if>
    <c:if test="${sessionScope.username!=null }">            <!--如果已登录，则显示欢迎信息-->
    <table width="100%" border="0" cellpadding="0" cellspacing="0">
        <tr><td width="94%" height="31" align="center">
                ${sessionScope.userTrueName }您好!<br>您现在可以购物了!</td>
        </tr>
        <tr><td height="24" align="center">
                <input name="Submit22" type="button" class="btn_grey" value="修改资料"
                    onClick="window.location.href='modifyMember.lzw';"></td>
        </tr>
```

```
        <tr><td height="24" align="center">
                <input name="Submit32" type="button" class="btn_grey"
        value="退出登录" onClick="window.location.href='login.lzw?loginOut=true';"> </td>
        </tr>
    </table>
    </c:if></td></tr>
</table>
</form>
```

（2）创建 LoginController 类，该类是处理会员注册业务的控制器，它继承了 Spring 的 SimpleFormController 类，并重写了该类的部分方法来实现业务处理。首先，在 LoginController 类的构造方法中调用 setCommandClass()方法设置 UserLoginForm 表单类；然后，在 onSubmit()方法中处理页面请求的业务逻辑，并返回数据模型和视图对象；最后，在 showForm()方法中处理会员注销请求。关键代码如下：

例程 26 代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\LoginController.java

```
public class LoginController extends SimpleFormController {
    private Dao dao;
    public LoginController(){
        setCommandClass(UserLoginForm.class);                    //设置表单类
    }
    protected ModelAndView onSubmit(HttpServletRequest request,    //处理登录请求
            HttpServletResponse response, Object command, BindException errors)
            throws Exception {
        HttpSession session = request.getSession();
        UserLoginForm form = (UserLoginForm) command;
        TbMember user = dao.getUser(form.getUsername(), form.getPassword());
        if(user!=null){                                          //如果数据库中存在登录会员，实现登录
            session.removeAttribute("manager");
            session.setAttribute("username", form.getUsername());
            session.setAttribute("userTrueName", user.getTrueName());
            return new ModelAndView(new RedirectView("index.lzw"));
        }else{
            //否则返回登录错误
            return new ModelAndView(new RedirectView("index.lzw"),"loger","error");
        }
    }
    protected ModelAndView showForm(HttpServletRequest request, HttpServletResponse response,
BindException errors) throws Exception {                         //处理用户注销请求
        HttpSession session=request.getSession();
        String loginOut=request.getParameter("loginOut");
        if(loginOut!=null&&loginOut.equalsIgnoreCase("true")){
            session.invalidate();                                //销毁 session 会话
        }
        return new ModelAndView(new RedirectView("index.lzw"));
    }
    …//省略部分代码
}
```

（3）在 controller-config.xml 文件中配置登录控制器的定义信息，设置该控制器的 Dao、formView 和 successView 属性，它们分别是数据库操作类、表单视图和登录成功的视图。关键代码如下：

例程 27　代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\controller-config.xml

```
<!--  登录控制器-->
<bean id="loginController" class="com.lzw.LoginController">
❶   <property name="dao">
        <ref bean="daoProxyFactory" />                    <!--从 Dao 代理工厂获取 Dao 对象-->
    </property>
❷   <property name="formView">                            <!--设置表单视图-->
        <value>index</value>
    </property>
❸   <property name="successView">                        <!--设置登录成功的视图-->
        <value>index</value>
    </property>
</bean>
```

代码贴士

❶ dao：采用依赖注入原则初始化登录控制器的 Dao 属性。

❷ formView：注入登录控制器的表单视图属性。

❸ successView：注入登录控制器的成功视图。

7.8　购物模块设计

7.8.1　购物模块概述

电子商城的购物模块主要用于辅助顾客完成购物的流程。当顾客选择并购买了商品，这些商品将被放置在购物车中，可以单击购物车中的“继续购物”超链接继续购买其他商品，也可以单击购物车中的“收银结账”超链接完成商品采购。

1．购物车

购物车主要包括添加购物商品、查看购物车、商品数量的修改、移除指定商品、清空购物车等功能。当顾客购买了所需的全部商品，可以单击“收银结账”超链接，完成商品购物。购物车的页面效果如图 7.21 所示。

2．收银台

收银台功能主要用于购物的收银结账，当顾客购物并到收银台结账后，才完成一次购物流程。收银台将为顾客生成订单，为顾客保存购买的商品信息、订单信息和订单号。收银台页面中包含顾客的会员基本信息，除用户名不可更改外，还需要填写当前的联系地址、邮政编码和联系电话等信息。收银台结账页面如图 7.22 所示。

图 7.21　购物车页面

图 7.22　收银台页面

7.8.2　购物模块技术分析

购物模块的购物车和收银台分别使用了 CartController 和 CartCheckoutController 控制器处理相应的业务逻辑。

购物车的控制器继承了 Spring 的 MultiActionController 控制器，所以它是一个多动作控制器，控制器中的不同方法可以分别处理购物车的各种操作，例如修改购物商品数量、商品退回等。

收银台使用了 Spring 的表单控制器，它继承了 SimpleFormController 类，拥有了表单处理的能力，它可以处理会员的表单信息，并返回该订单的编号，供会员日后查询。

7.8.3　购物车的实现过程

购物车使用的数据表：tb_goods

实现购物车功能需要经过创建购物车页面、编写购物车控制器和配置购物车控制器等步骤，本节将详细介绍这些步骤的实现方法。

1．创建购物车页面

购物车页面由购物商品列表和相应的操作链接组成。购物商品列表是一个 Form 表单，它包括商品数量的修改和商品退回操作，它们分别由控制器的 cartModify()和 cartMove()方法来实现相应的业务逻辑。“收银结账”超链接将转到收银台功能页面中完成本次购物的流程。“清空购物车”超链接将执行控制器的 cartClear()方法清空购物车中的所有商品。创建购物车页面的关键代码如下：

例程 28　代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\jsp\cart_see.jsp

```
<form method="post" action="cartModify.lzw" name="form1">
    <table width="92%" height="48" border="0" align="center" cellpadding="0" cellspacing="0">
        <tr align="center" valign="middle">                    <!--商品列表的表头-->
            <td height="27" class="tableBorder_B1">编号</td>
            <td height="27" class="tableBorder_B1">商品编号</td>
            <td class="tableBorder_B1">商品名称</td>
```

```
                <td height="27" class="tableBorder_B1">单价</td>
                <td height="27" class="tableBorder_B1">数量</td>
                <td height="27" class="tableBorder_B1">金额</td>
                <td class="tableBorder_B1">退回</td>
            </tr>
            <c:forEach varStatus="idx" var="goods" items="${sessionScope.cart }">
                <c:set var="sum" value="${sum+goods.number*goods.nowPrice }" />
                <tr align="center" valign="middle">                    <!--获取并显示购物车中的信息-->
                    <td width="32" height="27">${idx.index+1 }</td>
                    <td width="109" height="27">${goods.ID }</td>
                    <td width="199" height="27">${goods.goodsName }</td>
                    <td width="59" height="27">￥${goods.nowPrice }</td>
                    <td width="51" height="27">
                        <input name="num${idx.index }" size="7" type="text"
                            class="txt_grey" value="${goods.number }" onBlur="check(this.form)"></td>
                    <td width="65" height="27">￥${goods.nowPrice*goods.number }</td>
                    <td width="34">
                        <a href="cartMove.lzw?ID=${idx.index }">
                        <img src="images/del.gif" width="18" height="18"></a></td>
                </tr>
            </c:forEach>
        </table>
    </form>
    <table width="100%" height="52" border="0" align="center" cellpadding="0" cellspacing="0">
        <tr align="center" valign="middle">
            <td height="21" class="tableBorder_B1">  </td>
            <td height="21" colspan="-3" align="left" class="tableBorder_B1">
                合计总金额：￥${sum }</td>
        </tr>
        <tr align="center" valign="middle">                    <!--创建购物车的操作链接-->
            <td height="21" colspan="2">
                <a href="index.jsp">继续购物</a> |
                <a href="cartCheckout.lzw">收银结账</a> |
                <a href="cartClear.lzw">清空购物车</a> |
                <a href="#">修改数量</a>
            </td>
        </tr>
    </table>
```

2．编写购物车控制器

编写购物车控制器的步骤如下：

（1）创建 CartController 类，该类继承了 Spring 的 MultiActionController 类，它是购物车的控制器，负责处理购物车的操作请求。

在该控制器中编写 cartAdd()方法，该方法将用于处理添加购物商品到购物车的请求。当顾客单击

某商品的“购买”按钮，该商品将会通过 cartAdd()方法添加到购物车中。cartAdd()方法首先获取商品的 ID 编号并调用 Dao 属性的 getGoods()方法获取指定 ID 编号的商品实体对象，然后将商品实体对象的关键属性封装到 GoodsElement 类的实例对象中，最后将这些商品信息添加或累加到 Session 会话的 cart 属性中。cartAdd()方法的关键代码如下：

例程 29 代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\CartController.java

```
public ModelAndView cartAdd(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
    HttpSession session = request.getSession();                    //获取 Session 会话对象
    String id = request.getParameter("goodsID");                   //从请求中获取商品 ID
    TbGoods goods = dao.getGoods(Long.valueOf(id));                //调用 getGoods() 方法获取指定 ID 的商品
❶   GoodsElement myGoodsElement = new GoodsElement();              //封装商品的关键属性
❷   myGoodsElement.setID(goods.getId());
❸   myGoodsElement.setGoodsName(goods.getGoodsName());
❹   myGoodsElement.setNowPrice(goods.getNowPrice());
❺   myGoodsElement.setNumber(1);
    List<GoodsElement> cart = (List<GoodsElement>) session
            .getAttribute("cart");                                 //从 Session 会话中获取购物商品列表
    boolean Flag = true;
    if (cart == null) {                                            //如果购物商品列表为空
        cart = new ArrayList<GoodsElement>();                      //创建一个购物商品列表
    } else {
        for (int i = 0; i < cart.size(); i++) {
            GoodsElement goodsitem = cart.get(i);
            if (goodsitem.ID == myGoodsElement.ID) {               //如果购物商品列表中包含购买的商品
                goodsitem.number++;                                //将商品列表中该商品的数量累加
                Flag = false;
            }
        }
    }
    if (Flag) {                                                    //如果购物列表是新创建的
        cart.add(myGoodsElement);                                  //将封装的商品信息添加到购物列表中
        session.setAttribute("cart", cart);                        //将购物列表添加到 Session 会话中
    }
    return new ModelAndView(new RedirectView("cartSee.lzw"));      //转发到购物车页面中
}
```

代码贴士

❶ GoodsElement：商品信息的封装类。

❷ setID()：设置封装类中的商品 ID 编号。

❸ setGoodsName()：设置商品封装类中的商品名称。

❹ setNowPrice()：设置商品封装类中的商品单价信息。

❺ setNumber()：设置商品数量。

（2）在控制器中编写 cartModify()方法，该方法将处理修改购物车中商品数量的请求。cartModify()

方法首先获取顾客的购物列表，然后从请求对象中分别获取购物车中每个商品的数量并更新到购物列表，最后如果某个商品的数量小于或等于 0，它将被从购物列表中移除。cartModify()方法的关键代码如下：

例程 30　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\CartController.java

```
public ModelAndView cartModify(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
    HttpSession session = request.getSession();
    List<GoodsElement> cart = (List<GoodsElement>) session
            .getAttribute("cart");                          //获取购物车的商品列表
    for (int i = 0; i < cart.size(); i++) {
        GoodsElement myGoodsElement = cart.get(i);
        String num = request.getParameter("num" + i);       //从请求中获取每个商品的修改数量
        int newnum = Integer.parseInt(num);
        myGoodsElement.number = newnum;                     //更新商品数量
        if (newnum <= 0) {                                  //如果商品数量小于或等于 0
            cart.remove(myGoodsElement);                    //将该商品从购物车中移除
        }
    }
    return new ModelAndView(new RedirectView("cartSee.lzw")); //转到购物车页面
}
```

（3）在控制器中编写 cartClear()方法，该方法将处理清空购物车的请求。这个方法的实现非常简单，从 Session 会话中将购物车属性 cart 直接移除，那么购物车的购物列表就不存在了，自然也就实现了购物车清空。关键代码如下：

例程 31　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\CartController.java

```
public ModelAndView cartClear(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
    HttpSession session = request.getSession();             //获取 Session 会话
    session.removeAttribute("cart");                        //从会话中删除购物车属性
    return new ModelAndView("cartSee");                     //转到购物车页面
}
```

（4）在控制器中编写 cartMove()方法，该方法用于处理从购物车中退回单个商品的请求。实现单个商品的移除，首先需要获取该商品的 ID 编号，然后调用 remove()方法从购物列表中移除该商品。关键代码如下：

例程 32　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\CartController.java

```
public ModelAndView cartMove(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
    HttpSession session = request.getSession();
    List cart = (List) session.getAttribute("cart");
    int id = Integer.parseInt(request.getParameter("ID"));  //获取商品 ID 编号
    cart.remove(id);                                        //移除指定编号的商品
```

```
        return new ModelAndView(new RedirectView("cartSee.lzw"));
    }
```

3．配置购物车控制器

购物车控制器的配置信息分为控制器定义和请求映射两部分。其中购物车控制器的请求映射已经在 7.6 节公共模块设计中介绍过，它被映射处理以“/cart”为前缀的所有请求。这里将介绍购物车控制器的定义。

购物车控制器首先需要在 controller-config.xml 配置文件中定义购物车控制器的方法解析器，在解析器中定义控制器的不同方法所处理的请求，然后定义购物车的控制器，并为控制器分别注入方法解析器和 Dao 属性。关键代码如下：

例程 33 代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\controller-config.xml

```
<!--  购物车控制器-->
<bean id="cartMethodResolver"
    class="org.springframework.web.servlet.mvc.multiaction.PropertiesMethodNameResolver">
    <property name="mappings">
        <props>
            <prop key="/cartAdd.lzw">cartAdd</prop>              <!--cartAdd()方法处理的请求-->
            <prop key="/cartModify.lzw">cartModify</prop>        <!-- cartModify()方法处理的请求-->
            <prop key="/cartClear.lzw">cartClear</prop>          <!-- cartClear()方法处理的请求-->
            <prop key="/cartMove.lzw">cartMove</prop>            <!-- cartMove()方法处理的请求-->
        </props>
    </property>
</bean>
<bean id="cartController" class="com.lzw.CartController">        <!--配置控制器-->
    <property name="methodNameResolver">
        <ref local="cartMethodResolver" />                       <!--为控制器注入方法解析器-->
    </property>
    <property name="dao">
        <ref bean="daoProxyFactory" />                           <!--为控制器注入 Dao 属性-->
    </property>
</bean>
```

7.8.4 收银台的实现过程

收银台使用的数据表：tb_goods、tb_order、tb_order_detail

实现收银台功能需要经过创建收银台页面、编写收银台控制器、配置收银台的控制器和控制器映射等步骤，其中控制器映射已经在 7.6 节公共模块设计中介绍过，收银台控制器被映射处理“/cartCheckout.lzw”请求，本节将详细介绍其他步骤的实现方法。

1．创建收银台页面

创建收银台页面，在该页面中创建一个表单，表单的 Action 属性（即请求路径）设置为 cartCheckout.lzw，表单中包含用户名、真实姓名、联系地址等会员基本信息的字段。表单中的字段定义如表 7.2 所示。

表 7.2　收银台的表单定义

字段描述	字段名称	字段类型	只　读	字段描述	字段名称	字段类型	只　读
用户名	username	Text	是	联系电话	tel	Text	否
真实姓名	truename	Text	是	付款方式	pay	select	否
联系地址	address	Text	否	运送方式	carry	select	否
邮政编码	postcode	Text	否	备注	bz	textarea	否

2．编写收银台控制器

收银台控制器是 Spring 的表单控制器的子类，它具有处理表单的能力。收银台页面的表单包含订单的会员信息和联系方式，这些内容都由表单类 CartCheckoutForm 封装并传递给控制器，然后由控制器将表单信息和商品信息通过事务保存到数据库中。实现收银台控制器的步骤如下：

（1）创建 CartCheckoutController 类，在该类的构造方法中调用 setCommandClass()方法定义控制器的表单类。关键代码如下：

例程 34　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\CartCheckoutController.java

```
public class CartCheckoutController extends SimpleFormController {
    private Dao dao;                                          //定义 Dao 属性
    public CartCheckoutController() {
        setCommandClass(CartCheckoutForm.class);              //设置控制器的表单类
    }
…//省略部分代码
}
```

（2）在控制器中编写 onSubmit()方法，在该方法中首先获取用户的折扣信息，并将折扣信息和其他会员信息添加到订单主表的实体对象中。关键代码如下：

例程 35　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\CartCheckoutController.java

```
protected ModelAndView onSubmit(HttpServletRequest request,
        HttpServletResponse response, Object command, BindException errors)
        throws Exception {
    HttpSession session = request.getSession();
    CartCheckoutForm form = (CartCheckoutForm) command;
    TbMember user = dao.getUser(form.getUsername(), null);    //调用 getUser() 方法获取会员对象
    if (user == null) {                                       //如果会员不存在
        session.invalidate();                                 //销毁 Session 会话
        return new ModelAndView(getFormView());               //返回表单页面
    }
    float rebate = dao.getUserRebate(form.getUsername());     //获取会员的折扣
    List<GoodsElement> cart = (List<GoodsElement>) session
            .getAttribute("cart");                            //获取购物车
    int number = 0;
    double nowprice = (float) 0.0;
    double sum = (float) 0;
    double Totalsum = (float) 0;
    long ID = -1;
    short bnumber = 0;
```

```
    if (cart != null) {
        bnumber = (short) cart.size();
    }
    TbOrder order = new TbOrder();                          //创建订单主表的实体对象
    order.setAddress(form.getAddress());                    //用表单类的数据初始化订单主表的实体对象
    order.setBnumber(bnumber);
    order.setBz(form.getBz());
    order.setCarry(form.getCarry());
    order.setPay(form.getPay());
    order.setPostcode(form.getPostcode());
    order.setRebate(rebate);
    order.setTel(form.getTel());
    order.setTruename(form.getTruename());
    order.setUsername(form.getUsername());
    order.setOrderDate(new Date());
```

（3）在循环中遍历购物车中的所有商品，将这些商品添加到订单明细表的实体对象中。关键代码如下：

```
Set<TbOrderDetail> tbOrderDetails = order.getTbOrderDetails();    //获取订单明细表的集合对象
for (int i = 0; i < bnumber; i++) {                                //遍历购物车中的商品
    GoodsElement myGoodsElement = cart.get(i);
    ID = myGoodsElement.getID();
    nowprice = myGoodsElement.getNowPrice() * rebate;              //计算商品的折扣价格
    number = myGoodsElement.number;
    sum = nowprice * number;                                       //计算购物车中打折后的商品总价格
    TbOrderDetail details = new TbOrderDetail();                   //订单明细表的实体对象
    details.setNumber(number);                                     //初始化订单明细表的实体对象
    details.setPrice(nowprice);
    details.setTbOrder(order);
    TbGoods goods = dao.getGoods(ID);
    details.setTbGoods(goods);
    tbOrderDetails.add(details);
    Totalsum = Totalsum + sum;                                     //计算此次购物的消费总金额
}
```

（4）更新会员信息，并调用 Dao 属性的 insertObject()方法，在事务中将订单主表、订单明细表的实体对象和会员表的实体对象添加或更新到数据库中。关键代码如下：

```
user.setAmount(user.getAmount() == null ? 0.0 : user.getAmount()+ Totalsum);
int userGrade = dao.getGrade(user.getAmount());
if (user.getGrade() == null || userGrade > user.getGrade())
    user.setGrade(userGrade);                                      //更新会员等级
dao.insertObject(user, order);                                     //在事务中完成订单保存和会员的更新
session.removeAttribute("cart");                                   //清空购物车
return new ModelAndView(new RedirectView("cartSee.lzw?orderId="
        + order.getOrderId()));
}
```

3．配置收银台控制器

在 controller-config.xml 文件中定义收银台控制器的配置信息，并为控制器注入 Dao 属性和相应的视图属性。关键代码如下：

例程 36　代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\controller-config.xml

```
<!--  购物结账-->
<bean id="cartCheckoutController"
      class="com.lzw.CartCheckoutController">
      <property name="dao">
            <ref bean="daoProxyFactory" />
      </property>
      <property name="formView">
            <value>cartCheckout</value>
      </property>
      <property name="successView">
            <value>index</value>
      </property>
</bean>
```

7.8.5　单元测试

在开发完购物模块后一定要对其进行相应的测试，以确保模块能正常运行。收银台控制器所实现的业务逻辑比较复杂，它需要在事务中同时处理订单数据表、订单详细数据表和会员数据表 3 个数据表的存储操作，只要有一个数据表不能存储，这个事务中的所有任务都会回滚到原始状态，不对数据库做任何操作。

为确保收银台控制器能正常地存储数据，就必须保证前台表单所提交的数据格式。中文乱码问题是 Web 程序最频繁出现的错误，也是最容易忽略的错误。为方便测试，现在重现收银台控制器读取页面数据的关键代码：

```
❶  TbOrder order = new TbOrder();                    //创建订单主表的实体对象
❷  order.setAddress(form.getAddress());              //用表单类的数据初始化订单主表的实体对象
❸  order.setBnumber(bnumber);
❹  order.setBz(form.getBz());
❺  order.setCarry(form.getCarry());
❻  order.setPay(form.getPay());
❼  order.setPostcode(form.getPostcode());
❽  order.setRebate(rebate);
❾  order.setTel(form.getTel());
   order.setTruename(form.getTruename());
   order.setUsername(form.getUsername());
❿  order.setOrderDate(new Date());
```

代码贴士

❶ TbOrder：订单主表的实体类。

❷ setAddress()：设置订单实体中地址信息的方法。

❸ setBnumber()：设置订单实体中数量信息的方法。
❹ setBz()：设置订单实体中包装信息的方法。
❺ setCarry()：该方法用于设置订单实体的送货方式信息。
❻ setPay()：该方法用于设置订单实体中的支付方式信息。
❼ setPostcode()：该方法用于设置订单实体中的邮编信息。
❽ setRebate()：该方法用于设置订单实体中的折扣信息。
❾ setTel()：该方法用于设置订单实体中的联系电话信息。
❿ setOrderDate()：该方法用于设置订单实体的日期信息。

form 是控制器的表单类，它封装了收银台页面的表单数据，这些数据又被封装到 TbOrder 表单类中，正常情况下，这已经完成了获取前台表单数据的业务。但是实际的数据结果不能实现正常的收银结账工作，因为从前台表单所接收的数据是 ISO-8859-1 编码格式，这些数据以中文编码存储，就形成了所谓的中文乱码。

解决该问题的方法是重写父类（即 Spring 的 SimpleFormController 表单控制器）的 handleRequestInternal() 方法，在该方法中设置请求对象中的数据使用中文编码格式，而不是 ISO-8859-1 编码格式。关键代码如下：

例程 37 代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\CartCheckoutController.java

```
protected ModelAndView handleRequestInternal(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
    request.setCharacterEncoding("GBK");                    //设置请求对象的字符编码格式
    return super.handleRequestInternal(request, response);
}
```

7.9 商品管理模块设计

7.9.1 商品管理模块概述

商品信息的管理对电子商城来说十分重要，一个好的电子商务系统必须要有一个强大的商品管理模块。电子商务网站系统的商品管理模块主要实现对商品信息的管理，包括分页显示商品信息、添加商品信息、修改商品信息、删除商品信息和商品的分类管理等功能。本节将以商品列表和商品添加功能为例，介绍商品管理模块。

1．商品列表

商品列表是后台商品管理页面之一，它负责分页显示数据库中所有的商品，并为每个商品提供“修改”和“删除”超链接。商品列表页面显示的商品信息包括商品名称、简介、是否新品、是否特价等。商品列表页面的效果如图 7.23 所示。

2．商品添加

商品添加功能主要用于添加新商品到数据库，添加到数据库的商品信息将被显示到电子商城的前台页面中。新添加的商品信息需要填写所属分类、商品名称、图片文件、定价等商品信息。另外，如

果在添加商品时，指定该商品是新品和特价商品，那么这个新添加的商品将显示到特价商品页面和新品上架页面中。添加商品信息页面如图 7.24 所示。

图 7.23　后台商品列表页面

图 7.24　后台添加商品信息页面

7.9.2　商品管理模块技术分析

商品管理模块的商品列表和商品添加分别使用了 ProductManagerController 和 GoodsAddController 控制器处理相应的业务逻辑。

商品列表的控制器继承了 Spring 的 AbstractCommandController 类，成为一个命令控制器，该控制器将页面的请求参数封装到命令对象中，控制器从命令对象中获取指令，不再访问请求对象。该控制器的命令对象只包括一个 page 属性，用于接收和传递分页命令。

商品添加使用了 Spring 的表单控制器，它继承了 SimpleFormController 类，拥有了表单处理的能力，它可以接收商品添加页面的表单信息，并将商品添加到数据库中。

7.9.3　商品列表的实现过程

商品列表使用的数据表：tb_goods

商品列表功能由控制器提供页面的分页数据，所以它的实现步骤和其他功能相反，实现步骤为编写商品列表控制器、配置商品列表控制器、创建商品列表页面。

1．编写商品列表控制器

商品列表控制器是 Spring 命令控制器的子类，它将页面的请求参数封装到命令对象中，控制器从命令对象中获取指令，不再访问请求对象。实现商品列表控制器的步骤如下：

（1）创建 CartCheckoutController 类，在该类的构造方法中调用 setCommandClass()方法定义控制器的表单类。关键代码如下：

例程 38　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\manager\ProductManagerController.java

```
public class ProductManagerController extends AbstractCommandController {
    private Dao dao;                                          //定义 Dao 属性
```

```
    private int pageSize = 10;                                          //定义单页数据大小属性
    public ProductManagerController() {
        setCommandClass(ProductCommand.class);                          //设置表单类
    }
…//省略部分代码
}
```

（2）重写命令控制器的 handle()方法，在该方法中处理商品数据分页的业务逻辑，并将分页的数据传递到商品列表页面中。关键代码如下：

例程 39 代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\manager\ProductManagerController.java

```
protected ModelAndView handle(HttpServletRequest request,
        HttpServletResponse response, Object command, BindException errors)
        throws Exception {
    ProductCommand form = (ProductCommand) command;
    Map model = new HashMap();
    List list = dao.getGoodsByQuery("from TbGoods order by intime Desc");  //调用 getGoodsByQuery()方法
    int recordCount = list.size();                                      //获取商品总数
    int maxPage = (recordCount - 1) / pageSize + 1;                     //计算显示全部商品需要的页数
    int page = Integer.parseInt(form.getPage());                        //获取页面传递的当前页数
    if (page < 1) {                                                     //验证并修正当前页数
        page = 1;
    } else {
        if (page > maxPage) {
            page = maxPage;
        }
    }
    int currentRecord = (page - 1) * pageSize;                          //计算当前页的记录编号
    int step = currentRecord + pageSize >= recordCount ? recordCount    //防止记录越界
            : currentRecord + pageSize;
    List list2 = list.subList(currentRecord, step);                     //获取当前页的分页数据
    model.put("goodsList", list2);
    model.put("page", page);
    model.put("maxPage", maxPage);
    return new ModelAndView("indexM", model);                           //返回页面需要的数据模型
}
```

技巧 在设计分页显示功能时，计算显示数据的总页数的最简单的公式是“（数据总记录数-1）/单页记录数+1”。

2．配置商品列表控制器

在 controller-config.xml 文件中定义商品列表控制器的配置信息，并为控制器注入 Dao 属性。关键代码如下：

例程 40 代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\managerController-config.xml

```
<bean id="indexMController" class="com.lzw.manager.ProductManagerController">
    <property name="dao">
```

```
        <ref bean="daoProxyFactory" />                    <!--为控制器注入 Dao 属性-->
    </property>
</bean>
```

3．编写商品列表页面

商品列表页面由 content.jsp 文件定义，它将每页的商品信息放置在表格中，并为每个商品信息添加了“修改”和“删除”超链接，另外在表格的底部添加了控制分页的超链接。关键代码如下：

例程 41　代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\manage\content.jsp

```
<table width="100%" height="48"   border="1" cellpadding="0" cellspacing="0"
        bordercolor="#FFFFFF" bordercolordark="#CCCCCC" bordercolorlight="#FFFFFF">
<tr bgcolor="#eeeeee">
  <td width="22%" height="24" align="center">商品名称</td>
  <td width="40%" align="center">简介</td>
  <td width="11%" align="center">是否新品</td>
  <td width="11%" align="center">是否特价</td>
  <td width="8%" align="center">修改</td>
  <td width="8%" align="center">删除</td>
</tr>
<c:forEach var="goods" items="${goodsList }">
  <tr style="padding:5px;">
    <td height="20" align="center">
        <a href="goodsDetailM.lzw?id=${goods.id }">${goods.goodsName }</a></td>
    <td align="center">${goods.introduce }</td>
    <td align="center">${goods.newGoods==0? '否':'是' }</td>
    <td align="center">${goods.sale==0? '否':'是' }</td>
    <td align="center"><a href="goodsModify.lzw?id=${goods.id }">
        <img src="images/modify.gif" width="15" height="15"></a></td>
    <td align="center"><a href="goodsDel.lzw?id=${goods.id }">
        <img src="images/del.gif" width="16" height="16"></a></td>
  </tr>
</c:forEach>
</table>
<table width="100%"   border="0" cellspacing="0" cellpadding="0">
<tr>
  <td align="right">当前页数：[${page }/${maxPage }] 
    <c:if test="${page > 1 }">
        <a href="indexM.lzw?page=1">第一页</a>
        <a href="indexM.lzw?page=${page-1 }">上一页</a> </c:if>
    <c:if test="${page < maxPage }">
        <a href="indexM.lzw?page=${page+1 }">下一页</a>
        <a href="indexM.lzw?page=${maxPage }">最后一页 </a> </c:if>
  </td>
</tr>
</table>
```

7.9.4 商品添加的实现过程

商品添加使用的数据表：tb_goods

实现后台的商品添加功能需要经过创建商品添加页面、编写商品添加控制器、配置控制器和控制器映射等步骤，其中控制器映射已经在 7.6 节公共模块设计中介绍过，商品添加控制器被映射处理“/goodsAdd.lzw”请求，本节将详细介绍其他步骤的实现方法。

1．创建商品添加页面

添加商品页面主要用于向数据库中添加新的商品信息，在填写商品信息的表单时，可以同时指定该商品的分类、是否新品或者是否特价商品。关键代码如下：

例程 42 代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\manage\content.jsp

```
<form action="goodsAdd.lzw" method="post" name="form1">
<table width="100%"   border="0" align="center" cellpadding="-2" cellspacing="-2" bordercolordark="#FFFFFF">
  <tr>
    <td height="27"> 所属大类：</td>
    <td width="31%" height="27"> 
    <select name="supertype" class="textarea" id="supertype" onChange="F_super(this.value)">
          <c:forEach varStatus="n" var="name" items="${typeNameList }">                    <!--设置大分类-->
          <option value="${n.index }
                <c:choose>
                      <c:when test="${n.index==superSelect }">selected</c:when>
                      <c:when test="${superSelect==null }">
                            <c:set var="superSelect" value="${n.index }"/>
                      </c:when>
                </c:choose>>${name }</option>
          </c:forEach>
    </select></td>
    <td width="13%" height="27">  所属小类：</td>
    <td width="42%" height="27"> 
      <select name="typeId" class="textarea" id="typeId">                                    <!--设置小分类-->
          <c:forEach var="type" items="${typeList[typeNameList[superSelect]] }">
          <option value="${type.id.subId }">${type.id.subType }</option>
          </c:forEach></select></td></tr>
<tr>
    <td width="14%" height="27"> 商品名称：</td>
    <td height="27" colspan="3"> 
     <input name="goodsName" type="text" class="Sytle_text" id="bookID2" size="50"> </td>
</tr>
<tr>
    <td height="41"> 图片文件：</td>
    <td height="41"> 
          <input name="picture" type="text" class="Style_upload" id="picture"></td>
    <td height="41"> 定　价：</td>
    <td height="41"> 
          <input name="price" type="text" class="Sytle_text" id="price">(元)</td>
```

```
  </tr>
  <tr>
    <td height="45"> 是否新品：</td>
    <td> 
     <input name="newGoods" type="radio" class="noborder" value="1" checked>是
     <input name="newGoods" type="radio" class="noborder" value="0">否</td>
    <td> 是否特价：</td>
    <td><input name="sale" type="radio" class="noborder" value="1" checked>是
        <input name="sale" type="radio" class="noborder" value="0">否</td>
  </tr>
  <tr>
    <td height="103"> 商品简介：</td>
    <td colspan="3"><span class="style5">  </span>
        <textarea name="introduce" cols="60" rows="5" class="textarea" id="introduce"></textarea></td>
  </tr>
  <tr>
    <td height="38" colspan="4" align="center">
     <input name="Button" type="button" class="btn_grey" value="保存" onClick="mycheck()"> 
     <input name="Submit2" type="reset" class="btn_grey" value="重置"> 
     <input name="Submit3" type="button" class="btn_grey" value="返回" onClick="JScript:history.back(-1)"></td>
  </tr>
</table>
</form>
```

2．编写商品添加控制器

商品列表控制器是 Spring 命令控制器的子类，它将页面的请求参数封装到命令对象中，控制器从命令对象中获取指令，不再访问请求对象。实现商品添加控制器的步骤如下：

（1）继承 Spring 的 SimpleFormController 控制器创建 GoodsAddController 类，在该类的构造方法中调用 setCommandClass()方法定义控制器的表单类。关键代码如下：

例程 43　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\manager\GoodsAddController.java

```
public class GoodsAddController extends SimpleFormController {
    private Dao dao;                                          //定义 Dao 属性
    public GoodsAddController() {
        setCommandClass(TbGoods.class);                       //设置控制器使用的表单类
    }
…//省略部分代码
}
```

（2）重写父类的 onSubmit()方法，在该方法中处理表单数据并调用 Dao 属性的 insertObject()方法把商品的表单数据添加到数据库中。关键代码如下：

例程 44　代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\manager\GoodsAddController.java

```
protected ModelAndView onSubmit(Object command) throws Exception {
    TbGoods form=(TbGoods) command;
    form.setIntime(new Date());
```

```
        form.setNowPrice(form.getPrice());
        dao.insertObject(form);
        return super.onSubmit(null);
    }
```

（3）重写父类的 showForm()方法，为商品添加页面提供必要的商品类别数据。该方法在控制器接收到 GET 请求时被调用，主要用于显示表单页面。关键代码如下：

例程 45 代码位置：光盘\TM\07\XinXiangShop\src\com\lzw\manager\GoodsAddController.java

```
protected ModelAndView showForm(HttpServletRequest request,
            HttpServletResponse response, BindException errors, Map controlModel)
            throws Exception {
        String superSelect=request.getParameter("superSelect");
        return new ModelAndView(getFormView(),"superSelect",superSelect);
    }
```

3．配置控制器

在 controller-config.xml 文件中定义商品添加控制器的配置信息，并为控制器分别注入 Dao 属性和相应的视图属性。关键代码如下：

例程 46 代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\managerController-config.xml

```
<!--  商品添加-->
<bean id="goodsAddController"
        class="com.lzw.manager.GoodsAddController">
        <property name="dao">
            <ref bean="daoProxyFactory" />
        </property>
        <property name="formView">
            <value>goodsAdd</value>
        </property>
        <property name="successView">
            <value>redirect:indexM.lzw</value>
        </property>
</bean>
```

7.9.5 单元测试

商品添加控制器也是 Spring 的表单控制器的子类，它和收银台控制器有同样的乱码问题，但是在收银台的单元测试中已经介绍了解决的办法，本节介绍的是商品添加需要解决的测试问题。

商品添加的业务逻辑并不复杂，但是，在商品添加到数据库后，如何返回正确的视图呢？成功地添加商品信息后，必须转到商品列表页面遍历数据库中的所有商品，表单控制器的表单页面和成功页面经常在配置该控制器时就已经指定，例如下面的关键配置代码：

```
<!--  商品添加-->
<bean id="goodsAddController" class="com.lzw.manager.GoodsAddController">
```

```
        <property name="dao">
            <ref bean="daoProxyFactory" />
        </property>
        <property name="formView">
            <value>goodsAdd</value>                          <!--指定控制器的表单页面-->
        </property>
        <property name="successView">
            <value>indexM</value>                            <!--指定控制器的成功页面-->
        </property>
</bean>
```

在指定控制器的成功页面时，大部分表单控制器都直接指定一个显示结果的视图，但是如果该视图还需要其他控制器为其提供页面数据，就会导致页面无法正常显示的结果。错误的效果如图 7.25 所示。

图 7.25　空的商品列表

解决该问题的办法是设置表单控制器的成功页面使用重定向命令，将请求重新转发给其他控制器路径。修改后的代码如下：

例程 47　代码位置：光盘\TM\07\XinXiangShop\WebRoot\WEB-INF\config\managerController-config.xml

```
<!--  商品添加-->
<bean id="goodsAddController"
        class="com.lzw.manager.GoodsAddController">
        <property name="dao">
            <ref bean="daoProxyFactory" />
        </property>
        <property name="formView">
            <value>goodsAdd</value>
        </property>
        <property name="successView">
            <value>redirect:indexM.lzw</value>          <!--指定控制器的成功页面转向到其他控制器-->
        </property>
</bean>
```

7.10　发布与运行

本系统在 Eclipse 开发工具中完成从创建开发到系统调试的整个开发过程，但是这个开发过程还有所欠缺，欣想电子商城必须发布到服务器中，并使它稳定运行，才能给本次系统开发画上一个完美的句号。本节将以 Tomcat 服务器为例，介绍如何使用 MyEclipse 发布 Web 工程到服务器中。

1．系统发布

欣想电子商城结合了 Tiles 布局、Spring 和 Hibernate 框架等技术。在 7.3.5 节中已经介绍了如何为系统添加相应技术或框架的 JAR 类包和配置 Tomcat 服务器。如果正确搭建了开发环境，就可以按以

下步骤发布项目到服务器中：

（1）单击工具栏上的按钮，将弹出如图 7.26 所示的 Project Deployments（项目发布）对话框，其 Project 下拉列表框中选择本系统的项目名称 LiXiangShop，单击 Add 按钮进行项目发布的设置。

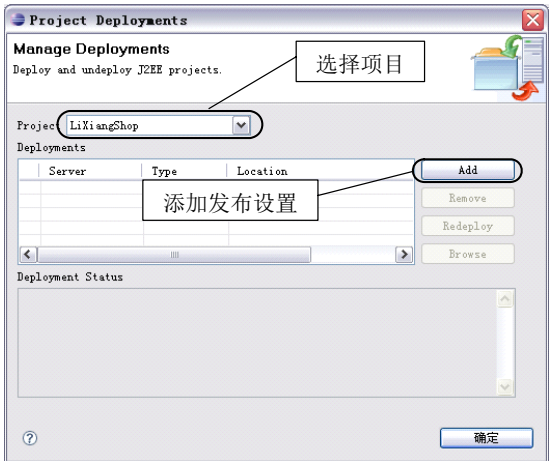


图 7.26　Project Deployments（项目发布）对话框

（2）在弹出的如图 7.27 所示的 New Deployment 对话框中，选择 Server 下拉列表框中的 Tomcat 5 服务器，Exploded Archive(development mode)和 Packaged Archive (production mode)单选按钮分别实现发布模式和产品模式，如果选择项目模式，MyEclipse 会将项目的所有文件和类库文件发布到 Web 服务器中；如果选择了产品模式，项目将会被打包成 WAR 文件，然后将该文件发布到服务器中。这里选择默认值（发布模式），单击“完成”按钮程序将自动发布到服务器中。如果需要重新发布项目，可以单击图 7.26 所示对话框中的 Redeploy 按钮。

2．系统运行

单击工具栏上的按钮右侧的下拉箭头，在弹出的菜单中选择 Tomcat 5/Start 命令，如图 7.28 所示，便可以在 MyEclipse 中启动 Web 服务器，这非常方便程序开发中的调试过程。

但是程序开发完成以后不可能一直运行在开发工具中，可以选择“开始”/“所有程序”/Apache Tomcat 5.5/Configure Tomcat 命令启动服务器控制台，并单击控制台中的 Start 按钮启动服务器，如图 7.29 所示。

图 7.27　New Deployment 对话框

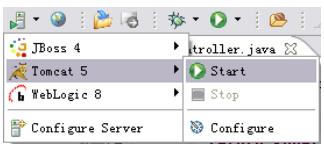


图 7.28　服务器运行菜单

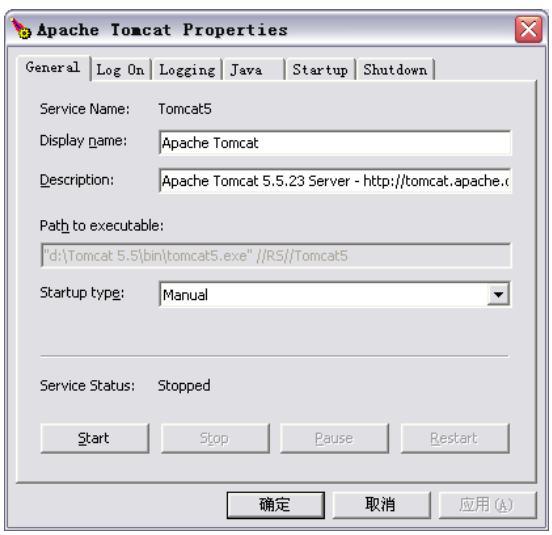


图 7.29　Tomcat 控制台

启动服务器后，在浏览器地址栏中输入程序的访问路径，例如“http://localhost:8080/ LiXiangShop”，就可以访问网站的内容。

7.11　开发技巧与难点分析

7.11.1　为 Spring 的数据源配置正确的 URL

Spring 的数据源对象负责连接数据库，所有数据库操作都需要使用该数据源对象连接数据库。在配置 Spring 的数据源对象时，需要注意数据源 URL 属性的赋值。在严格的 XML 格式中，通常使用如下关键配置代码：

```
<property name="url">
    <value>
        jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_shop
    </value>
</property>
```

在编写这段代码时或者使用 Eclipse 的代码格式化功能之后，都会使用这种严格的 XML 格式，但是访问数据库的 URL 属性值被分割成单独的一行代码，这会导致程序抛出 SQLException 异常，提示“No suitable driver found”错误信息，原因就是因为 Spring 将数据源的 URL 属性值前后的空格和换行符都赋值给 URL 属性，从而导致访问数据库的 URL 出错。

避免该错误有两种方法：

☑　重新整理 XML 格式

这种方法是去掉数据源的 URL 属性值和成对的<value>标签中的空格和换行符，使它们在一行代码中实现 URL 属性的赋值。整理后的关键配置代码如下：

```
<property name="url">
    <value>jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_shop</value>
</property>
```

☑　使用参数赋值

整理 XML 格式的方法虽然可以避免出现类似的错误，但是一个拥有良好的程序开发习惯的程序开发人员，会经常格式化自己的程序代码，如果频繁地使用 Eclipse 开发工具的代码格式化功能，在每次格式化程序代码后，都需要重新整理 XML 格式，但是这个步骤很容易被程序开发人员忽略，而使程序再次出现 SQLException 异常。

更简单实用的方法是使用<property>标签的 value 参数为 URL 属性赋值，这样开发工具的代码自动格式化功能就不会破坏 URL 属性值的格式。关键代码如下：

```
<property name="url" value="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_shop" />
```

7.11.2 为 Tiles 指定错误页面

Tiles 布局技术将页面分成不同的版块，增加了代码重用率和提高了代码的清晰度，使代码更方便阅读。但是使用 Tiles 布局网站页面时，如果出现错误信息或程序抛出异常，那么页面将显示一片空白，而不是经常见到的错误信息。这非常不利于程序的调试，并影响开发速度。

解决该问题的办法是在页面中指定处理错误信息的 JSP 页面。可以在 Tiles 的模板页面中定义处理错误信息的 JSP 页面，避免遍历修改系统中所有的页面文件。关键代码如下：

```
<%@ page contentType="text/html; charset=gb2312" errorPage="/WEB-INF/jsp/template/error.jsp" %>
```

7.12 使用 MyEclipse 生成 Hibernate 实体类和映射文件

Hibernate 是非常流行和强悍的 ORM 持久层框架，它提供了强大的数据持久化的能力，使 Java 开发人员可以按照 Java 语法开发程序的数据库持久层。但是，如果数据库比较庞大，特别是拥有大量数据表的数据库使用 Hibernate 框架，就需要编写大量的实体类和相应的映射文件，这将极大地影响程序的开发速度，但是使用 MyEclipse 工具的 Hibernate 反向工程可以从数据库中自动生成实体类和关系映射文件。本节将介绍具体的操作方法。

（1）在 Eclipse 中选择“窗口”/“首选项”命令，将弹出“首选项”对话框。在该对话框左上角的过滤器文本框中输入“drivers”，或者选择 MyEclipse/Database Explorer/Drivers 树节点，然后单击右侧的 New 按钮创建新的数据库驱动配置，如图 7.30 所示。

图 7.30 配置 MyEclipse 数据库驱动的“首选项”对话框

（2）在弹出的如图 7.31 所示的 New Driver 对话框中选择 Driver template 下拉列表框中的 Microsoft SQL Server 驱动模板，其他的驱动名和连接数据库的 URL 都会自动添加模板中预定义的内容，但是数据库的访问路径（即 URL）需要根据环境进行修改，然后单击 Add JARs 按钮添加相应的数据库驱动包，Driver Classname 下拉列表框会自动添加可选择的驱动类名称。最后单击“确定”按钮并关闭“首选项”对话框。

（3）在 Eclipse 中选择“窗口”/“打开透视图”/MyEclipse Hibernate 命令，将打开 MyEclipse Hibernate 透视图。在左上角的 DB Browser 视图中右击，在弹出的快捷菜单中选择 New 命令，将弹出如图 7.32 所示的对话框。

（4）单击“下一步”按钮，在弹出的如图 7.33 所示的对话框中选中 Display the selected schemas：单选按钮，然后单击右侧的 Add 按钮，选择当前数据库连接配置所关联的数据库，并单击“完成”按钮。

（5）在左上角的 DB Browser 视图中，选择新建立的 ShopDatabaseConnection 数据库连接，单击

鼠标右键，在弹出的快捷菜单中选择 Open Connection 命令，打开数据库连接。在弹出的对话框中输入访问数据库的用户名和密码，单击“确定”按钮。

（6）依次展开 DB Browser 视图中的 ShopDatabaseConnection/Connected to ShopDatabaseConnection/db_shop/TABLE 节点，选择需要执行反向工程的数据表，在这些数据表上单击鼠标右键，在弹出的快捷菜单中选择 Hibernate Reverse Engineering 命令，如图 7.34 所示。

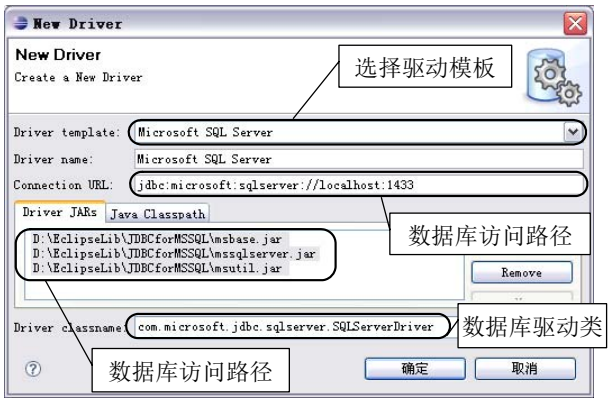


图 7.31　创建新驱动对话框

图 7.32　新建数据库连接

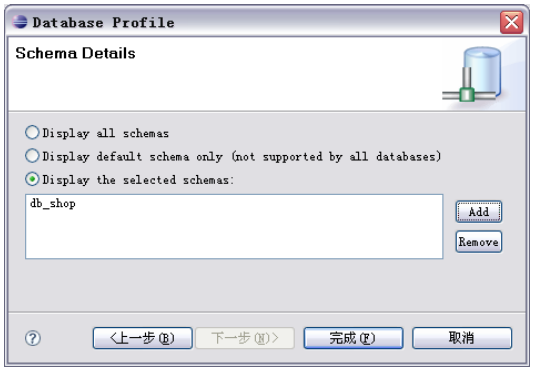


图 7.33　选择连接数据库

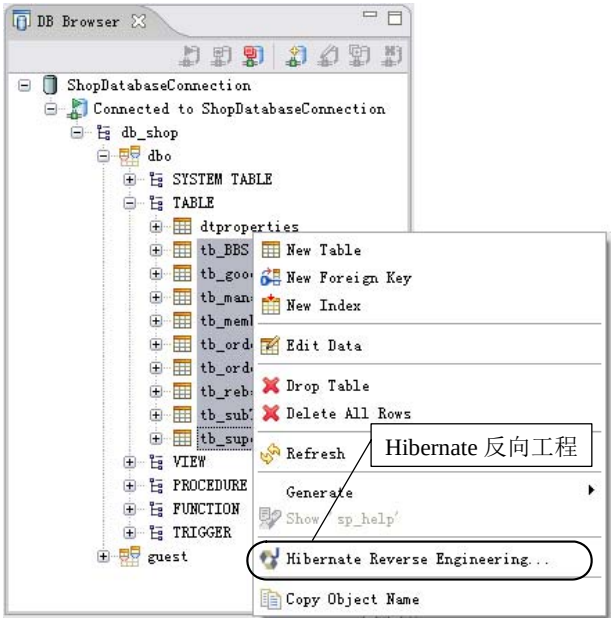


图 7.34　选择 Hibernate Reverse Engineering 命令

（7）在弹出的如图 7.35 所示的对话框中单击 Browse 按钮，选择存放实体类和映射文件的路径，例如 com.lzw.model，选中 Hibernate mapping file 和 Java Data Object 复选框，单击“完成”按钮，MyEclipse 将在指定的路径中生成指定数据表的实体类和映射文件。

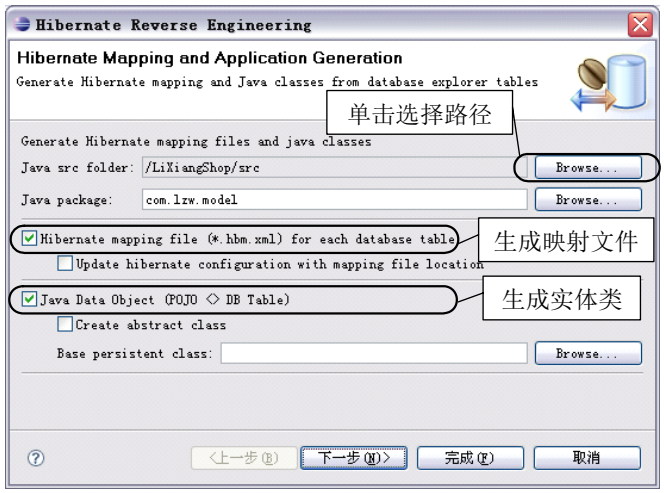


图 7.35　Hibernate Reverse Engineering（Hibernate 反向工程）对话框

7.13　本 章 小 结

本章运用软件工程的设计思想，通过一个完整的电子商务平台带领读者详细了解了一个系统的开发流程。同时，在电子商务平台的开发过程中，采用了 Tiles 布局，使整个系统的设计思路更加清晰。通过本章的学习，读者可以了解一般网站的开发流程，熟悉购物车、收银台等技术的开发思想。另外，还可以掌握开发过程中的开发技巧、程序的发布与运行。

第 8 章

超市管理系统

（Swing+ SQL Server 2005 实现）

进入21世纪，随着经济的高速发展，各行业的竞争也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、采购管理等。

通过阅读本章，可以学习到：

- 合理的设计项目数据库
- 创建带背景的窗体
- 创建平移面板
- 在表格中添加按钮
- 使用触发器级联删除数据

8.1 开 发 背 景

××超市是一家大型的超市，在市内有多家连锁，满足周边市民生活上的各种物质需求。虽然是一家连锁超市，但是各分店都有自己的仓库、员工和商品，是一个个独立的个体。近几年来，随着业务的不断壮大，货品、人员的不断增加，在管理上出现了一些问题。急需一款软件帮助超市的管理者来管理超市的相关业务。

笔者受超市经理委托，开发一款超市管理系统，开发宗旨是实现超市管理的系统化、规范化、实用化，对商品进行统一管理。

8.2 系 统 分 析

8.2.1 需求分析

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、部门管理等 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；采购订货管理用来对日常的采购订货信息进行管理；仓库入库管理用于管理各种商品入库的信息；仓库出库管理用于管理商品出库记录；人员管理实现对超市内员工的管理；部门管理实现对超市的各个独立部门进行管理。笔者通过对超市的实地考察，从管理者的角度出发，要求本系统具有以下特点：

- ☑ 具有良好的系统性能，友好的用户界面。
- ☑ 较高的处理效率，便于使用和维护。
- ☑ 采用成熟的技术开发，全系统具有较高的技术水平和较长的生命周期。
- ☑ 系统尽可能地简化超市管理员的重复工作，提高工作效率。

8.2.2 可行性分析

在超市的管理中，经常出现以下情况：

- ☑ 由于商品量较大，经常出现错登记与漏登记的情况。
- ☑ 全用传统的手工方式管理，浪费大量的纸张，且不能对商品进行快速查询。
- ☑ 只能通过现场清点商品进行了解库存信息。
- ☑ 对库存、人员、采购等内容都是分类统计，不利于管理。

因此，在超市的管理中，从经营者的角度来看，使用计算机管理系统对超市进行管理，都具有一定的必要性，以少量的人力资源、高效的工作效率、最低的误差进行管理，将使超市经营更好、顾客更信赖。

8.3　系 统 设 计

8.3.1　系统目标

根据超市管理的要求，制定超市管理系统目标如下：

- ☑ 灵活的人机交互界面，操作简单方便，界面简洁美观。
- ☑ 对采购信息进行统计分析。
- ☑ 对超市基本档案进行管理，并提供类别统计功能。
- ☑ 实现各种查询，如多条件查询、模糊查询等。
- ☑ 提供日历功能，方便用户查询日期。
- ☑ 提供超市人员管理功能。
- ☑ 系统运行稳定、安全可靠。

8.3.2　系统功能结构

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，本系统的功能结果如图 8.1 所示。

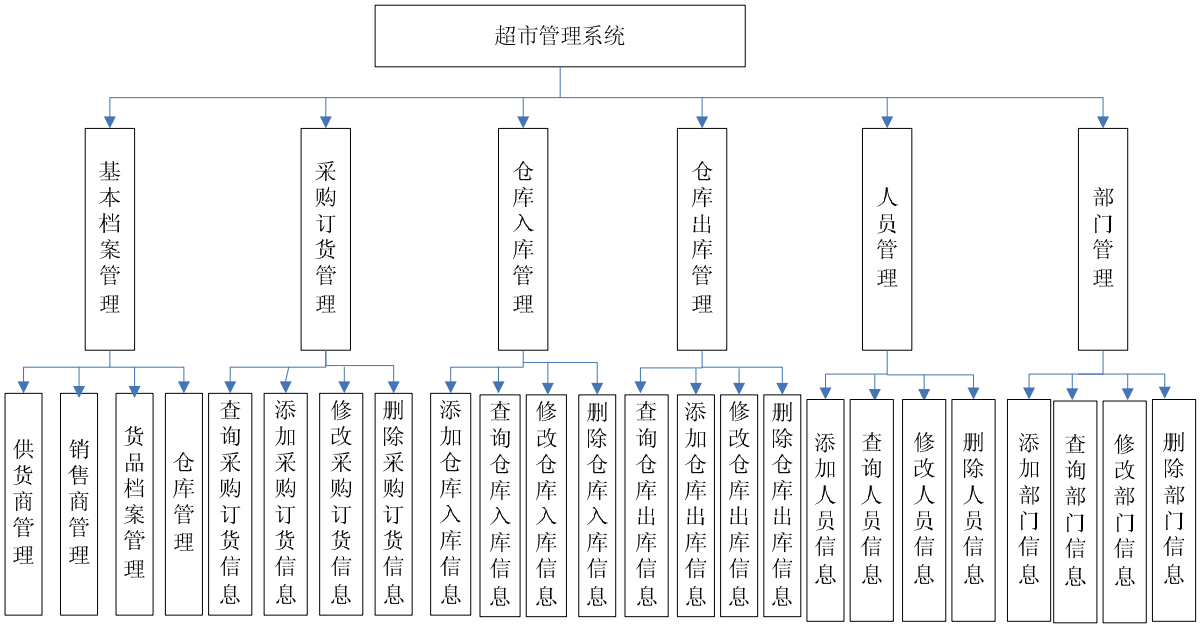


图 8.1　系统功能结构图

8.3.3　系统流程图

超市管理系统流程如图 8.2 所示。

8.3.4 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、项目的主窗体、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。

系统登录窗体效果如图 8.3 所示。

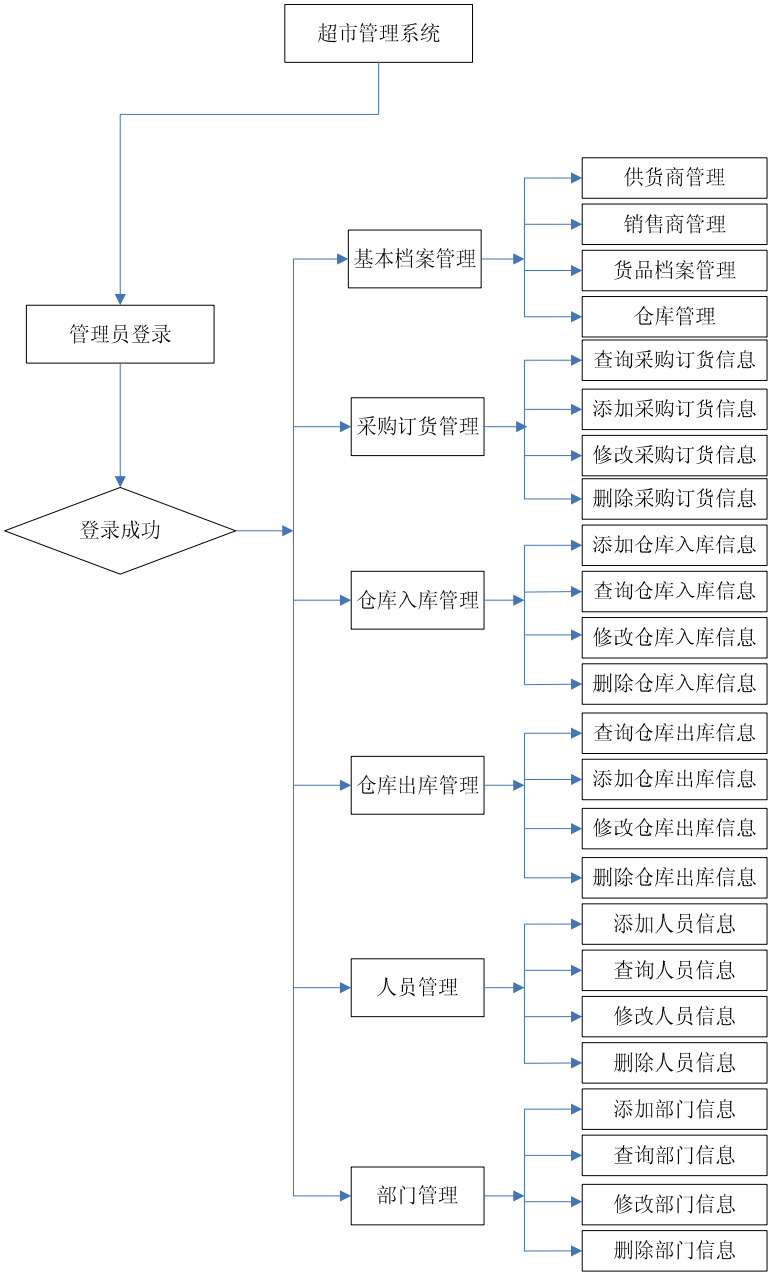


图 8.2　超市管理系统流程图

图 8.3　登录窗体（光盘\…\main\Enter.java）

当用户输入合法的用户名和密码后，单击“登录”按钮，即可进入系统主窗体，运行效果如图 8.4 所示。

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效果如图 8.5 所示。

图 8.4　系统主窗体（光盘\…\mainFrame\RemoveButtomFrame.java）

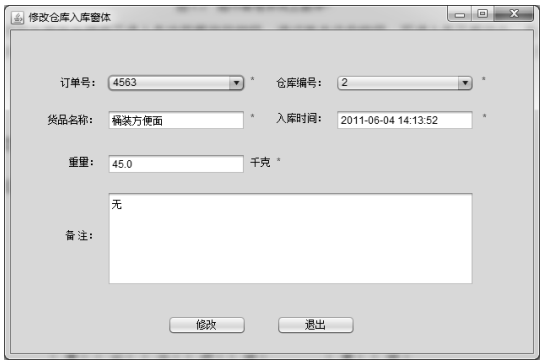


图 8.5　修改仓库入库信息（光盘\…\archives\UpdateOutDepotFrame.java）

说明　由于路径太长，因此省略了部分路径，省略的路径是“TM\08\src\com\mingrisoft”。

8.3.5　文件夹组织结构

规范系统的整体架构是一个项目开发的标准，特别是在团队进行开发项目中。在编写项目代码前，需要定制好项目的系统文件夹组织结构，使程序条理清晰，有利于后期项目的整合。在 Java 项目中，可以将不同作用、功能相类似的文件放置于同一个包中，这样做既可以保证团队开发的一致性，又可以将系统的整体结构规范化。创建完系统中可能用到的文件夹或者 Java 包后，在开发时，只需将所创建的类文件或资源文件保存到相应的文件夹中即可。超市管理系统的文件夹组织结构图如图 8.6 所示。

图 8.6　系统文件夹结构

8.4　数据库设计

数据库是超市管理系统的信息基地，其中包括仓库入库信息、供应商信息、销售商信息等内容，这些数据之间有各种直接或间接的对应关系。本节将介绍超市管理系统的数据库分析与设计过程。

8.4.1 数据库分析

为防止数据访问量增加使系统资源不足而导致系统崩溃，超市管理系统的数据库采用了独立 SQL Server 数据服务器，将数据库单独放在一个服务器中。这样即使服务器系统崩溃了，数据库服务器也不会受到影响；另外一个好处就是能够更快、更好地处理更多的数据。其数据库运行环境如下。

☑ 硬件平台
- CPU：P4 3.2GHz。
- 内存：2GB 以上。
- 硬盘空间：160GB。

☑ 软件平台
- 操作系统：Windows 2003 以上。
- 数据库：SQL Server 2005。

8.4.2 数据库概念设计

分析系统功能结构图，每个功能模块都需要操作一个或多个数据实体，如仓库入库实体对象、供应商实体对象、销售商实体对象等。本节将介绍系统中比较重要的几个数据实体，最终这些数据实体对象将创建成对应的数据表结构。

1．仓库入库实体对象

仓库入库实体包括编号、订单号、仓库编号、货品名称、入库时间等属性。编号是识别不同仓库入库实体的唯一标识，其数据类型是 int，并且是数据库自增的（它随数据库记录的增加而增加）。其余的属性都是仓库入库实体通用的特性，如订单号、仓库编号、货品名称等。仓库入库信息的实体对象如图 8.7 所示，表结构如图 8.8 所示。

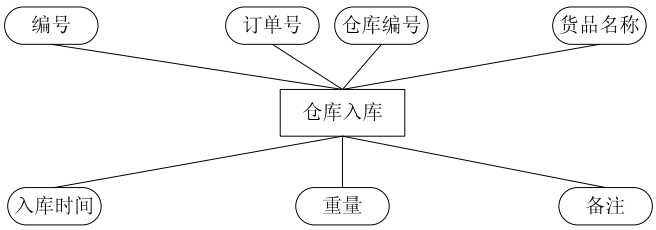


图 8.7　仓库入库实体 E-R 图

表 - dbo.tb_joinDepot

列名	数据类型	允许空
id	int	□
oid	varchar(50)	□
dId	int	□
wareName	varchar(40)	☑
joinTime	varchar(50)	☑
weight	float	☑
remark	varchar(200)	☑

图 8.8　表 tb_joinDepot

2．供应商实体对象

供应商实体对象对应着编号、客户名称、地址、联系人、联系电话、传真、备注等。供应商实体对象如图 8.9 所示，表结构如图 8.10 所示。

3．销售商实体对象

销售商实体对象拥有销售商的基本属性，这些属性包括编号、客户名称、地址、联系人、联系电话等。销售商的实体对象如图 8.11 所示，表结构如图 8.12 所示。

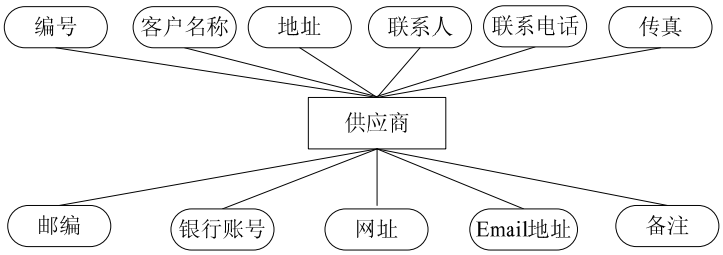


图 8.9　供应商实体 E-R 图

表 - dbo.tb_provide

列名	数据类型	允许空
id	int	
cName	varchar(20)	
address	varchar(40)	✓
linkman	varchar(50)	✓
linkPhone	varchar(20)	✓
faxes	varchar(20)	✓
postNum	varchar(10)	✓
bankNum	varchar(30)	✓
netAddress	varchar(30)	✓
emaillAddress	varchar(50)	✓
remark	varchar(200)	✓

图 8.10　表 tb_category

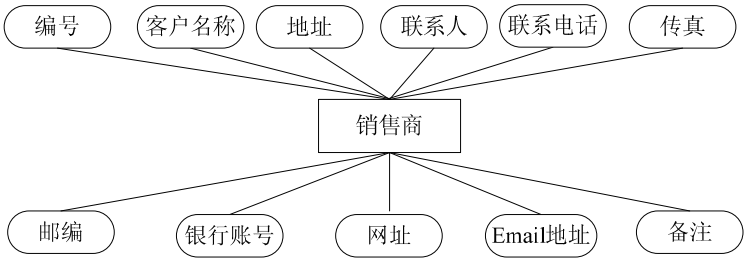


图 8.11　销售商实体 E-R 图

表 - dbo.tb_sell

列名	数据类型	允许空
id	int	
sellName	varchar(50)	
address	varchar(50)	✓
linkman	varchar(50)	✓
linkPhone	varchar(50)	✓
faxNum	varchar(50)	✓
postNum	varchar(50)	✓
bankNum	varchar(50)	✓
netAddress	varchar(50)	✓
emaillAddress	varchar(50)	✓
remark	varchar(50)	✓

图 8.12　表 tb_selldetail

8.5　公共类设计

在 Java 程序开发中，如果一个功能反复被调用，可以将这个功能抽取出来封装为一个类作为公共类，在需要此功能的地方通过此类进行实现。公共类实质是代码重用的一种方式，在面向对象的开发模式中经常被使用，它可以简化程序中的代码，提高程序的可读性。本节将向读者介绍超市管理系统中的公共类设计。

8.5.1　连接数据库公共类

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 GetConnection 类，在该类的构造方法中加载数据库驱动。关键代码如下：

例程 01　代码位置：光盘\TM\08\src\com\mingrisoft\dao\GetConnection.java

```
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的方法 getCon()，该方法返回值为 Connection 对象。关键代码如下：

例程 02　代码位置：光盘\TM\08\src\com\mingrisoft\dao\GetConnection.java

```
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
}
```

8.5.2　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的方法 getDateTime()。关键代码如下：

例程 03　代码位置：光盘\TM\08\src\com\mingrisoft\util\GetDate.java

```
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
}
```

8.6　登录模块设计

系统登录是一个对用户身份验证的过程，只有登录成功的用户才可以对系统进行操作，否则不能对系统进行管理维护，它是系统的一道安全门。

8.6.1　登录模块概述

运行程序，首先进入系统登录窗体。为了使窗体中的各个组件摆放得更加随意美观，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 8.3.4 节中的图 8.3。

8.6.2　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)
```

graphics：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。关键代码如下：

例程 04　代码位置：光盘\TM\08\src\com\mingrisoft\main\BackgroundPanel.java

```
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);
        }
}
```

8.6.3 登录模块的实现过程

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，该窗体的设计如图 8.13 所示。

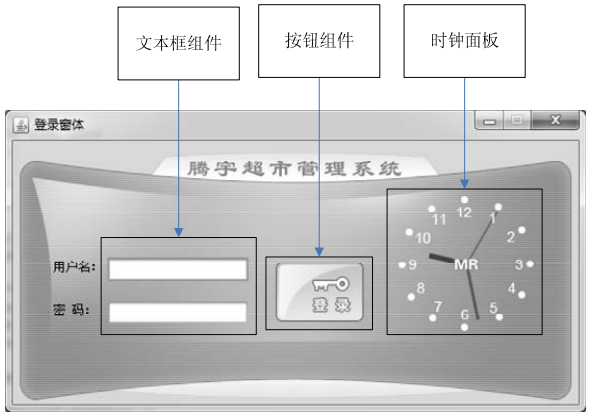


图 8.13　登录窗体设计效果

说明　本系统中显示时钟的面板也是笔者自定义的面板，读者可在光盘中查看时钟面板的源代码，光盘位置为“光盘\mr\08\src\com\mingrisoft\panel\ClockPanel.java”。

下面详细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。关键代码如下：

例程 05　代码位置：光盘\TM\08\ src\com\mingrisoft\bean\User.java

```
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 User 对象的属性，并含有该属性的 setXXX()与 getXXX()方法。关键代码如下：

例程 06　代码位置：光盘\TM\08\src\com\mingrisoft\util\Session.java

```
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 UserDao 类，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。关键代码如下：

例程 07　代码位置：光盘\TM\08\com\mingrisoft\dao\UserDao.java

```
GetConnection connection = new GetConnection();
Connection conn = null;
//编写按用户名和密码查询用户方法
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的 getUser()方法，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。关键代码如下：

例程 08 代码位置：光盘\TM\08\src\com\mingrisoft\main\Enter.java

```
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
        }
    }
});
```

8.7 主窗体设计

8.7.1 主窗体概述

成功登录系统后，即可进入程序的主窗体，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。主窗体的运行效果如图 8.14 所示。

图 8.14 程序主窗体运行效果

8.7.2 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，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)
```

value：滚动条新的当前值。

成功创建滚动面板后，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。关键代码如下：

例程 09　代码位置：光盘\TM\08\src\com\mingrisoft\widget\SmallScrollPanel.java

```
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。关键代码如下：

例程 10　代码位置：光盘\TM\08\src\com\mingrisoft\widget\SmallScrollPanel.java

```
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，其实为两个添加背景的按钮，将该按钮的边框去掉，即可显示为大家看到的效果。下面以左侧微调按钮为例，介绍微调按钮的实现代码。

例程 11 代码位置：光盘\TM\08\src\com\mingrisoft\widget\SmallScrollPanel.java

```
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。关键代码如下：

例程 12 代码位置：光盘\TM\08\src\com\mingrisoft\widget\SmallScrollPanel.java

```
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 8.15 所示。

图 8.15　平移面板设计效果

8.7.3 主窗体的实现过程

主窗体由多个面板组成，除了前面为大家介绍过的功能按钮面板、时钟面板、日历面板外，还包括功能显示区面板，与主窗体中的其他面板不同，功能显示面板是随时更换的，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 8.16 所示。

图 8.16 主窗体设计效果

说明 主窗体中的日历面板为自定义面板，读者可参考光盘中的源程序，光盘位置为“光盘\mr\08\src\com\mingrisoft\widget\CalendarPanel.java”。

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 8.16 所示的主窗体的设计效果可以看到，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的代码如下：

例程 13 代码位置：光盘\TM\08\src\com\mingrisoft\mainFrame\RemoveButtomFrame.java

```
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 8.16 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 8.17 所示。

图 8.17 按钮组面板设计效果

按钮组面板的实现代码如下：

例程 14 代码位置：光盘\TM\08\src\com\mingrisoft\mainFrame\RemoveButtomFrame.java

```
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;
    }
```

说明　该方法中返回的是 BGPanel 对象，该对象也是一个自定义的面板，读者可参考光盘来看该类的源代码，光盘位置为“光盘\mr\08\src\com\mingrisoft\widget\BGPanel.java”。

（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，介绍平移面板中各按钮的实现代码。

例程 15　代码位置：光盘\TM\08\src\com\mingrisoft\mainFrame\RemoveButtomFrame.java

```
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的图片
        workSpaceButton.setSelected(true);
        workSpaceButton.addActionListener(new toolsButtonActionAdapter());   //按钮的监听器
    }
    return workSpaceButton;
}
```

8.8 采购订货模块设计

8.8.1 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”按钮时，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 8.18 所示。

图 8.18 采购订货模块运行效果

8.8.2 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 8.1 所示。

表 8.1 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，用于指示正在进行编辑
row	要绘制的单元格的行索引。绘制头时，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。关键代码如下：

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

8.8.3　添加采购订货信息的实现过程

当用户单击采购订货窗体中的“添加”按钮时，即可弹出添加采购订货信息窗体，该窗体的运行效果如图 8.19 所示。

图 8.19　添加采购订货信息窗体运行效果

下面详细介绍添加采购订货窗体的实现过程。

（1）创建与采购订货表 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。关键代码如下：

例程 16　代码位置：光盘\TM\08\src\com\mingrisoft\bean\Stock.java

```
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）定义对采购订货表 tb_stock 中数据进行操作的 StockDao 类，其中添加采购订货信息方法 insertStock()，该方法以 Stock 为对象。关键代码如下：

例程 17　代码位置：光盘\TM\08\src\com\mingrisoft\dao\StockDao.java

```
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）在添加采购订货信息窗体的“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。关键代码如下：

例程 18 代码位置：光盘\TM\08\src\com\mingrisoft\archives\InserStockFrame.java

```
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
    }
});
```

8.8.4　搜索采购订货信息的实现过程

在采购订货模块中，添加了按指定条件搜索采购订货信息的功能，用户可按照自己的需求指定条件，搜索采购订货信息窗体的运行效果如图 8.20 所示。

图 8.20　搜索采购订货窗体

下面介绍搜索采购订货信息窗体的具体实现过程。

（1）在搜索采购订货信息窗体中，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，介绍查询数据库的方法。关键代码如下：

例程 19　代码位置：光盘\TM\08\src\com\mingrisoft\dao\Stockdao.java

```
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();                           //定义与数据表对应的 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：

例程 20　代码位置：光盘\TM\08\src\com\mingrisoft\panel\JButtonTablePanel.java

```
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的方法
            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()});
            }
            }
        }
    }
}
```

8.8.5 修改采购订货信息的实现过程

采购订货模块中提供了修改采购订货信息的功能，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货信息窗体，运行效果如图 8.21 所示。

图 8.21　修改采购订货信息窗体

下面详细介绍修改采购订货信息窗体的实现过程。

（1）创建修改采购订货信息窗体的方法 updateStock()，该方法以 Stock 对象作为参数。关键代码如下：

例程 21　代码位置：光盘\TM\08\src\com\mingrisoft\dao\Stockdao.java

```
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，首先要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的方法 selectStockByid()，关键代码如下：

例程 22　代码位置：光盘\TM\08\src\com\mingrisoft\dao\Stockdao.java

```
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;//定义查询的 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）由于显示采购订货信息窗体与修改采购订货信息窗体是两个独立的窗体，用户需要在显示采购订货信息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到文本文件中，之后在修改采购订货信息窗体中读取出来，这样即可实现在修改采购订货信息窗体中显示要修改的订货信息。在显示采购订货信息窗体中，将用户选择的采购订货信息保存在文本文件中。关键代码如下：

例程 23 代码位置：光盘\TM\08\src\com\mingrisoft\panel\JButtonTablePanel.java

```
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）在修改采购订货信息窗体中，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：

例程 24 代码位置：光盘\TM\08\src\com\mingrisoft\archives\UpdateStockFrame.java

```
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）在修改采购订货信息窗体的“修改”按钮中，调用修改采购订货信息的方法，将用户修改的信息保存到数据库中。关键代码如下：

例程 25　代码位置：光盘\TM\08\src\com\mingrisoft\archives\UpdateStockFrame.java

```
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);
    }
});
```

8.8.6　删除采购订货信息的实现过程

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步骤如下：

（1）定义删除数据的 deleteStock()方法，该用户有一个 int 类型参数，用于指定要删除采购订货信息的编号。关键代码如下：

例程 26 代码位置：光盘\TM\08\src\com\mingrisoft\dao\Stockdao.java

```
public void deleteStock(int id){
    conn = connection.getCon();                                    //获取数据库连接
    String sql = "delete from tb_stock where id ="+id;             //定义删除数据的 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要删除的采购订货信息的编号，再调用删除采购订货信息的方法。关键代码如下：

例程 27 代码位置：光盘\TM\08\src\com\mingrisoft\panel\JButtonTablePanel.java

```
JButton deleteButton = new JButton("删除");
deleteButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int row = table.getSelectedRow();                          //获取用户选择的表格的行号
        if (row < 0) {                                             //判断用户选择的行号是否大于 0
            JOptionPane.showMessageDialog(getParent(), "没有选择要删除的数据！",
                    "信息提示框", JOptionPane.INFORMATION_MESSAGE);
            return;                                                //退出程序
        }
        String column = dm.getValueAt(row, 1).toString();          //获取用户选择的行的第一列数据
        dao.deleteStock(Integer.parseInt(column));                 //调用删除数据的方法
        JOptionPane.showMessageDialog(getParent(), "数据删除成功！",
                "信息提示框", JOptionPane.INFORMATION_MESSAGE); //给出提示信息
    }
});
```

8.9 人员管理模块设计

8.9.1 人员管理模块概述

人员管理模块为超市管理员提供了管理超市内部员工的功能，人员管理模块涉及 4 张表，分别为部门表、职务信息表、员工基本信息表、员工详细信息表，人员管理窗体的运行效果如图 8.22 所示。

图 8.22　人员管理窗体运行效果

8.9.2　使用触发器级联删除数据

本模块在保存员工信息时使用了两张表，分别为员工基本信息表与员工详细信息表，这两个表中的数据是一一对应的，如果在员工基本表中删除数据后，对应的员工详细信息表中的数据也应该删除，因此可以通过创建 DELETE 触发器来实现。创建触发器要在数据库中实现，具体语法如下：

```
CREATE TRIGGER trigger_name
ON { table | view }
[ WITH ENCRYPTION ]
{
    { { FOR | AFTER | INSTEAD OF } { [ INSERT ] [ , ] [ UPDATE ] }
        [ WITH APPEND ]
        [ NOT FOR REPLICATION ]
        AS
        [ { IF UPDATE ( column )
            [ { AND | OR } UPDATE ( column ) ]
                [ ...n]
        | IF ( COLUMNS_UPDATED ( ) { bitwise_operator } updated_bitmask )
                { comparison_operator } column_bitmask [ ...n]
      } ]
        sql_statement [ ...n]
    }
}
```

参数说明如表 8.2 所示。

表 8.2　CREATE TRIGGER 函数的参数说明

参　数	说　明
trigger_name	所要创建的触发器的名称
table|view	指创建触发器所在的表或视图，也可以称为触发器表或触发器视图
AFTER	指定触发器只有在完成指定的所有 SQL 语句后才会被触发
AS	触发器要执行的操作
sql_statement	触发器的条件或操作。触发器条件指定其他准则，以确定 DELETE、INSERT 或 UPDATE 语句是否导致执行触发器

例如在本系统中员工基本信息表上创建触发器，实现删除指定的员工信息时，其对应员工详细信息表中的数据也将删除。代码如下：

例程 28　代码位置：光盘\TM\08\SQL\SQLQuery1.sql

```
create trigger triGradeDelete on tb_basicMessage
for delete
 as
   declare @id varchar(10)
   select @id = id from deleted
   delete from tb_contact where tb_contact.id = @id
```

8.9.3　显示查询条件的实现过程

本系统中将查询员工信息的条件以列表的形式给出，其中部门列表中的数据是从部门信息表中查询并显示在窗体中的，当用户选择了要查询员工的部门，系统会将该部门中的所有员工名称都显示在姓名列表中。人员管理模块的查询条件设计效果如图 8.23 所示。

图 8.23　显示查询条件设计效果

下面详细介绍显示查询条件的实现过程。

（1）定义查询部门表中查询所有数据的方法 selectDept()，该方法将查询结果以 List 形式返回。关键代码如下：

例程 29　代码位置：光盘\TM\08\src\com\mingrisoft\dao\DeptDao.java

```
public List selectDept() {
    List list = new ArrayList<Dept>();                          //定义 List 集合对象
    conn = connection.getCon();                                 //获取数据库连接
    try {
        Statement statement = conn.createStatement();           //获取 Statement 方法
        ResultSet rest = statement.executeQuery("select * from tb_dept"); //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
            Dept dept = new Dept();
            dept.setId(rest.getInt(1));                         //应用查询结果设置对象属性
            dept.setdName(rest.getString(2));
            dept.setPrincipal(rest.getString(3));
            dept.setBewrite(rest.getString(4));
```

```
                list.add(dept);                                    //将对象添加到集合中
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
}
```

（2）在人员管理窗体中，调用查询所有部门信息的方法，并将查询出的结果显示在窗体中。关键代码如下：

例程 30　代码位置：光盘\TM\08\src\com\mingrisoft\dao\PersonnelPanel.java

```
List list = dao.selectDept();                                      //调用查询所有部门信息的方法
String dName[] = new String[list.size() + 1];                      //根据查询结果创建字符串数组对象
dName[0] = "";
for (int i = 0; i < list.size(); i++) {                            //循环遍历查询结果集
    Dept dept = (Dept) list.get(i);
    dName[i + 1] = dept.getdName();                                //获取查询结果中部门名称
}
final JComboBox dNamecomboBox = new JComboBox(dName);              //实例化下拉列表对象
```

（3）定义查询指定部门中查询所有员工信息的方法 selectBasicMessageByDept()，该方法将查询结果以 List 形式返回。关键代码如下：

例程 31　代码位置：光盘\TM\08\src\com\mingrisoft\dao\PersonnelDao.java

```
public List selectBasicMessageByDept(int dept) {
    conn = connection.getCon();                                    //获取数据库连接
    List list = new ArrayList<String>();                           //定义保存查询结果的集合对象
    try {
        Statement statement = conn.createStatement();              //实例化 Statement 对象
        String sql = "select name from tb_basicMessage where dept = " + dept +"";//定义按照部门名称查
询员工信息方法
        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
            list.add(rest.getString(1));                           //将查询信息保存到集合中
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;                                                   //返回查询集合
}
```

（4）在“部门”下拉列表框中添加监听事件，实现当用户在“部门”下拉列表框中更改部门名称时，“姓名”下拉列表框中的内容也随之更新。关键代码如下：

例程 32　代码位置：光盘\TM\08\src\com\mingrisoft\panel\PersonnelPanel.java

```
final JComboBox dNamecomboBox = new JComboBox(dName);              //实例化下拉列表对象
    dNamecomboBox.addActionListener((new ActionListener() {        //添加下拉列表监听事件
```

```
            @Override
            public void actionPerformed(ActionEvent e) {
                String dName = dNamecomboBox.getSelectedItem().toString();//获取用户选择的部门名称
                DeptDao deptDao = new DeptDao();                    //定义保存有操作数据库类对象
                int id = deptDao.selectDeptIdByName(dName);         //调用获取部门编号的方法
                List<String> listName = perdao.selectBasicMessageByDept(id);//调用按部门编号查询所有
员工信息的方法
                for (int i = 0; i < listName.size(); i++) {         //循环遍历查询结果集
                    pNameComboBox.addItem(listName.get(i));         //向“姓名”下拉列表中添加元素

                }
                repaint();
            }
});
```

8.9.4 显示员工基本信息的实现过程

当用户选择了要查询的员工，单击“员工信息”列表中的“基本信息”列表项后，系统会将该用户的基本信息显示在窗体中，运行效果如图 8.24 所示。

图 8.24 显示员工信息

下面介绍具体的实现过程。

（1）本模块的员工基本信息是通过员工部门信息与员工姓名查询出来的。首先编写按照部门名称和员工信息查询员工基本信息的方法。关键代码如下：

例程 33 代码位置：光盘\TM\08\src\com\mingrisoft\dao\PersonnelDao.java

```
public BasicMessage selectBNameById(String dept,String name) {
    conn = connection.getCon();                                     //获取数据库连接
    BasicMessage message = new BasicMessage();                      //创建与数据表对应的 JavaBean 对象
    try {
        Statement statement = conn.createStatement();
        String sql = "select * from tb_basicMessage where name = '"+name+"' and dept = (select id
from tb_dept" + " where dName = '"+dept+"')";                       //定义查询数据的 SQL 语句
        ResultSet rest = statement.executeQuery(sql);               //执行查询语句
        while (rest.next()) {                                       //循环遍历查询结果集
            message.setId(rest.getInt(1));                          //应用查询结果设置对象属性
            message.setName(rest.getString(2));
            message.setAge(rest.getInt(3));
            message.setSex(rest.getString(4));
            message.setDept(rest.getInt(5));
            message.setHeadship(rest.getInt(6));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
```

```
        return message;
}
```

（2）在显示用户基本信息窗体时，首先判断用户是否选择了合法的查询条件，再将用户选择要查询的员工的基本信息显示在窗体中。关键代码如下：

例程 34　代码位置：光盘\TM\08\src\com\mingrisoft\panel\PersonnelPanel.java

```
jlist.addListSelectionListener(new ListSelectionListener() {
    public void valueChanged(ListSelectionEvent e) {
        if (!e.getValueIsAdjusting()) {
            String deptName = dNamecomboBox.getSelectedItem().toString();//判断用户选择的查询条件
            if(deptName.equals("")){
                JOptionPane.showMessageDialog(getParent(), "没有选择查询的员工！",
                        "信息提示框", JOptionPane.INFORMATION_MESSAGE); //提示信息
                return;
            }
            JList list = (JList) e.getSource();                          //获取事件源
            String value = (String) list.getSelectedValue();             //获取列表选项并转换为字符串
            if(value.equals("基本信息")){                                 //判断用户是否选择了“基本信息”
                String name = pNameComboBox.getSelectedItem().toString(); //获取用户选择的员工姓名
                panel_1.remove(particular);                              //移除显示员工详细信息的面板
                panel_1.add(bpanel);
                bpanel.setBounds(140, 53, 409, 208);
                message = perdao.selectBNameById(deptName,name);         //调用查询数据方法
                pId = message.getId();
                nameTextField.setText(message.getName());        //设置员工基本系统中的组件内容
                ageTextField.setText(message.getAge()+"");
                String sex = message.getSex();
                if(sex.equals("男")){
                    manRadioButton.setSelected(true);        //设置窗体中“性别”单选按钮的显示内容
                }
                else{
                    wradioButton.setSelected(true);
                }
                int dept = message.getDept();
                Dept depts = dao.selectDepotById(dept);        //按照部门编号查询员工所在部门的信息
                deptField.setText(depts.getdName());                 //设置员工部门内容
                String hName = perdao.selectHeadshipById(message.getHeadship());
                headshipField.setText(hName);
                repaint();
            }
        }
});
```

8.9.5　添加员工信息的实现过程

当用户单击如图 8.22 所示的人员管理窗体的“添加”按钮后，将弹出添加员工信息窗体，添加员

工信息由两部分组成，分别为添加员工基本信息与添加员工联系资料，添加员工窗体使用了选项卡面板，添加员工信息窗体的运行效果如图 8.25 所示。

图 8.25　添加员工信息窗体

下面介绍具体的实现过程。

（1）定义向员工基本信息表中添加数据的方法 insertBasicMessage()，该方法将与员工基本表对应的 JavaBean 对象 BasicMessage 作为参数。关键代码如下：

例程 35　代码位置：光盘\TM\08\src\com\mingrisoft\dao\PersonnelDao.java

```
public void insertBasicMessage(BasicMessage message) {
    conn = connection.getCon();                                    //获取数据库连接
    try {
        PreparedStatement statement = conn
                .prepareStatement("insert into tb_basicMessage values(?,?,?,?,?)");//定义添加数据的
SQL 语句
        statement.setString(1,message.getName());                  //设置预处理语句参数值
        statement.setInt(2, message.getAge());
        statement.setString(3, message.getSex());
        statement.setInt(4, message.getDept());
        statement.setInt(5, message.getHeadship());
        statement.executeUpdate();                                 //执行插入语句
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

（2）定义向员工联系人表中添加数据的方法 insertContact()。关键代码如下：

例程 36　代码位置：光盘\TM\08\src\com\mingrisoft\dao\PersonnelDao.java

```
public void insertContact(Contact contact) {
    conn = connection.getCon();                                    //获取数据库连接
    try {
        PreparedStatement statement = conn
                .prepareStatement("insert into tb_contact values(?,?,?,?,?,?)");//定义插入数据的 SQL 语句
        statement.setInt(1, contact.getHid());                     //设置插入语句参数
        statement.setString(2, contact.getContact());
        statement.setString(3, contact.getOfficePhone());
        statement.setString(4, contact.getFax());
        statement.setString(5, contact.getEmail());
        statement.setString(6, contact.getFaddress());
        statement.executeUpdate();                                 //执行插入语句
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

（3）由于在添加员工信息窗体中，部门名称以文字的形式显示给用户，而员工基本信息表中的部门字段存储的是部门表中的部门编号，因此要定义按部门名称查询部门编号的方法 selectDeptByName()。关键代码如下：

例程 37　代码位置：光盘\TM\08\src\com\mingrisoft\dao\DeptDao.java

```
public Dept selectDeptByName(String name) {
    conn = connection.getCon();                                    //获取数据库连接
    Dept dept = null;
    try {
        Statement statement = conn.createStatement();              //实例化 Statement 对象
        String sql = "select * from tb_dept where dName = '" + name +"'";//定义按部门名称查询部门信息的
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
            dept = new Dept();                                     //定义与部门表对应的 JavaBean 对象
            dept.setId(rest.getInt(1));                            //应用查询结果设置对象属性
            dept.setdName(rest.getString(2));
            dept.setPrincipal(rest.getString(3));
            dept.setBewrite(rest.getString(4));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return dept;                                                   //返回 JavaBean 对象
}
```

（4）定义按照职位名称查询职务编号的方法 selectIdByHeadship()，该方法以表示 String 对象为参数，将查询结果以 int 形式返回。关键代码如下：

例程 38　代码位置：光盘\TM\08\src\com\mingrisoft\dao\PersonnelDao.java

```
public int selectIdByHeadship(String hName) {
    int id = 0;                                                    //定义保存查询结果的 int 对象
    conn = connection.getCon();                                    //获取数据库连接
    try {
        Statement statement = conn.createStatement();              //定义 Statement 对象
        String sql = "select id from tb_headship where headshipName = '" + hName+"'";//定义执行查询的
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
            id = rest.getInt(1);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return id;
}
```

（5）在添加员工信息窗体中，要求用户输入年龄信息的文本框只允许用户输入数字，可通过在年龄文本框中添加键盘监听事件，实现当用户输入字母时不显示在文本框中。关键代码如下：

例程 39 代码位置：光盘\TM\08\src\com\mingrisoft\archives\InsertPersonnelFrame.java

```
ageTextField = new JTextField();
ageTextField.addKeyListener(new KeyAdapter() {
    public void keyTyped(KeyEvent event) {                    //某键按下时调用的方法
        char ch = event.getKeyChar();                         //获取用户输入的字符
        if((ch<'0'||ch>'9')){                                 //如果用户输入的信息不为数字
            event.consume();                                  //不允许用户输入
        }
    }
});
```

（6）在“添加”按钮的监听事件中，首先判断用户是否输入合法的信息，如果输入合法，实现将用户添加的信息保存到数据库中。关键代码如下：

例程 40 代码位置：光盘\TM\08\src\com\mingrisoft\archives\InsertPersonnelFrame.java

```
JButton insertutton = new JButton("添加");
insertutton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String name = nameTextField.getText();                          //获取用户添加的姓名信息
        String age = ageTextField.getText();                            //获取用户添加的年龄信息
        String dept = deptComboBox.getSelectedItem().toString();        //获取用户添加的部门信息
        String headship = headshipComboBox.getSelectedItem().toString(); //获取用户添加的职务信息
        int id = dao.selectIdByHeadship(headship);                      //调用根据职务名称查询职务编号的方法
        if((name.equals(""))||(age.equals(""))){                        //判断用户添加的信息是否为空
            JOptionPane.showMessageDialog(getContentPane(), "将带星号的信息填写完整！",
                    "信息提示框", JOptionPane.INFORMATION_MESSAGE); //给出提示信息
            return;                                                     //退出程序
        }
        int ageid = Integer.parseInt(age);                              //将用户添加的年龄信息转换为整型数据
        DeptDao deptDao = new DeptDao();                                //创建保存操作部门表数据的方法
        Dept dpt = deptDao.selectDeptByName(dept);                      //调用根据部门名称查询部门编号的方法
        message.setName(name);                                          //设置 JavaBean 对象名称属性
        message.setAge(ageid);
        message.setDept(dpt.getId());
        message.setHeadship(id);
        dao.insertBasicMessage(message);                                //调用向员工信息表中添加数据的方法
        JOptionPane.showMessageDialog(getContentPane(), "将信息添加成功！",
                "信息提示框", JOptionPane.INFORMATION_MESSAGE);
    }
});
```

8.9.6 删除员工信息的实现过程

当用户单击如图 8.22 所示的显示员工信息窗体中的“删除”按钮后，系统会将用户选择的员工删除。由于笔者在数据库中创建了触发器，因此当用户实现将员工信息表中的数据删除时，员工联系人信息表中对应的数据也会被删除。

删除员工显示信息的具体实现过程如下：

（1）定义删除员工信息的方法 deleteBasicMessage()，该方法以员工编号作为参数。关键代码如下：

例程 41　代码位置：光盘\TM\08\src\com\mingrisoft\dao\PersonnelDao.java

```
public void deleteBasicMessage(int id){
    conn = connection.getCon();                                    //调用获取数据库连接的方法
    String sql = "delete from tb_basicMessage where id ="+id;      //定义删除数据的 SQL 语句
    try {
        Statement statement = conn.createStatement();              //定义 Statement 方法
        statement.executeUpdate(sql);                              //执行删除数据的 SQL 语句
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

（2）在“删除”按钮的单击事件中，实现删除员工基本信息，此时触发器 trigger_name 会自动执行，将对应员工的详细信息也删除。关键代码如下：

例程 42　代码位置：光盘\TM\08\src\com\mingrisoft\panel\PersonnelPanel.java

```
JButton deleteButton = new JButton("删除");
deleteButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int n = JOptionPane.showConfirmDialog(getParent(),
                "确认正确吗？","确认对话框", JOptionPane.YES_NO_CANCEL_OPTION);
        if(n == JOptionPane.YES_OPTION){                           //如果用户确认信息
            perdao.deleteBasicMessage(pId);                        //调用删除数据的方法
        }
    }
});
```

8.10　在 Eclipse 中实现程序打包

完成了简易滕宇超市管理系统的开发后，接下来的工作就是对该系统进行打包并交付用户使用，下面就以在 Eclipse 中将应用程序打包成 JAR 文件为例，讲解应用程序的打包过程。

将超市管理系统打包成 JAR 文件的步骤如下：

（1）编写 JAR 的清单文件，在清单文件中完成 JAR 文件的配置，例如闪屏界面、主类名称、类路径等。在 Eclipse 的“包资源管理器”视图中的“超市管理系统”节点上单击鼠标右键，在弹出的快捷菜单中选择“新建”/“文件”命令，打开“新建文件”对话框，如图 8.26 所示，在“文件名”文本框中输入“MANIFEST.MF”，单击“完成”按钮，完成 MANIFEST.MF 文件的建立。

（2）双击项目节点中的 MANIFEST.MF 文件，在打开 MANIFEST.MF 文件的编辑器中输入如下代码：

```
Manifest-Version: 1.0
SplashScreen-Image: res/sys_splash.jpg
Main-Class: com.mingrisoft.main.Enter
Class-Path: . lib/sqljdbc.jar
```

上面代码的第一行的 Manifest-Version 用于指定清单文件的版本。第二行的 SplashScreen-Image 用于指定闪屏界面所使用的图片资源，这里设置为 res/sys_splash.jpg，表示使用的是 res 包中的 sys_splash.jpg 文件。第 3 行 Main-Class 用于定义 JAR 文件中的主类，这里设置为 zzk.zhuoyue.main.Main。第 4 行 Class-Path 用于设置程序执行时的类路径、运行程序所需的第三方类库，本应用程序使用的是 MySQL 数据库，所以需要把 MySQL 数据库的驱动类添加到该类路径中。

注意 代码中的“:”必须要有一个空格字符做分隔符。Class-Path 中的不同类库要使用空格分隔，并且在清单文件的最后一行要有一个空行。

（3）保存 MANIFEST.MF 文件，在“包资源管理器”视图的“超市管理系统”节点上单击鼠标右键，在弹出的快捷菜单中选择“导出”命令，将打开“导出”向导对话框，如图 8.27 所示。单击 Java/“JAR 文件”节点，然后单击“下一步”按钮。

图 8.26 “新建文件”对话框

图 8.27 “导出”向导对话框

（4）在打开的“JAR 导出”对话框的“JAR 文件”文本框中输入要生成的 JAR 文件的存放路径和文件名，这里输入“D:\超市管理系统\滕宇超市管理系统.jar”，如图 8.28 所示。单击“下一步”按钮。

（5）在弹出的“JAR 导出”对话框中选中“导出带有编译错误的类文件”和“导出带有编译警告的类文件”复选框，如图 8.29 所示。因为类文件的编译警告信息不一定会导致程序无法运行，甚至有的警告信息并不影响项目要实现的业务逻辑。单击“下一步”按钮。

（6）在“JAR 导出”对话框的“JAR 清单规范”页中选中“从工作空间中使用现有清单”单选按钮，单击“清单文件”文本框右侧的“浏览”按钮，从打开的“选择清单”对话框中选择“腾宇超市管理系统”节点中的 MANIFEST.MF 清单文件，单击“确定”按钮。再单击“完成”按钮完成清单文件的选择，如图 8.30 所示。

（7）打开“我的电脑”，双击 D 盘中的“进销存”文件夹。在该文件夹中创建 lib 文件夹和 lib 文件夹的子文件夹“sqljdbc.jar”，如图 8.31 所示。把 MySQL 数据库的 JDBC 驱动类的 JAR 文件复制到 lib

文件夹中。如果客户端的 Java 环境安装正确，双击“超市管理系统.jar”文件即可运行程序。

图 8.28　“JAR 导出”对话框

图 8.29　选中复选框

图 8.30　选择清单文件

图 8.31　jar 文件的存放地址

8.11　本 章 小 结

本章介绍的是一个实用项目超市管理系统开发中应用的关键技术和开发技巧，这些技巧在开发中都是很关键的。例如如何安排项目的包资源结构、合理地设计窗体布局、面板的灵活使用、在表格中添加特殊内容、合理使用数据库对象等，最后还介绍了如何在 Eclipse 中实现程序打包。

第 9 章

企业门户网站

（JSP+JavaBean+SQL Server 2000 实现）

随着计算机与电子技术的飞速发展以及网络越来越广泛，国内外很多大中小企业都意识到网络信息传递带给企业的效益是任何其他传递方式不可比拟的，网络在某种程度上可以大大提高员工的办事效率、提升整个企业的竞争力，所以很多企业选择通过互联网为企业做宣传、树立企业良好形象以及提高企业知名度等。本章通过一个小型企业门户网站介绍如何使用 JSP+JavaBean+SQL Server 2000 快速开发一个企业门户网站。

通过阅读本章，可以学习到：

- 如何进行网站的需求分析
- 如何进行系统设计
- 如何进行数据库分析与数据库建模
- 如何配置 Tomcat 连接池
- 如何在 JSP 中创建过滤器
- 如何使用 Proxool 连接池

9.1　开 发 背 景

××公司是一家以经营电子产品为主的小型企业，大多数中小企业都不愿意花费巨额的经费去做广告宣传，××公司也不例外，企业领导人深知网络宣传具有低投入、高回报的特点，所以委托笔者制作一个企业门户网站，意在为自己的企业进行网络宣传。这里所谓的网络宣传并不仅是简单的网站展示建设，或通过网络媒介做一些广告宣传，它还包括利用网络在企业之间、企业内部以及企业和用户之间传递信息，以达到用户更深入地了解企业及企业商品的目的。

9.2　需 求 分 析

成功的企业门户网站需要一个高质量的前台页面和可以提升企业信息延续性和扩展性的后台管理系统。这里所说的高质量的前台页面不仅具有美观、动态的特点，它还需要具有网站信息传输的高效性、安全性、可靠性等优势，并确保网站中商业信息不被丢失。为了实现网站功能具有较高的延续性和可扩展性，使网站的建设紧跟企业发展的需求，就需要一个网站后台管理系统。同时考虑到企业所能承担的成本，决定使用 JSP+JavaBean 开发模式，这种模式更加适合中小型项目的开发。

通过实际调查，要求企业门户网站具有以下功能：

- ☑ 门户网站前台页面设计美观、大方，凸显企业商品、新闻、文化信息等。
- ☑ 门户网站后台页面简洁，应具有企业新闻、商品、用户管理等功能模块。
- ☑ 前后台设计明确，并保证前后台的安全性。
- ☑ 充分考虑架设网站平台时节约企业的成本，应用 JSP+JavaBean+SQL Server 2000 开发模式。

9.3　系 统 设 计

9.3.1　系统目标

开发企业门户网站的最终目的是为企业提供一个简单、易用、开放、可扩展的企业信息门户平台。通过需求分析以及与客户的沟通，现制定网站实现目标如下：

- ☑ 网站使用人性化设计，界面友好、安全、实用。
- ☑ 网站操作便捷并具有高度信息延续性、可扩展性。
- ☑ 提供建立在关系型数据库系统上的数字信息组织、管理、查询等功能。
- ☑ 对用户输入的数据进行严格的数据检索，尽可能地排除人为错误。
- ☑ 最大限度地实现网站易维护性和易操作性。

9.3.2 系统功能结构

根据企业门户网站的特点，可以将网站分为前、后台两个部分。前台部分主要实现企业与客户交互，后台部分主要实现网站相关信息管理功能。

☑ 网站前台

网站前台部分主要包括企业新闻展示、产品信息介绍、公司文化、技术支持、管理员登录等功能模块。

网站前台功能结构如图 9.1 所示。

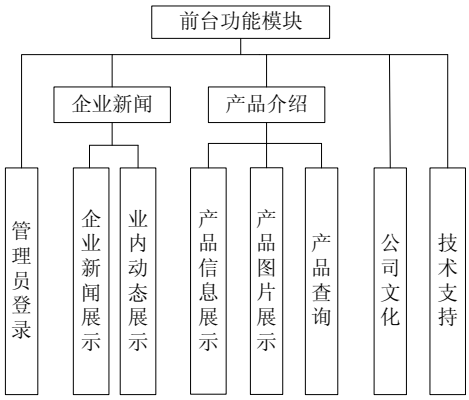


图 9.1 网站前台功能结构

☑ 网站后台

网站后台部分主要包括企业新闻管理、商品管理、管理员注销等功能模块。

网站后台功能结构如图 9.2 所示。

9.3.3 业务流程图

企业门户网站业务流程如图 9.3 所示。

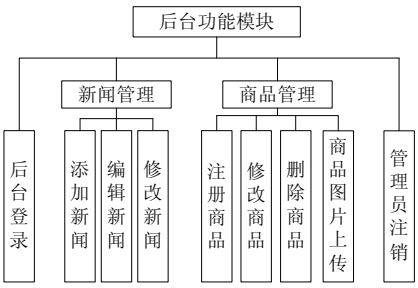


图 9.2 网站后台功能结构图

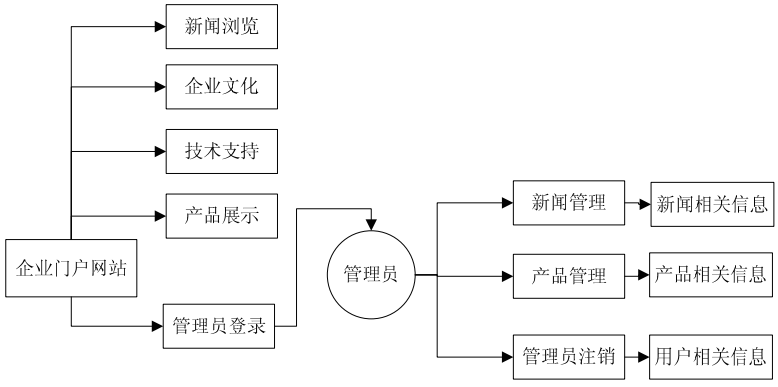


图 9.3 企业门户网站业务流程

9.3.4 系统预览

企业门户网站由多个页面组成，下面列出几个典型页面，其他页面参见光盘中的源程序。

企业新闻页面如图 9.4 所示，该页面用于实现企业新闻展示等功能。公司文化展示页面如图 9.5 所示，主要用于展示企业文化。

技术支持页面如图 9.6 所示，该页面用于向用户提供联系方式等功能。产品介绍页面如图 9.7 所示，主要用于实现商品添加、商品删除、商品图片上传等功能。

图 9.4　新闻页面（光盘\TM\09\net\WebRoot\qyxw.jsp）

图 9.5　文化页面（光盘\TM\09\net\WebRoot\qywh.jsp）

图 9.6　技术支持页面（光盘\TM\09\net\WebRoot\jszc.jsp）

图 9.7　产品介绍页面（光盘\TM\09\net\WebRoot\cpjs.jsp）

9.3.5　构建开发环境

1．MyEclipse 中配置 Tomcat 服务器

MyEclipse 提供了与各种 Java Web 服务器的连接方式，经过设置后，可以直接在 MyEclipse 中启动 Tomcat 服务器，这样便可以在 MyEclipse 中控制服务器的启动和停止。本系统的运行环境采用的是 Tomcat 5.5，具体配置步骤读者可以参照第 7 章相关部分的内容。

2．配置 Tomcat 连接池

创建 Tomcat 连接池的目的是在 Tomcat 启动时可以产生足够多的数据库连接，并提供给程序使用。通过使用连接池，可以提高程序的运行速度，同时也节省内存，提高服务器的效率，能够支持更多的

用户连接。连接的建立、断开都由连接池自身来管理，当程序需要建立数据库连接时，只需从内存中取一个来用而不用新建。同样，使用完毕后，将其放回内存即可。另外，连接池还可以通过其自身的管理机制来监视数据库连接的数量、使用情况等。

配置 Tomcat 连接池的步骤如下：

（1）打开 Tomcat 所在目录下的 conf\server.xml 文件，在文件中最后一个</host>前面增加如下代码：

```
<Context path="/net" docBase="net" debug="5" reloadable="true" corssContext="true">
<Resource name="jdbc/ConnectionPool"
      auth="Container"
❶     type="javax.sql.DataSource"
❷     maxActive="20"
❸     maxIdle="5"
❹     maxWait="10000"
❺     username="sa"
❻     password=""
❼     driverClassName="com.microsoft.jdbc.sqlserver.SQLServerDriver"
❽     url="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_net"/>
</Context>
<parameter>
<name>factory</name>
<value>org.apache.commons.dbcp.BasicDataSourceFactory</value>       <!--使用 DBCP 连接池-->
</parameter>
```

代码贴士

❶ type：类型为 javax.sql.DataSource。
❷ maxActive：连接池处于活动状态时的最大连接数。
❸ maxIdle：连接池处于空闲状态时的最小连接数。
❹ maxWait：一个连接最长的空闲时间，单位为 ms。
❺ username：数据库连接用户名。
❻ password：数据库连接密码。
❼ driverClassName：数据库连接的驱动包名。
❽ url：数据库连接的 URL。

（2）将 SQL Server 驱动包复制到 Tomcat 路径下的 common\lib 目录中。

（3）在项目目录下的 WebRoot\WEB-INF\web.xml 文件中添加如下代码：

```
<resource-ref>
    <description>SQL server text app</description>
    <res-ref-name>jdbc/ConnectionPool</res-ref-name>                    //指定连接池名称
    <res-type>javax.sql.DataSource</res-type>                           //指定连接池类型
    <res-auth>Container</res-auth>
</resource-ref>
```

（4）至此，Tomcat 连接池配置完毕，重启 Tomcat 服务器，即可在 JSP 或 JavaBean 文件中使用 Tomcat 数据源。例如：

```
Context initCtx = new javax.naming.InitialContext();                    //初始化 Context 对象
Context envCtx = (Context) initCtx.lookup("java:comp/env");
ds = (DataSource)envCtx.lookup("jdbc/ConnectionPool");                  //取得数据源
conn=ds.getConnection();                                                //获取数据库连接
```

3．构建页面风格

网站的前台页面是网站建设中不可忽略的，美观的网页除了美工的设计外，还需要定义一个良好的 CSS 样式文件，命名为 style.css。关键代码如下：

```
.zi {                                                   //定义名为 zi 的样式，字体大小为 12px，颜色为#FF6501
      font-size: 12px;
      color: #FF6501;
}
input{                                                  //定义网页中文本框的样式
      font-size: 9pt;                                   //字体大小为 9pt
      color: #333333;                                   //颜色为#333333
      border: 1px solid #999999;                        //边框颜色、粗细
}
a:hover {                                               //定义鼠标经过链接文字时的样式
      font-size: 9pt;   color: #FF0000;                 //定义字体与颜色
}
a {                                                     //定义链接文字的样式
      font-size: 9pt;   text-decoration: none;      color: #FFC000;     //定义链接文字字体与颜色
}
```

4．错误处理页面

当 JSP 页面发生异常时，需要创建一些错误处理页面。在本网站建设中，创建一个错误页面，命名为 error.jsp。关键代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
❶          pageEncoding="UTF-8" isErrorPage="true"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<!--省略部分代码-->
<body>
程序中发生了以下的错误：
❷    <%=exception.getMessage()%>
</body>
</html>
```

代码贴士

❶ isErrorPage="true"：当使用 exception 对象时，必须在 page 指令中指定 isErrorPage="true"。

❷ exception.getMessage()：打印错误信息。

当 JSP 文件需要使用错误页面时，需要使用以下代码：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"   errorPage="../error.jsp"%>                   //在 JSP 页面中引用错误页面
```

当一个使用错误页面的 JSP 文件出现错误时，系统会自动调用 error.jsp 页面。

9.3.6 文件夹组织结构

在开发项目前，将可能用到的文件夹创建出来，可以方便以后的开发工作，还可以规范网站的整体架构。本网站的文件夹组织结构如图 9.8 所示。

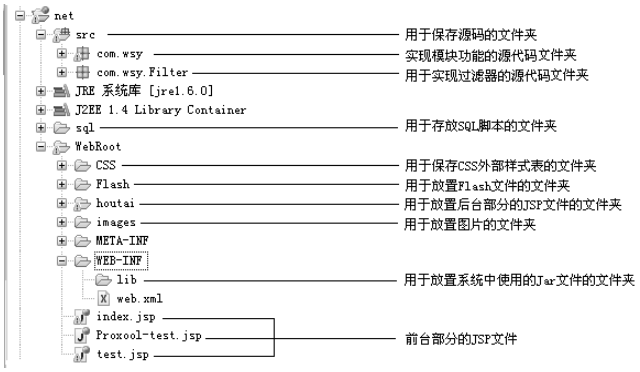


图 9.8　企业门户网站文件夹组织结构

9.4 数据库设计

9.4.1 数据库需求分析

企业门户网站的数据库访问量是比较大的，开发企业门户网站使用的数据库不仅应能承载巨大的数据量，而且还需要具有强大的稳定性和可靠性。考虑到节约网站开发成本，笔者决定使用 SQL Server 2000 数据库。

SQL Server 2000 是一种客户/服务器模式的关系型数据库。它具有很强的数据完整性、可伸缩性、可管理性、可编程性；具有均衡与完备的功能；性价比较高。SQL Server 2000 数据库提供了复制服务、数据转换服务、报表服务，并支持 XML 语言。使用 SQL Server 2000 数据库可以大容量地存储数据，并对数据进行合理的逻辑布局，应用数据库对象可以对数据进行复杂的操作。SQL Server 2000 是在 SQL Server 7.0 的基础上扩展升级而来，它继承了 SQL Server 以前版本的优点，同时又增加了许多新功能，与微软公司的其他产品具有良好的兼容性。

9.4.2 数据库概念设计

通过对系统进行的需求分析、系统流程设计以及系统功能结构的确定，规划出本系统中使用的主要数据库实体对象分别为新闻实体、商品实体、商品类别实体、用户实体。其中商品实体与商品类别实体需要以外键进行联系。

☑ 新闻实体对象

新闻实体对象包括新闻标题、新闻内容、新闻作者、提交时间及新闻编号等属性。这几个属性均

为新闻实体的基本信息，其中新闻编号为新闻实体对象的唯一标识，设置为自动增长类型。新闻实体 E-R 图如图 9.9 所示。

☑　商品实体对象

商品实体对象包括商品编号、商品名称、商品样图、商品描述、商品类别、商品提交时间等属性。其中，商品编号为商品实体对象的唯一标识，设置为自动增长类型；商品样图存储商品样图的文件名称；商品提交时间属性设置为 datetime 类型。商品实体 E-R 图如图 9.10 所示。

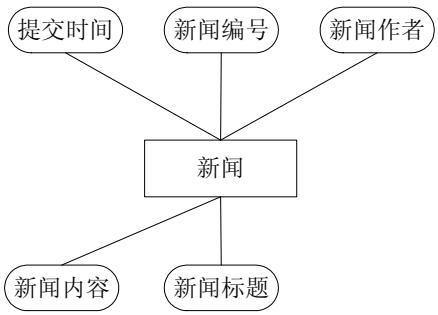


图 9.9　新闻实体 E-R 图

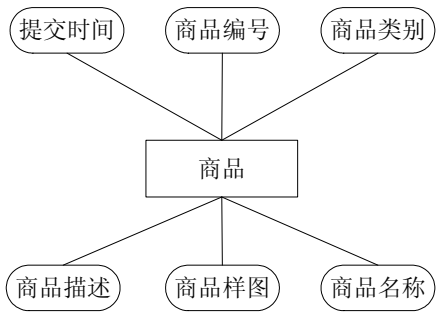


图 9.10　商品实体 E-R 图

☑　商品类别实体对象

商品类别实体对象包括商品类别名称、商品类别编号、提交时间等属性。其中商品类别编号属性为商品类别实体对象的唯一标识，设置为自动增长类型。商品类别实体 E-R 图如图 9.11 所示。

☑　用户实体对象

用户实体对象包括用户名称、用户编号、用户密码等属性。其中用户编号属性为用户实体对象的唯一标识，设置为自动增长类型。用户实体 E-R 图如图 9.12 所示。

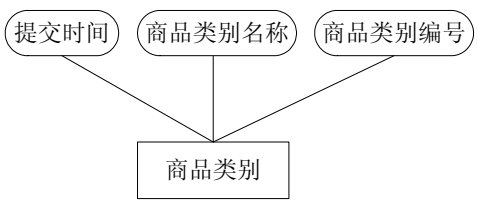


图 9.11　商品类别实体 E-R 图

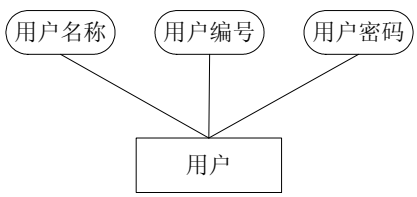


图 9.12　用户实体 E-R 图

9.4.3　数据库逻辑结构设计

根据在数据库概念设计中给出的数据库实体 E-R 图，可以设计数据表结构。本网站包括以下数据表：

☑　tb_business（企业商品表）

企业商品表主要用于存放企业商品信息。tb_business 表结构如图 9.13 所示。

	列名	数据类型	长度	允许空	描述
🔑	id	int	4		商品ID
	name	varchar	50	✓	商品名称
	img	varchar	50	✓	商品样图
	ms	varchar	100	✓	商品描述
	category	varchar	20	✓	商品类别
	submittime	datetime	8	✓	商品提交时间
▶					

图 9.13　商品表结构

☑ tb_usertable（用户表）

用户表主要保存用户的相关信息。tb_usertable 表结构如图 9.14 所示。

	列名	数据类型	长度	允许空	描述
🔑	id	int	4		用户编号
	name	varchar	50	✓	用户名
	password	varchar	50	✓	用户密码

图 9.14　用户表结构

☑ tb_category（商品类别表）

商品类别表主要用于保存商品类别的相关信息。tb_category 表结构如图 9.15 所示。

	列名	数据类型	长度	允许空	描述
🔑	id	int	4		商品类别编号
	categoryname	varchar	50	✓	商品类别名称
	submittime	datetime	8	✓	提交时间

图 9.15　商品类别表结构

☑ tb_news（企业新闻表）

企业新闻表主要用于存放企业新闻的相关信息。tb_news 表结构如图 9.16 所示。

	列名	数据类型	长度	允许空	描述
🔑	id	int	4		新闻编号
	title	varchar	50	✓	新闻名称
	content	varchar	100	✓	新闻内容
	author	varchar	20	✓	新闻作者
	submittime	datetime	8	✓	提交时间

图 9.16　新闻表结构

9.5　公共模块设计

9.5.1　定义 connsqlserver 类

在 9.3.5 节中已经介绍了如何配置 Tomcat 连接池，配置连接池后，Tomcat 就会将这个数据源绑定到 JNDI 命名空间，可以通过(DataSource)envCtx.look("jdbc/ConnectionPool")来获取这个数据源，使用 getConnection()方法获取数据库连接。有了连接池的数据源连接，不仅可以提高访问数据库的效率，而且使操作数据库变得更为简单。为了使连接数据库的代码高度重用，在这里笔者将数据库连接操作封装到 JavaBean 中，命名为 connsqlserver.java，作为公用类使用。创建 connsqlserver.java 文件的步骤如下：

（1）创建数据库连接方法。关键代码如下：

例程 01　代码位置：光盘\TM\09\net\src\com\wsy\connsqlserver.java

```
private void getConnection(){
    if (cn!=null) {
        //如果连接不为空，不新建连接
        return;                                                    //返回，不执行以下代码
    }
    Context ctx;
    try{
    ctx =new InitialContext();
```

```
        DataSource ds=(DataSource)ctx.lookup("java:comp/env/jdbc/ConnectionPool"); //获取数据源
        cn=ds.getConnection();                                                     //获取数据库连接
        }catch (NamingException e){
            e.printStackTrace();                                                   //捕捉异常
        }catch (SQLException e){
            e.printStackTrace();                                                   //捕捉异常
        }
            return;                                                                //返回
}
```

（2）connsqlserver.java 文件中除了设置数据库连接方法外，还要设置数据库查询方法。关键代码如下：

例程 02　代码位置：光盘\TM\09\net\src\com\wsy\connsqlserver.java

```
public ResultSet executeQuery(String sql) {
        if (cn == null)                                                            //如果连接为空
            //如果数据库连接为空，新建数据库连接
            getConnection();
        try {
            return cn.createStatement().executeQuery(sql);                         //执行 SQL 语句
        } catch (SQLException e) {
            e.printStackTrace();                                                   //捕捉异常
            return null;                                                           //在异常中返回空值
        } finally {
        }
}
```

（3）创建数据表更新方法。关键代码如下：

例程 03　代码位置：光盘\TM\09\net\src\com\wsy\connsqlserver.java

```
public int executeUpdate(String sql) {
        if (cn == null)
            //如果数据库连接为空，新建数据库连接
            getConnection();
        try {
            return cn.createStatement().executeUpdate(sql);                        //执行 SQL 语句
        } catch (SQLException e) {
            e.printStackTrace();                                                   //捕捉异常
            return -1;                                                             //在异常中返回-1
        } finally {
        }
}
```

（4）创建数据库连接关闭方法。尽管程序开发使用了连接池这种高效的数据库连接方式，但如果一个数据库连接不被关闭，还是很容易使数据库连接枯竭，抛出异常，实际上在这里使用的数据库关闭方法不是真正地销毁一个数据库连接，而是将数据库连接返回到连接池中。数据库连接关闭的关键代码如下：

例程 04 代码位置：光盘\TM\09\net\src\com\wsy\connsqlserver.java

```
public void close() {
    try {
        cn.close();                                          //关闭数据库连接
    } catch (SQLException e) {
        e.printStackTrace();                                 //捕捉异常
    }finally{
        cn = null;                                           //最终将连接置空
    }
}
```

9.5.2 创建 Web 应用过滤器

Web 应用中过滤器可以获取客户端的请求，并对请求做相应的处理，例如可以验证用户是否来自可信网络，对用户提交的数据进行重新编码等。在实际的开发中可以对 Web 应用组件配置多个过滤器，每个过滤器执行不同的功能。创建 Web 应用过滤器的步骤如下：

（1）在“项目名称\WebRoot\WEB-INF”路径下找到 web.xml 文件，将以下配置过滤器代码添加到 web.xml 文件中的<web_app></web_app>标签之间。

例程 05 代码位置：光盘\TM\09\net\WebRoot\WEB-INF\web.xml

```
<filter>
<filter-name>modifycode</filter-name>                    <!--过滤器名称-->
<filter-class>com.wsy.Filter.ModifyCode</filter-class>   <!--配置过滤器类所在的位置-->
<init-param>
    <param-name>code</param-name>                <!--在过滤器中配置参数名称-->
    <param-value>UTF-8</param-value>             <!--赋予参数 code 值-->
</init-param>
</filter>
<filter-mapping>
<filter-name>modifycode</filter-name>            <!--过滤器名称，与<filter-name>标签中的配置相同-->
<url-pattern>/*</url-pattern>                    <!--过滤器对应的 URL，表示所有网页都会使用到过滤器-->
<dispatcher>REQUEST</dispatcher>                 <!--当用户提出请求动作时，才会通过此过滤器-->
<dispatcher>FORWARD</dispatcher>                 <!--当用户发出转发动作时，才会通过此过滤器-->
<dispatcher>INCLUDE</dispatcher>                 <!--当用户发出包含文件动作时，才会通过此过滤器-->
<dispatcher>ERROR</dispatcher>                   <!--当用户使用错误机制时，才会通过此过滤器-->

</filter-mapping>
```

（2）创建字符编码过滤器。如果开发一个过滤器，必须实现 Filter 接口。Filter 接口定义如下方法：

☑ init()方法

当一个过滤器被加载时，首先执行 init()方法，一般在这里做初始化操作。

☑ doFilter(ServletRequest,ServletReponse,FilterChain chain)方法

这个方法有 3 个参数，前两个参数是 request、response 对象，最后一个参数是 FilterChain 对象，它使用 doFilter()方法将 request、response 对象传递到下一个过滤器。

☑ destroy()方法

销毁过滤器方法。

创建过滤器类 com.wsy.Filter.ModifyCode.java。关键代码如下：

例程 06　代码位置：光盘\TM\09\net\src\com\wsy\Filter\ModifyCode.java

```
public class ModifyCode implements Filter{
    protected FilterConfig filterConfig;
    private String targetEncoding="UTF-8";
    public void init(FilterConfig config)throws ServletException{          //init()方法
        this.filterConfig=config;
❶       this.targetEncoding=config.getInitParameter("code");              //获取 web.xml 中参数 code 的值
    }
    public void doFilter(ServletRequest request,ServletResponse response,FilterChain chain)throws Servlet
Exception{
        HttpServletRequest srequest=(HttpServletRequest)request;
        try{
❷           srequest.setCharacterEncoding(this.targetEncoding); //进行转码操作
            chain.doFilter(request, response);          //将 request、response 对象传递给下一个过滤器
        }catch(Exception e){
            e.printStackTrace();                                           //捕捉异常
        }
    }
    public void destroy(){                                                 //销毁过滤器
        this.filterConfig=null;                                            //置空
    }
}
```

代码贴士

❶ targetEncoding：获取在 web.xml 中定义的 code 参数的值。

❷ srequest.setCharacterEncoding(this.targetEncoding)：将页面的编码统一修改为此编码。

9.5.3　构建转码类

在项目开发过程中，数据库的编码通常是 ISO-8859-1，而项目编码往往是 UTF-8、GBK、GB2312 等，此时如果不在显示过程中对数据进行转码操作，页面上的中文就会出现乱码现象。

在本系统中，项目为 UTF-8 编码，所以笔者设计一个将 ISO-8859-1 与 UTF-8 编码之间互相转换的 com.wsy.StringTrans.java 类。其中包括将 ISO-8859-1 编码转换为 UTF-8 的方法。关键代码如下：

例程 07　代码位置：光盘\TM\09\net\src\com\wsy\StringTrans.java

```
public static String tranC(String chB){
    String result=null;
    byte temp[];
    try{
      temp=chB.getBytes("iso-8859-1");                    //将字符串以 byte 形式初始化 temp 数组
      result=new String(temp,"UTF-8");                    //将 temp 数组初始化为 UTF-8 编码的字符串
    }catch(UnsupportedEncodingException e){
```

```
            System.out.println(e.toString());                    //捕捉异常
        }
        return result;                                           //返回转换后的字符串
    }
```

9.6 网站首页设计

9.6.1 首页概述

现今的网站多得数不胜数，且网站的主题有的也大同小异，吸引浏览者的将不再只是网站所承载的信息，而是其美观、和谐的页面设计。在进行网站首页设计时，不但要求网站布局合理，而且还应该通过网站首页的主要功能模块充分体现出网站所要体现的主题内容，从而给浏览者留下更深刻的印象。

本企业门户网站主要包括信息栏、导航栏、企业信息展示和版权信息 4 部分。网站首页的运行效果如图 9.17 所示。

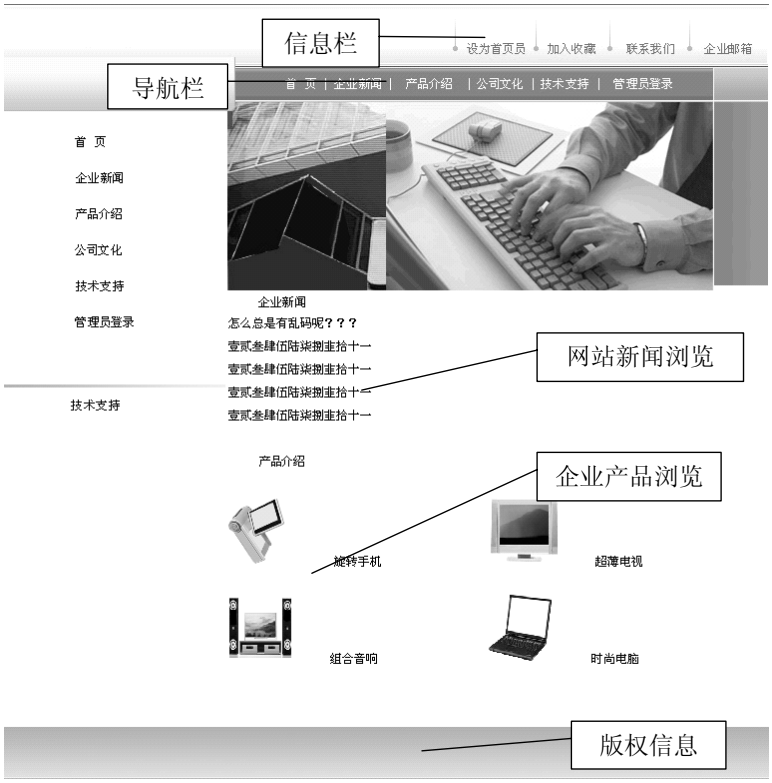


图 9.17 网站首页面

9.6.2 首页技术分析

网站首页主要包括企业新闻展示、产品介绍、公司文化、技术支持、管理员登录等功能链接。

在开发网站首页过程中，其中两个动态的部分分别为企业新闻信息浏览与企业商品信息浏览，管理员在后台管理系统中添加了网站新闻以及做了商品注册等操作，相应地将新闻与商品信息存入数据库中，在首页中只要调用对应的 JavaBean 中的数据库查询方法即可在首页显示新闻与商品的相关信息。另外，在网站首页中商品的展示位置需要在查询代码中使用分栏代码，将商品以分栏的格式进行显示。

同时在企业网站的首页中，通过图片热点超链接来实现图片链接。应用图片热点超链接实现图片链接，主要通过 HTML 的<map>标记为图片添加热点。语法如下：

```
<img src="file_name" usemap="#MapName">
<map name="MapName">
```

```
<area shape="value" coords="坐标" href="URL" alt="描述文字">
…
</map>
```

<map>标记的属性及说明如表 9.1 所示。

表 9.1　<map>标记的属性说明

属　性	说　明	属　性	说　明
name	图片热点的名称	href	设定区域的链接地址
shape	定义图片热点区域的形状	alt	设定区域链接的描述文字
coords	设定区域坐标		

在<map>标记中，属性 shape 的取值不同，相应坐标的设定也不同。下面介绍属性 shape 的 3 种取值以及相应坐标的设定。

☑　设定属性 shape 的属性值为 rect

属性 shape 取值为 rect，表示矩形区域，属性 coords 的坐标形式为“x1，y1，x2，y2”。其中，x1、y1 代表矩形左上角的 x 和 y 坐标，x2、y2 代表矩形右下角的 x 和 y 坐标。

☑　设定属性 shape 的属性值为 circle

属性 shape 取值为 circle，表示圆形区域，属性 coords 的坐标形式为“x，y，r”。其中，x、y 为圆心坐标，r 为圆的半径。

☑　设定属性 shape 的属性值为 poly

属性 shape 取值为 poly，表示多边形区域，属性 coords 的坐标形式为“x1，y1，x2，y2，…，xn，yn”。其中，xn，yn 代表构成多边形每一点的坐标值，n 的取值为“1，2，3，…，n”，多边形有几个边就有几对 x、y 坐标。

注意　可以在<body>区域中的任一位置定义<mapname=mapname></map>标签，name 是图片热点的名称。

9.6.3　首页的实现过程

开发首页主要包括以下几个功能操作。

☑　企业新闻信息展示

实现企业新闻信息展示功能的步骤如下：

（1）调用 JavaBean 中的企业新闻浏览方法，以集合的形式返回。关键代码如下：

例程 08　代码位置：光盘\TM\09\net\WebRoot\index.jsp

```
<%
    Collection temp2=sql.selectNews();          //调用 JavaBean 中的方法
    Iterator it2=temp2.iterator();              //以 iterator 函数获得集合中的数据
    while(it2.hasNext()){                       //循环结果集
        news news=(news)it2.next();             //将集合中的数据转换为 news.java 类输出
```

```
%>
<tr valign="top" >
                                                    <!-- 在页面显示新闻标题-->
<td height="19" colspan="2" background="images/014.jpg" class="zczi"><%=news.getTitle() %></td>
</tr>
<%} %>
```

（2）在 JavaBean 中的企业新闻查询方法，主要用于实现在数据库中查询企业新闻的相关信息。由于前台首页位置要求，所以这里笔者只取出新闻表中的前 5 条数据。关键代码如下：

例程 09 代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
public Collection selectNews(){
    Collection ret=new ArrayList();                          //初始化 Collection 集合
    try{
        connsqlserver connsqlserver=new connsqlserver();     //新建数据库连接
❶       String sql="select top 5 * from tb_news";            //查询新闻表中信息的 SQL 语句
❷       ResultSet rs=connsqlserver.executeQuery(sql);        //执行 SQL 语句
        while(rs.next()){                                    //循环结果集
            String title=rs.getString(2);                    //将数据库信息取出
            String author=rs.getString(3);                   //获取作者信息
            String news=rs.getString(4);                     //获取新闻信息
            news news1=new news();                           //初始化 news 类
            news1.setTitle(title);                           //将数据添加到 news.java 这个 JavaBean 中
            news1.setContent(news);                          //将新闻信息放入 JavaBean 中
            news1.setAuthor(author);                         //将作者信息放入 JavaBean 中
            ret.add(news1);                                  //将 news.java 对象添加到集合中
        }
    }catch(Exception e){
        e.printStackTrace();
    }
❸   connsqlserver.close();                                   //关闭数据库连接
    return ret;                                              //将集合返回
}
```

代码贴士

❶ sql：取出新闻表中的前 5 条记录。

❷ rs：执行 SQL 语句返回 ResultSet 结果集。

❸ connsqlserver.close()：关闭数据库连接。

☑ 企业商品信息展示

（1）在 JavaBean 中创建查询企业商品信息的方法。关键代码如下：

例程 10 代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
public ResultSet selectbusiness(){
    ResultSet rs=null;
    try{
```

```
        connsqlserver connsqlserver=new connsqlserver();        //新建数据库连接
        String sql="select   * from tb_business";               //查询商品表
        rs=connsqlserver.executeQuery(sql);                     //执行 SQL 语句
        rs.last();                                              //将 rs 游标放置到队列尾部
    }catch(Exception e){
        e.printStackTrace();                                    //捕捉异常
    }
    connsqlserver.close();                                      //关闭数据库连接
    return rs;                                                  //返回 ResultSet
}
```

（2）在首页中需要将商品信息以分栏形式排列，分栏具有很高的灵活性，可以使商品信息更清晰、一目了然。本实例所实现的分栏并不是用表格分出来的，而是单纯地使用行和列输出信息，然后通过双重循环控制行、列的输出信息。循环行、列，以分栏的形式输出商品相关信息，关键代码如下：

例程 11　代码位置：光盘\TM\09\net\WebRoot\index.jsp

```
<!-- 分栏显示 -->
<%
    int RowCount=4;                                             //显示数据总个数
    ResultSet Rs = sql.selectbusiness();                        //调用 JavaBean 中的方法
    int HRow = RowCount/2;                                      //预计分栏的行数
    if (RowCount%2>0)
    HRow++;
    for (int i = 0 ;i<HRow;i++){%>                              //循环行数
    <%
        for (int j=i*2+1;j<=(i+1)*2;j++){                       //循环列数
        Rs.absolute(j);
        if (Rs.isAfterLast())                                   //如果到最后一个记录
            break;                                              //终止程序运行
    %>
<!--显示商品的相关信息-->
<img src="images/spimg/<%=Rs.getString("img")%>" width="70" height="70"><%=Rs.getString("name")%>
<%}}%>
```

9.7　商品介绍模块设计

9.7.1　商品介绍模块概述

商品介绍模块主要用于实现企业商品展示功能，一个企业门户网站是否能将自身企业的优势展现给用户，丰富的商品资源是必不可少的因素，所以此模块在整个企业门户网站中占据着非常重要的地位，如何将商品合理安置在页面中是开发此模块时需要考虑的内容。兼顾整个企业网站页面风格，将商品以分栏方式罗列在商品介绍页面中。商品介绍页面如图 9.18 所示。

图 9.18　商品介绍页面

9.7.2　商品介绍模块技术分析

商品介绍功能模块主要是将数据库中企业的商品信息罗列到页面中，此时需要使用数据库查询语句。

无论是 Web 程序还是应用程序，当用户进行数据库查询时，都会对数据表中的数据进行显示，但是反馈给用户的记录数是不确定的。如果记录集中的记录较多或者兼顾前台页面相关信息的摆放位置，可以选择分页或者分栏进行数据显示。

在这里笔者选择了分栏显示商品信息的方式。分栏语句中不包含表格的行与列，而单纯地使用循环控制数据的摆放位置。

9.7.3　商品介绍模块的实现过程

开发商品介绍模块的步骤如下：

（1）在 JavaBean 中创建商品信息查询方法。关键代码如下：

例程 12　代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
public ResultSet selectbusiness(){
    ResultSet rs=null;
    try{
        connsqlserver connsqlserver=new connsqlserver();                    //创建数据库连接
        String sql="select   * from tb_business";                           //查询商品
```

```
            rs=connsqlserver.executeQuery(sql);                    //返回结果集
            rs.last();                                             //将游标放置在队列最后
        }catch(Exception e){
            e.printStackTrace();                                   //捕捉异常
        }
        return rs;                                                 //返回结果集
}
```

（2）在前台页面中使用分栏语句将商品信息放入页面中，由于需要分 2 行 3 列，所以总个数设置为 6。关键代码如下：

例程 13　代码位置：光盘\TM\09\net\WebRoot\cpjs.jsp

```
<tr>
    <td rowspan="2" width="543" height="339" background="images/cpjs/5.gif" class="zczi">
    <!-- 分栏显示 -->
    <%
        int RowCount=6;
❶       ResultSet Rs = sql.selectbusiness();                       //调用 JavaBean 中的方法
        int HRow = RowCount/2;                                     //获取行数
❷       if (RowCount%2>0)                                          //如果行数为双数行
            HRow++;                                                //行数自增
        for (int i = 0 ;i<HRow;i++){                               //循环每行
    %>
    <%
        for (int j=i*2+1;j<=(i+1)*2;j++){
        Rs.absolute(j);
        if (Rs.isAfterLast())                                      //如果游标到最后一行
            break;                                                 //终止程序
    %>
<img src="images/spimg/<%=Rs.getString("img")%>" width="70" height="70"><%=Rs.getString("name")%>
    <%}%><%}%>
</td>
</tr>
```

代码贴士

❶ sql.selectbusiness()：调用 JavaBean 方法，返回结果集。

❷ if (RowCount%2>0)：如果当前行数能整除数字 2，说明为双数行。

9.8　后台登录模块设计

9.8.1　后台登录模块概述

后台登录页面是进入企业门户网站后台管理的入口，在该页面中，系统管理员可以输入正确的用户名和密码登录到后台管理系统。当管理员没有输入用户名或密码时，系统会通过 JavaScript 脚本进行

判断，并给予提示信息。输入用户名和密码后，单击“提交”按钮，系统会将用户名和密码文本框放入提交表单中，然后在另一个页面获得表单中用户名与密码的值，使用 SQL 语句判断是否与数据库中的用户名、密码相符。后台登录页面如图 9.19 所示。

图 9.19　后台管理员登录页面

9.8.2　后台登录模块技术分析

后台登录模块使用 JavaBean 技术开发。JavaBean 往往封装了程序的页面逻辑，它是可重用的组件，通过使用 JavaBean 可以减少在 JSP 中脚本代码的使用，这样使得 JSP 易于维护、易于被非编程人员接受。

管理员进入后台管理页面必须通过系统登录页面进入，这是任何一个管理系统的保密性的需要。为了获取用户输入的用户名与密码文本框的值，首先将这两个文本框放入表单中，进行表单提交。这里笔者使用 JSP 页面接收表单中的值，在接收页面中使用<jsp:useBean id="" class="" scope=""/>标签引用 JavaBean。例如：

```
<jsp:useBean id="sql" class="com.wsy.selectsql" scope="page"/>
```

上述代码中的 id 元素为此 JavaBean 实例化的对象，名称为 sql；class 元素指明 JavaBean 所在的具体位置；scope 元素指明 JavaBean 的作用范围，这里指明 JavaBean 的作用范围为 page。

使用<jsp:useBean id="" class="" scope=""/>中元素 id 的值 sql 调用 JavaBean 中验证登录是否成功的方法，与此同时在这里用到了 com.wsy.connsqlserver.java 文件中的连接数据库方法、查询数据库方法、关闭数据库连接方法。

为了避免用户输入错误信息，这里笔者使用了 JavaScript 脚本代码验证用户名和密码文本框是否为空，如果为空，在页面中会弹出相应的错误提示。登录模块中 JavaScript 的关键代码如下：

例程 14　代码位置：光盘\TM\09\net\WebRoot\houtai\adminlogin.jsp

```
<script type="text/javascript">
<!--
function submit2(){
    if(document.all.name.value.length==0){          //判断表单中“用户名”文本框是否为空
        alert("请填写用户名!");                      //如果表单中“用户名”文本框为空，弹出错误提示
        return false;                               //返回 false
    }
    if(document.all.password.value.length==0){      //判断表单中“密码”文本框是否为空
        alert("请填写密码!");                        //如果表单中“密码”文本框为空，弹出错误提示
        return false;                               //返回 false
    }
    document.all.loginForm.submit();                //提交表单
    return true;
}
</script>
```

9.8.3　后台登录模块的实现过程

后台登录模块的实现步骤如下：

（1）创建后台登录页面 adminlogin.jsp 文件，将“用户名”与“密码”文本框放入表单中，提交到 houtaitest.jsp 文件。关键代码如下：

例程 15　代码位置：光盘\TM\09\net\WebRoot\houtai\adminlogin.jsp

```
<form action="houtaitest.jsp" name="loginForm">          <!--表单提交-->
<table width="527" height="356" border="0" align="center" cellpadding="0" cellspacing="0" id="__01">
…//省略部分代码
                                                         <!—“用户名”文本框-->
    <td width="58%" valign="baseline"><input type="text" name="name" size="20" maxlength="20"/></td>
  </tr>
  <tr>
    <td> </td>
                                                         <!—“密码”文本框-->
    <td valign="baseline"><input type="password" name="password" size="22" maxlength="20"/></td>
            <!--在页面中的图片按钮处做热点操作-->
            <img src="../images/ht03.gif" alt="" width="527" height="206" border="0" usemap="#Map"></td>
     </tr>
</table>
</form>
```

（2）为了获取表单中的值，需要一个承载表单中文本框属性的 JavaBean，命名为 user.java，此 JavaBean 除了 setXXX()方法与 getXXX()方法外还有两个属性，分别为 name 与 password。关键代码如下：

例程 16　代码位置：光盘\TM\09\net\src\com\wsy\user.java

```
public class user {
❶   String name;                                  //name 属性
❷   String password;                              //password 属性
    public user(){
        name="";                                  //将 name 属性置空
        password="";                              //将 password 属性置空
    }
    public String getName(){                      //name 属性的 getXXX() 方法
        return this.name;
    }
    public String getPassword(){                  //password 属性的 getXXX() 方法
        return this.password;
    }
    public void setName(String name){
        this.name=name;                           //name 属性的 setXXX() 方法
    }
    public void setPassword(String password){
```

```
        this.password=password;                                    //password 属性的 setXXX() 方法
    }
}
```

代码贴士

❶ name：JavaBean 中的成员变量，name 为用户名。

❷ password：JavaBean 中的成员变量，password 为密码。

（3）在 houtaitest.jsp 文件中，使用<jsp:setProperty property="*" name="user"/>获取表单中的值。关键代码如下：

例程 17 代码位置：光盘\TM\09\net\WebRoot\houtai\houtaitest.jsp

```
<jsp:useBean id="user" scope="page" class="com.wsy.user"/>      <!--引用 JavaBean-->
<!--使用 JavaBean 中的 setXXX()方法为 JavaBean 中的属性值赋值-->
<jsp:setProperty property="*" name="user"/>
<%
String name=user.getName().trim();                              <!--获取“用户名”文本框的值-->
String password=user.getPassword().trim();                      <!--获取“密码”文本框的值-->
%>
```

在上述代码中，使用了<jsp:setProperty property="*" name="user"/>，这个标签通常与<jsp:useBean/>标签结合使用，用于设置 JavaBean 中的属性值。当 property 属性被设置为“*”时（这是一种设置 JavaBean 属性的快捷方式），它自动将用户输入的值赋予 JavaBean 中的 setXXX()方法，这时如调用 getXXX()方法，即可取出用户在文本框中输入的值。

为了避免取出用户输入带有空格的值，需要使用 trim()方法，它可以将字符串中的空格去掉返回非空格的字符串。

注意 在使用<jsp:setProperty property="*" name="user"/>标签时，JavaBean 中的属性名称、类型必须与表单中的文本框名称相同。如果使用了 property="*"，JavaBean 中的属性没有必要按照表单中的顺序排序。

（4）在 com.wsy.selectsql.java 文件中添加登录验证方法——check()方法。关键代码如下：

例程 18 代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
public static int check(String name,String password){
    int i=0;
    String names="";
    String passwords="";
    try{
        //登录验证 SQL 语句
        String sql="select * from tb_usertable where name='"+name+"'and password='"+password+"'";
        rs=connsqlserver.executeQuery(sql);                        //执行 SQL 语句
        while(rs.next()){
            names=rs.getString("name");
            passwords=rs.getString("password");
```

```
                if(names!=null){
                    i=1;                                    //如果验证成功，给变量 i 赋值为 1
                }
            }
        }
        catch(Exception e){
            e.printStackTrace();                            //捕捉异常
        }
        connsqlserver.close();                              //关闭数据库连接
        return i;
}
```

（5）在 houtaitest.jsp 文件中引用 selectsql.java 文件，调用 check()方法。关键代码如下：

例程 19　代码位置：光盘\TM\09\net\WebRoot\houtai\houtaitest.jsp

```
<jsp:useBean id="sql" scope="page" class="com.wsy.selectsql"/>
<%
i=sql.check(name,password);                                 //调用表单验证方法
if(i==1){
    session.setAttribute("ok","ok");
    response.sendRedirect("index.jsp");                     //如果登录成功则转到后台管理页面
}
if(i==0){                                                   //如果登录失败
%>
<script>
    javaScript:window.alert("登录失败");                    //弹出相应对话框
</script>
<%
    response.sendRedirect("adminlogin.jsp");                //转到登录页面
}
%>
```

9.8.4　单元测试

在设计登录功能模块时容易产生一个漏洞，这就是如果用户直接在浏览器地址栏中输入“http://localhost:8080/net/houtai/index.jsp”，则无须登录即可进入后台管理页面。为了避免这样的错误，笔者采用 session 进行控制。步骤如下：

（1）在登录验证成功处添加 session。关键代码如下：

例程 20　代码位置：光盘\TM\09\net\WebRoot\houtai\houtaitest.jsp

```
i=sql.check(name,password);                                 //调用表单验证方法
<%
if(i==1){
    session.setAttribute("ok","ok");
    response.sendRedirect("index.jsp");                     //如果登录成功则转到后台管理页面
}
%>
```

（2）在后台管理页面开头添加接收 session 代码。关键代码如下：

例程 21 代码位置：光盘\TM\09\net\WebRoot\houtai\index.jsp

```
<%
    if(session.getAttribute("ok")!="ok")
     response.sendRedirect("adminlogin.jsp");              //如果 session 值不同，转入登录页面
%>
```

进行如上控制后，即使用户在浏览器地址栏中输入“http://localhost/：8080/net/houtai/index.jsp”路径，系统依然会返回 adminlogin.jsp 页面。

9.9 商品管理模块设计

9.9.1 商品管理模块概述

管理员登录成功后，进入后台管理页面，单击左侧的“商品管理”链接，即可进入商品管理页面。商品管理模块主要包括商品类别浏览、商品类别删除、商品类别修改、商品注册、商品浏览、商品删除等功能。其中商品浏览添加了分页显示功能。商品注册页面如图 9.20 所示。

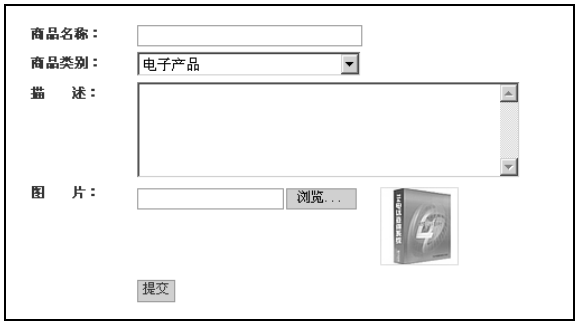


图 9.20 商品注册页面

9.9.2 商品管理模块技术分析

商品管理模块分为商品类别管理和商品管理，其中商品表与商品类别表具有外键联系。商品与商品类别管理主要包括添加、删除、修改、浏览等功能，其实现主要应用了以下技术。

（1）使用 Insert Into 语句

使用 Insert Into 语句实现商品类别与商品的添加，有关 Insert 语句的语法格式及参数说明可参见 9.10.2 节。

（2）使用 Update 语句

Update 语句主要用于更新单行上的一列或多列的值，或是更新单个表中选定的一些行上的多个列值。当然，为了在 Update 语句中修改指定表中的数据，必须有对表的 Update 访问权限。在本模块中主要应用 Update 语句实现对商品类别与商品的修改。Update 语句的语法格式如下：

```
UPDATE<table_name | view_name>
SET <column_name>=<expression>
    [….,<last column_name>=<last expression>]
[WHERE<search_condition>]
```

Update 语法中的参数说明如表 9.2 所示。

表 9.2　Update 语法中的参数说明

参　数	描　述
table_name	需要更新的表的名称。如果该表不在当前服务器或数据库中，或不为当前用户所有，这个名称可用链接服务器、数据库和所有者名称来限定
view_name	要更新的视图的名称。通过 view_name 来引用的视图必须是可更新的
SET	指定要更新的列或变量名称的列表
column_name	含有要更改数据的列的名称。column_name 必须位于 Update 子句中所指定的表或视图中
expression	变量、表达式或加上括号返回单个值的 subSELECT 语句。expression 返回的值将替换 column_name 或@variable 中的现有值
WHERE	指定条件来限定所更新的行
<search_condition>	为要更新行指定需满足的条件

一定要确保不要忽略 WHERE 子句，除非想要更新表中的所有行。

（3）使用 Delete 语句

本模块主要应用 Delete 语句实现商品类别信息与商品的删除。Delete 语句的语法格式如下：

```
DELETE FROM <table_name >
[WHERE<search_condition>]
```

- ☑ FROM：是可选的关键字，可用在 Delete 关键字与目标 table_name、view_name 或 rowset_function_ limited 之间。
- ☑ table_name：是要删除数据的表的名称。
- ☑ <search_condition>：指定删除行的限定条件。

如果想要一次性删除数据表中的所有记录，也可以使用 TRUNCATE TABLE 语句。其语法如下：

```
TRUNCATE TABLE table
```

TRUNCATE TABLE 语句的执行过程不会记录于事务日志文件中，因此速度较快，但删除后就无法利用事务日志文件恢复了。

9.9.3　商品管理模块的实现过程

1．商品类别管理

商品类别管理主要包括以下功能。

（1）商品类别添加

实现商品类别添加的步骤如下：

① 要实现商品类别添加功能，需要将类别文本框置于表单中。关键代码如下：

例程 22 代码位置：光盘\TM\09\net\WebRoot\houtai\categoryadd.jsp

```
<form action=categoryaddtest.jsp method="post">
    <p align=center class="lunzi"><Strong>商品类别-添加</strong></p>
<table width=75% border="0" cellspacing="0">
    <tr>
        <td width="42%" align="right"><font color="#663300" class="lunzi">商品类别名称:</font></td>
        <td width="58%">
            <input type="text" name="categoryname">          <!--商品类别文本框-->
            <input type="submit" value="保存">               <!--提交表单按钮，按钮名称为保存-->
        </td>
    </tr>
</table>
```

② 提交表单到相应的处理页面，此时可以获取表单中商品类别的值。关键代码如下：

例程 23 代码位置：光盘\TM\09\net\WebRoot\houtai\categoryaddtest.jsp

```
<%
    String additem=category.getCategoryname().trim();      <!--获取商品类别的值-->
    int i=sql.InsertCategory(a.tranC(additem));            <!--调用 JavaBean 中添加类别的方法-->
    response.sendRedirect("categoryBrowse.jsp");           <!--转入商品类别浏览页面-->
%>
```

③ 使用 Insert Into 语句将商品类别新增到数据库中，笔者将商品添加方法封装到 JavaBean 中。关键代码如下：

例程 24 代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
public int InsertCategory(String categoryname){
    int i=0;
    try{
        //数据库插入 SQL 语句
        String sql="insert into tb_category(categoryname) values('"+categoryname+"')";
        //执行 SQL 语句
        i=connsqlserver.executeUpdate(sql);
    }catch(Exception e){
        e.printStackTrace();                                  //捕捉异常
    }
    connsqlserver.close();                                    //关闭数据库连接
    return i;                                                 //将执行结果返回
}
```

（2）商品类别删除

当管理员单击商品类别浏览页面中的“删除”链接时，会弹出商品删除对话框，询问管理员是否确认删除此项。在商品类别浏览页面做“删除”链接时，需要将商品类别 id 传入商品类别删除处理页面。关键代码如下：

例程 25 代码位置：光盘\TM\09\net\WebRoot\houtai\categoryBrowse.jsp

```
out.println("<td><div align='center'><strong><a href='#'
onClick=window.open('categoryDelPage.jsp?catid="+category.getId()+"','newwindow','width=276,heigh
t=174,top=400,left=500')>删除</a></div></td>");          //弹出商品删除页面
```

说明　在页面中弹出窗口控制技术可参见 9.11.1 节，在此不再赘述。

在商品删除页面中，可以获取管理员需要删除的商品类别 id，以 id 作为参数调用 JavaBean 中商品类别删除方法，实现商品类别删除功能。商品类别删除方法的关键代码如下：

例程 26　代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
public int DelCategory(String id){
    int i=0;
    try{
❶       String sql="delete from tb_category where id='"+id+"'";      //商品类别删除的 SQL 语句
❷       i=connsqlserver.executeUpdate(sql);                          //执行 SQL 语句
    }catch(Exception e){
        e.printStackTrace();                                         //捕捉异常
    }
    connsqlserver.close();                                           //关闭数据库连接
    return i;
}
```

代码贴士

❶ sql：定义删除商品类别表的 SQL 语句。

❷ i：执行 SQL 语句，并将结果返回到变量 i 中。

（3）商品类别修改

当管理员单击商品类别浏览页面的“修改”链接时，会转入商品类别修改页面，管理员将需要修改的内容添加到页面的文本框中，此时需要在文本框中取出修改前的商品类别名称，为了实现这个功能，需要在“修改”链接处添加商品类别 id，在商品类别修改页面便可根据商品类别 id 调用商品类别查询方法显示未修改的商品类别名称。关键代码如下：

例程 27　代码位置：光盘\TM\09\net\WebRoot\houtai\categoryBrowse.jsp

```
out.println("<td><div align='center'><a
href='categoryEditPage.jsp?catid="+category.getId()+"&name="+category.getCategoryname()+"'>修改
</a></div></td>");
```

在商品类别修改页面，可以获取管理员修改的商品类别名称，调用 JavaBean 中商品类别修改方法，实现商品类别修改功能。JavaBean 中的关键代码如下：

例程 28　代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
public int UpdateCategory(String edititem,String id){
    int i=0;
    try{
        connsqlserver connsqlserver=new connsqlserver();             //获取数据库连接
        //定义数据库修改的 SQL 语句
        String sql="update tb_category set categoryname='"+edititem+"' where id='"+id+"'";
        //执行 SQL 语句
        connsqlserver.executeUpdate(sql);
```

```
    }catch(Exception e){
        e.printStackTrace();                                    //捕捉异常
    }
    connsqlserver.close();                                      //关闭数据库连接
    return i;                                                   //将修改结果返回
}
```

2．商品管理

1）商品注册

商品注册应用到 Insert Into 语句，将用户的注册信息添加到数据表中；另外，本模块中还实现了上传图片并根据上传文本框中的值即时更改页面中图片的功能。

（1）图片上传。在这里笔者使用 FileUpload 组件实现图片上传。步骤如下：

① 首先将 FileUpload 组件的 commons-fileupload-1.0.jar 放入项目路径下的 WEB-INF\lib 目录中，在 MyEclipse 中刷新项目，commons-fileupload-1.0.jar 会自动加载到项目中。

② 在商品添加页面添加 Form 表单，在表单属性中，方法必须是 Post，并且必须添加<ENCTYPE="multipart/form-data">属性，否则不能实现上传。关键代码如下：

例程 29 代码位置：光盘\TM\09\net\WebRoot\houtai\productadd.jsp

```
<form name="form1" method="post" action="save.jsp" ENCTYPE="multipart/form-data">
    <tr>
                                                                //上传文本框
        <td width="213" height="78" valign="top"><input type="file" name="file" onChange="showlogo()"></td>
        <SCRIPT language=javascript>
        //通过下拉列表选择头像时应用该函数
        function showlogo(){
                document.form1.img.src=document.form1.file.value;
        }
        </SCRIPT>
        <td width="231" valign="top"><img src="../images/spimg/11.bmp" width="70" height="70" id="img"></td>
    </tr>
</form>
```

③ 在文件上传接收页面中，使用 FileUpload 组件中的方法读取管理员上传的图片相关信息，将图片文件保存到相应的目录下。关键代码如下：

例程 30 代码位置：光盘\TM\09\net\WebRoot\houtai\save.jsp

```
<%
    DiskFileUpload fu=new DiskFileUpload();                     //设置允许用户上传文件的大小，单位：字节
    List fileitems=fu.parseRequest(request);                    //开始读取上传信息
    String name2=null;
%>
<%
   Iterator iter=fileitems.iterator();                          //将结果集转换为迭代函数的形式
    while(iter.hasNext()){
```

```
        FileItem item = (FileItem) iter.next();
        if(!item.isFormField()){                          //如果是文件域，即使用<input type="file"/>标签
            String name1=item.getName();                  //读取文件域的名称
            long size=item.getSize();                     //获取上传文件的大小
❶           name1=name1.replace(':','-');                 //将“:”字符以“-”字符替换
❷           name1 = name1.replace('\\','-');              //将“\\”字符以“-”字符替换
❸           String name[]=name1.split("-");               //以“-”字符分割字符串返回 name 数组
❹           name2=name[name.length-1];                    //取数组中最后一个元素，即不带路径的文件名称
            //将文件保存到指定文件夹
            File path = new File("D:\\Upload");           //声明 File 类型的对象
            if (!path.isDirectory()) {                    //如果本地硬盘不存在此路径
❺               path.mkdir();                             //创建此路径
            }
❻           item.write(new File(path + "\\" + name2));    //将文件保存在此路径下
        }
    }
%>
```

代码贴士

❶ name1 = name1.replace(':','-'): 将字符串中的“:”字符以“-”字符替换。

❷ name1 = name1.replace('\\','-'): 将字符串中的“\\”字符以“-”字符替换。

❸ String name[]=name1.split("-"): 以“-”字符分割字符串返回 name 数组。

❹ name2=name[name.length-1]: 取数组中最后一个字符赋予 name2 变量。

❺ path.mkdir(): 创建目录。

❻ item.write(new File(path + "\\" + name2)): 将文件保存在此目录中。

④ 以上代码是针对文件域，细心的读者也许会发现非文件域文本框的值在 save.jsp 文件中使用 request.getParameter()方法获取不到。这时有两种解决方案，分别为使用两个表单或使用 FileUpload 组件自带方法获取表单中非文件域文本框中的值。在这里笔者选择使用两个表单提交的方案。至于第二种方案，笔者会在 9.11.2 节中进行介绍。

在非文件域中再设置一个表单，名称为 form2，这个表单没有设置 action。关键代码如下：

例程 31　代码位置：光盘\TM\09\net\WebRoot\houtai\productadd.jsp

```
<form action="" name="form2"/>                            <!--第二个表单-->
    <tr>
        <td width="91" class="lunzi">商品名称：</td>
        <td colspan="2"><input name="name" type="text" size="34"></td>
    </tr>
    <tr>
      <td class="lunzi">商品类别：</td>
      <td colspan="2"><select name="category" style="width:200px ">
        <!--调用 JavaBean 在数据库中取出商品类别名称放入表单的下拉列表中-->
         <%
    Collection temp=sql.selectCategoryAll();              //查询商品类别
    Iterator it=temp.iterator();
    while(it.hasNext()){                                  //循环结果集
        category category=(category)it.next();            //将结果放入 category 类中
```

```
 %>
<!--将数据查询结果放入下拉列表中-->
 <option value=<%=category.getCategoryname() %>><%=category.getCategoryname() %></option>
 <%} %>
  </select></td>
</tr>
<tr>
  <td class="lunzi">描        述：</td>
  <td colspan="2" rowspan="3"><textarea name="ms" cols="40" rows="5"></textarea></td>
</tr>
```

使用 JavaScript 代码设置两个表单提交到同一页面。关键代码如下：

例程 32 代码位置：光盘\TM\09\net\WebRoot\houtai\productadd.jsp

```
<SCRIPT>
    function ok(){
        var name = form2.name.value;                          <!--获取 form2 中的元素的值-->
        var category=form2.category.value;
        var ms= form2.ms.value;
        <!--将 form2 中的元素的值作为参数提交到 save.jsp 文件-->
        form1.action="save.jsp?name="+name+"&category="+category+"&ms="+ms;
        form1.submit();                                       <!--表单提交-->
    }
</SCRIPT>
```

最后再上传处理页面获取 form2 表单中的值，调用 JavaBean 中商品信息数据插入方法，实现商品注册功能。关键代码如下：

例程 33 代码位置：光盘\TM\09\net\WebRoot\houtai\save.jsp

```
<%
if((request.getParameter("name")!=null)&&(request.getParameter("category")!=null)&&(request.getParameter("ms")!=
null)){
    String productname=a.tranC(request.getParameter("name"));          //获取 form2 表单中的值
    String category=a.tranC(request.getParameter("category"));         //获取 form2 表单中的类别
    String ms=a.tranC(request.getParameter("ms"));                     //获取 form2 表单中的名称
    int i=sql.InsertBusiness(category,productname,ms,name2);           //执行数据库插入操作
    if(i==1){                                                          //如果插入成功，进行如下处理
%>
        <script language="JavaScript">
        window.alert("添加成功");                                      //弹出“添加成功”对话框
        window.close();                                                //关闭当前窗口
        </script>
<%
    }
}
%>
```

（2）根据上传文本框中的值即时更改页面中的图片。此功能主要通过 JavaScript 代码实现，实质上就是取得文件域文本框中的值赋予页面中图片的路径。关键代码如下：

例程 34　代码位置：光盘\TM\09\net\WebRoot\houtai\productadd.jsp

```
<SCRIPT language=javascript>
    //通过下拉列表选择头像时应用该函数
    function showlogo(){
        //将表单中文件域文本框中的值赋予表单中图片路径
        document.form1.img.src=document.form1.file.value;
    }
</SCRIPT>
```

在文件域文本框中调用上述 JavaScript 代码。关键代码如下：

```
<input type="file" name="file" onChange="showlogo()"> <!--在 file 文本框中调用 JavaScript 代码-->
```

2）商品浏览

商品浏览功能主要调用 JavaBean 中商品浏览方法实现。由于此页面添加了分页功能，所以 JavaBean 的方法中使用的 SQL 语句比一般的查询语句复杂一些。关键代码如下：

例程 35　代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
Public Collection selectBusinessFy(int page){                    //参数 page 为当前页
    Collection ret=new ArrayList();                              //实例化一个集合对象
    DownTable down=new DownTable();                              //初始化 DownTable 对象
    try{
        connsqlserver connsqlserver=new connsqlserver();         //新建数据库连接
        String sql="select top 10 * from tb_business where id not in(select top "+down.getPageSize()*
page+" id from tb_business order by id)order by id";             //分页查询语句
        ResultSet rs=connsqlserver.executeQuery(sql);            //执行 SQL 语句
        while(rs.next()){
            product product=new product();
            product.setId(rs.getString("id"));//将数据库中的值放入 product.java 类中的 setXXX() 方法中
            product.setImg(rs.getString("img"));                 //将商品图片名称放入 JavaBean 中
            product.setMsg(rs.getString("ms"));                  //将商品描述放入 JavaBean 中
            product.setName(rs.getString("name"));               //将商品名称放入 JavaBean 中
            product.setSubmittime(rs.getString("submittime"));   //将提交时间放入 JavaBean 中
            product.setCategory(rs.getString("category"));       //将商品类别放入 JavaBean 中
            ret.add(product);                                    //将对象添加到集合中
        }
    }catch(Exception e){
        e.printStackTrace();                                     //捕捉异常
    }
    connsqlserver.close();                                       //关闭数据库连接
    return ret;                                                  //将集合返回
}
```

在上述代码中引用到了一个 JavaBean，名为 product.java，它的作用在于承载商品相关信息，其属性与商品相关信息相互对应，除此之外，JavaBean 还有与这些属性相对应的 setXXX()与 getXXX()方法。关键代码如下：

例程 36 代码位置：光盘\TM\09\net\src\com\wsy\product.java

```
public class product {
    private String id;                                          //商品编号
    private String name;                                        //商品名称
    private String img;                                         //商品图片
    private String msg;                                         //商品描述
    private String category;                                    //商品类别
    private String submittime;                                  //商品提交时间
    public String getCategory() {                               //商品类别 getXXX() 方法
        return category;
    }
    public void setCategory(String category) {                  //商品类别 setXXX() 方法
        this.category = category;
    }
    public String getId() {                                     //商品编号 getXXX() 方法
        return id;
    }
    public void setId(String id) {                              //商品编号 setXXX() 方法
        this.id = id;
    }
    public String getImg() {                                    //商品图片 getXXX() 方法
        return img;
    }
    public void setImg(String img) {                            //商品图片 setXXX() 方法
        this.img = img;
    }
    public String getMsg() {                                    //商品描述 getXXX() 方法
        return msg;
    }
    public void setMsg(String msg) {                            //商品描述 setXXX() 方法
        this.msg = msg;
    }
    public String getName() {                                   //商品名称 getXXX() 方法
        return name;
    }
    public void setName(String name) {                          //商品名称 setXXX() 方法
        this.name = name;
    }
    public String getSubmittime() {                             //商品提交时间 getXXX() 方法
        return submittime;
    }
    public void setSubmittime(String submittime) {              //商品提交时间 setXXX() 方法
        this.submittime = submittime;
    }
}
```

除了 Selectsql.java 类与 Product.java 类之外，实现分页功能还需要一个分页辅助类，用于计算所要查询表的相关信息。关键代码如下：

例程 37　代码位置：光盘\TM\09\net\src\com\wsy\DownTable.java

```
public class DownTable {
    int totalPages=1;                                          //总页数
    int pageSize=10;                                           //每页显示行数
    int currentPage=1;                                         //当前页
    ResultSet rs=null;
    int totalRows;                                             //总行数
❶   public int getTotalPage(){                                 //获取总页数
❷       if(getRows()%getPageSize()==0)                         //如果总行数可以整除每页显示的行数
            return getRows()/getPageSize();                    //返回总行数与每页显示记录个数的商
        else
            return getRows()/getPageSize()+1;                  //如果不可以整除，返回两者的商加 1
    }
    public void setpageSize(int size){                         //每页显示的行数属性的 setXXX() 方法
        this.pageSize=size;
    }
    public int getPageSize(){                                  //每页显示的行数属性的 getXXX() 方法
        return pageSize;
    }
    public void setCurrentPage(int current){                   //当前页属性的 setXXX() 方法
        this.currentPage=current;
    }
    public int getCurrentPage(){                               //当前页属性的 getXXX()方法
        return currentPage;
    }
    public int getRows(){
        connsqlserver con=new connsqlserver();                 //获取数据库连接
        try{
            //查询商品表的总个数
❸           rs=con.executeQuery("select count(*) from tb_business");
            if(rs.next()){
❹               totalRows=rs.getInt(1);                        //将总行数赋予 totalRows 变量
            }
        }
        catch(Exception e){
        }
❺       con.close();                                           //关闭数据库连接
        return totalRows;                                      //返回总行数
    }
}
```

代码贴士

❶ getTotalPage()：自定义获取总页数的方法。

❷ getRows()%getPageSize()==0：判断数据表的总行数是否能整除数据表的总记录数。其中 getRows()方法与 getPageSize()方法为自定义方法，分别为取表格的总行数与总记录数。

❸ executeQuery("select count(*) from tb_business")：查询 tb_business 表的总个数。

❹ totalRows=rs.getInt(1)：获取表格中数据总行数赋予变量 totalRows。

❺ con.close()：关闭数据库连接。

在商品浏览页面设置了分页链接，使用 JavaScript 代码控制页面显示。关键代码如下：

例程 38 代码位置：光盘\TM\09\net\WebRoot\houtai\productadd.jsp

```
function gotoPage(pagenum){
    document.PageForm.current.value=pagenum;  // PageForm 为表单名称，current 为分页下拉列表名称
    document.PageForm.submit();                    //进行表单提交
    return;
}
<td width=19%><div align="center" class="whitezi"><a href="javascript:gotoPage(1)">首页</a></div></td>
<td width=22%><div align="center" class="whitezi">
//其中 down 为 Downtable 类的对象，调用获取当前页的方法
<a href="javascript:gotoPage(<%=down.getCurrentPage()-1 %>)">上一页</a></div></td>
<td width=22% align="center"><span class="whitezi"><a href="javascript:gotoPage(<%=down.getCurrentPage()+1 %>)">下一页</a></span></td>
<td width=18% align="center"><span class="whitezi"><a href="javascript:gotoPage(<%=down.getTotalPage() %>)">尾页</a></span></td>
```

在商品浏览页面，除了添加“上一页”、“下一页”、“首页”、“尾页”等链接外，还添加了分页跳转下拉列表。将上述两个控件放入表单中，表单使用 JavaScript 代码设置提交到本页面，提交后根据当前的页数在页面中显示相应的数据集。关键代码如下：

例程 39 代码位置：光盘\TM\09\net\WebRoot\houtai\productadd.jsp

```
function Jumping(){
    document.PageForm.submit();                    //提交表单
    return;
}
<select name="current" onChange="Jumping()">
<%for(int i=1;i<=down.getTotalPage();i++){          //根据总页数显示下拉列表
    if(i==down.getCurrentPage()){
%>
<option selected value=<%=i %>><%=i %></option>     //将页码放入下拉列表中
<%}else{ %>
<option value=<%=i %>><%=i %></option>
<%}} %>
```

最后在页面中获取分页跳转下拉列表值，以此值作为参数调用 JavaBean 中的分页方法实现分页功能。关键代码如下：

例程 40 代码位置：光盘\TM\09\net\WebRoot\houtai\productadd.jsp

```
<%
    if(request.getParameter("current")==null){          //第一次进入页面无表单提交时赋予 current 为 1
    current=1;
}
else{
    current=Integer.parseInt(request.getParameter("current"));     //获取提交过来的 current 的值
}
if(current>=MaxPage){                                   //如果当前页码大于数据表最大行数
```

```
        current=MaxPage;                                    //如果当前页大于最大页数，则将最大页数赋予当前页
    }
    if(current<=MinPage){
        current=MinPage;                                    //如果当前页小于最小页数，则将最小页数赋予当前页
    }
    <%

    Collection temp=sql.selectBusinessFy(current-1);        //调用分页方法
    Iterator it=temp.iterator();
    int count=0;
    while(it.hasNext())                                     //循环结果集
    {
        product product=(product)it.next();                 //将集合中的数据转换为 product.java 形式
        if(count%2==0)
        out.println("<tr bordercolor='#FFFFCC' bgcolor='#CCFFFF'>");
        else
        out.println("<tr bgcolor='#CCCCFF'>");
        //显示商品名称
        out.println("<td><div align='center' class='zczi'>"+product.getName()+"</td>");
        //显示商品信息
        out.println("<td colspan='3'><div align='center'    class='zczi'>"+product.getMsg()+"</div></td>");
        //做商品删除链接
        out.println("<td><div align='center'><a href='#'
    onclick=window.open('productDelPage.jsp?catid="+product.getId()+"&name="+product.getName()+"','n
    ewwindow','width=276,height=174,top=400,left=500')>删除</a></td>");
        //做商品修改链接
        out.println("<td><div align='center'><a href='#'
    onclick=window.open('productview.jsp?catid="+product.getId()+"','newwindow','width=600,height=350,
    top=300,left=300')>查看</a></td>");
        out.println("</tr>");
        count++;
    }
    %>
```

3）商品删除

商品删除功能与商品类别删除功能实现基本相同，唯一不同的是在商品删除处理页面调用了 JavaBean 中的商品删除方法。此方法的关键代码如下：

例程 41　代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
public int DelBusiness(String id){
    int i=0;
    try{
        connsqlserver connsqlserver=new connsqlserver();                    //创建数据库连接
❶           String sql="delete from tb_business where id='"+id+"'";         //删除商品 SQL 语句
❷           i=connsqlserver.executeUpdate(sql);                             //执行 SQL 语句
        }catch(Exception e){
            e.printStackTrace();
        }
        connsqlserver.close();                                              //关闭数据库连接
```

```
        return i;                                                    //将删除结果返回
}
```

代码贴士

❶ sql：定义用户在数据表中选择需要删除行的 SQL 语句。

❷ i：调用 executeUpdate()方法，执行 SQL 语句，将结果返回给变量 i。

9.9.4 单元测试

在数据库连接操作中，如果在创建连接时不判断连接是否为空，一味创建数据库连接，即使使用连接池这种高效数据库连接技术，如果连接过多连接池最终也会抛出数据库连接异常，如图 9.21 所示。

```
org.apache.tomcat.dbcp.dbcp.SQLNestedException: Cannot get a connection, pool exhausted
        at org.apache.tomcat.dbcp.dbcp.PoolingDataSource.getConnection(PoolingDataSource.java:103)
        at org.apache.tomcat.dbcp.dbcp.BasicDataSource.getConnection(BasicDataSource.java:540)
        at com.wsy.connsqlserver.getConnection(connsqlserver.java:28)
        at com.wsy.connsqlserver.executeQuery(connsqlserver.java:40)
```

图 9.21 连接池抛出异常

这时需要优化数据库连接 JavaBean，使数据库连接更为高效。关键代码如下：

例程 42 代码位置：光盘\TM\09\net\src\com\wsy\connsqlserver.java

```
private void getConnection() {
        if (cn != null) {                    //新增代码
            return;                          //当数据库连接不为空，进行返回操作，不进行数据库连接操作
        }
        …//非关键代码省略
        return;                              //返回
    }
public ResultSet executeQuery(String sql) {
        if (cn == null)
            getConnection();                 //当数据库连接为空时，创建数据库连接
        …//非关键代码省略
    }
```

除此之外，还要注意在处理完成一个数据业务逻辑后，需要及时关闭数据库连接，否则程序极容易抛出异常，影响项目的质量。

9.10 新闻管理模块设计

9.10.1 新闻管理模块概述

进入系统后台管理页面后，单击左侧的“网页新闻管理”链接，即可进入新闻管理页面。新闻管理模块主要包括新闻添加、新闻删除、新闻修改、新闻浏览等功能。

新闻添加页面如图 9.22 所示。

9.10.2　新闻管理模块技术分析

1．新闻添加

新闻添加主要实现网页新闻添加，为了获取新闻相关信息，需要将这些信息的文本框放入表单中，提交到其他页面调用 JavaBean 中新闻添加的方法进行操作。新闻添加功能主要用到如下两种技术：

（1）使用 Insert Into 语句实现向指定的数据表中插入数据信息。其语法格式如下：

```
INSERT INTO table_name [(column_list)] Values(data_values)
```

- ☑ table_name：要添加记录的数据表名称。
- ☑ column_list：表中的字段列表，表示向表中哪些字段插入数据。如果是多个字段，字段之间用逗号分隔。不指定 column_list，默认向数据表中所有字段插入数据。
- ☑ data_values：要添加的数据列表，各个数据之间使用逗号分隔。数据列表中的数据个数、数据类型必须和字段列表中的字段个数、数据类型一致。

注意　对于省略的字段，SQL Server 按下列顺序进行处理：

① 如果字段为计算字段、标识字段，则自动产生其值。

② 如果不能自动产生其值，但字段设置了默认值，则填入默认值。

③ 如果该字段不能自动产生值，又没有设置默认值，但字段允许空值，则填入 NULL。

④ 如果字段不能自动产生值，又没有设置默认值，并且字段不允许空值，则显示错误提示信息，不输入任何数据。

在创建新闻信息表时，笔者将 submittime 字段默认值设置为 getdate()，它的作用是在添加数据时自动添加当前时间，如图 9.23 所示。

图 9.22　网站新闻添加页面

图 9.23　SQL Server 2000 设置字段默认值

（2）使用 JavaScript 脚本为了避免用户在文本区域中输入的数值超过数据库中定义的数据长度，所以在页面中使用 JavaScript 脚本控制用户输入的值。关键代码如下：

例程 43　代码位置：光盘\TM\09\net\WebRoot\houtai\news.jsp

```
<SCRIPT language=JavaScript>
```

```
var LastCount =0;
function CountStrByte(Message,Total,Used,Remain){
  var ByteCount = 0;
  var StrValue = Message.value;                          //取文本区域字符串的内容
  var StrLength = Message.value.length;                  //取文本区域字符串的长度
  var MaxValue = Total.value;                            //取总字数
if(LastCount != StrLength){                              //如果总字数不等于当前字数
      for (i=0;i<StrLength;i++){
      ByteCount    = (StrValue.charCodeAt(i)<=256) ? ByteCount + 1 : ByteCount + 2;
        if (ByteCount>MaxValue) {
              Message.value = StrValue.substring(0,i);
      alert("留言内容最多不能超过 " +MaxValue+ " 个字节！\n 注意：一个汉字为两字节。");
           ByteCount = MaxValue;
           break;                                        //跳出循环
        }
      }
    Used.value = ByteCount;                              //将当前字数赋予已经写入的字数文本框
    Remain.value = MaxValue - ByteCount;                 //将总字数减去当前的字数为可以输入的字数
    LastCount = StrLength;
 }
}
</SCRIPT>
```

2．新闻浏览

新闻浏览实现将数据库中的新闻相关信息显示在页面中，当用户单击后台管理页面左侧的“新闻信息浏览”链接时，在右侧主页中将显示网站新闻的相关信息。

在新闻浏览页面中调用 JavaBean 的数据表查询方法，返回 Collection 集合，使用 iterator 函数将新闻相关信息取出。为了实现此功能，需要用到用于承载新闻相关信息的 JavaBean，此 JavaBean 为 news.java 文件，它的属性为新闻相关信息，名称、类型必须与 JSP 页面表单中的字段名称严格保持一致。

3．新闻删除

新闻删除实现将数据库中的新闻删除，它同样需要调用操作数据库的 JavaBean，根据 id 值使用 delete 语句将新闻信息表中某一行信息删除。

4．新闻修改

新闻修改主要实现将数据库中的新闻某个字段的内容修改，它同样需要调用操作数据库的 JavaBean，根据 id 值使用 update 语句将新闻信息表中某一个行信息进行修改。

9.10.3 新闻管理模块的实现过程

1．新闻添加

实现新闻添加功能的步骤如下：

（1）为了避免用户输入的文字个数大于数据表中设定的长度，在新闻添加页面的文本区域调用 9.10.2 节中新闻添加所描述的 JavaScript 代码。关键代码如下：

例程 44　代码位置：光盘\TM\09\net\WebRoot\houtai\news.jsp

```
<textarea name="content" cols="70" rows="10" wrap="VIRTUAL" id="content"
onkeydown="CountStrByte(this.form.content,this.form.total,this.form.used,this.form.remain);"  onkeyup=
"CountStrByte(this.form.content,this.form.total,this.form.used,this.form.remain);"></textarea>
```

在文本区域设置 3 个文本框，分别用于显示最多字数、已用字数、剩余字数，为了使这 3 个文本框为只读，可以使用如下代码：

```
<input name="used" type="text" disabled class="noborder" id="used"   value="0" size="4" class="zczi">
```

（2）将新闻相关信息文本框放入表单中，提交到相应的逻辑处理页面。关键代码如下：

例程 45　代码位置：光盘\TM\09\net\WebRoot\houtai\news.jsp

```
<form name="form1" method="post" action="insert.jsp">
<table width="80%" border="0" align="center" cellpadding="0" cellspacing="1" class="tableBorder">
<%
  String test=(String)session.getAttribute("test");      //在逻辑处理页面返回一个 session，在此进行判断
  if(test!=null){
%>
    <tr>
        <th colspan="2" class="lunzi">恭喜，添加成功！</th>      //如果添加成功，在页面显示相应文字
    </tr>
<%}
    session.removeAttribute("test");                                    //删除 session
%>
    <tr>
        <th colspan="2" class="whitezi"><p class="lunzi">新闻添加</p>
        <p> </p></th>
    </tr>
    <tr>
        <td width="17%" height="22" align="right"><span class="lunzi">名   
        称：</span></td>
        <td width="83%"><font color="navy" face="Arial">
            <input name="title" size="37"><!--新闻标题文本框-->
        </font>
        </td>
    </tr>
    <tr>
        <td height="22" align="right" class="lunzi"><span class="lunzi">发    布
    者</span>：</td>
            <!--新闻作者文本框-->
        <td><input name="author" type="text" id="author" value="未知" size="37">
        </td>
    </tr>
    <tr>
        <td height="22" align="right" class="lunzi">内         
   容：</td>
```

```
            <td align="left">
            <!--文本区域调用 JavaScript 代码-->
❶   <textarea name="content" cols="70" rows="10" wrap="VIRTUAL" id="content"
onkeydown="CountStrByte(this.form.content,this.form.total,this.form.used,this.form.remain);"
onkeyup="CountStrByte(this.form.content,this.form.total,this.form.used,this.form.remain);"></textarea>
            </td>
        </tr>
        <tr>
            <td colspan="2" id="upid"> </td>
        </tr>
        <tr><td colspan="2" align="center" class="lunzi">
❷              最多允许 <input name="total" type="text" disabled class="noborder" id="total"  value="1600"
size= "4" class="zczi"> 个字节 已用字节： 
❸   <input name="used" type="text" disabled class="noborder" id="used" value="0" size="4" class=
"zczi">
❹   剩余字节：<input name="remain" type="text" disabled class="noborder" id="remain" value="1600"
size="4" class="zczi">
            </td>
        </tr>
        <tr>
            <td height="22" align="right"> </td>
            <td align="left" class="forumRow">
            <input type=Submit class=button value="提  交" name=Submit>      <!--“提交”按钮-->
            <input type=reset name=Submit2 class=button value="清  除">      <!--“重置”按钮-->
            </td>
        </tr>
</table>
</form>
```

代码贴士

❶ <textarea name="content">：定义文本区域，调用 JavaScript 代码。

❷ <input name="total">：定义总字数文本框，给定 value 值。

❸ <input name="used">：定义已经在文本区域中写入的字数的文本框，其值会随着字符的输入数字增加。

❹ <input name="remain">：定义还剩下多少字数可以输入的文本框，它会随着字符的输入数字减少。

（3）实现新闻添加操作，需要调用 JavaBean 中的数据库插入方法。关键代码如下：

例程 46　代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
public int Insert(String title,String author,String news){
    int i=0;
    try{
        //数据库插入语句
        String sql="insert into tb_news(title,author,content) values('"+title+"','"+author+"','"+news+"')";
        i=connsqlserver.executeUpdate(sql);              //执行 SQL 语句
    }catch(Exception e){
        e.printStackTrace();                             //捕捉异常
    }
    connsqlserver.close();                               //关闭数据库连接
```

```
    return i;                                                    //将执行结果返回
}
```

（4）在 insert.jsp 页面中，获取表单提交过来的数据，调用 JavaBean 中的数据库插入方法，将新闻相关信息添加到数据库中。关键代码如下：

例程 47　代码位置：光盘\TM\09\net\WebRoot\houtai\insert.jsp

```
<%
    String title=news.getTitle().trim();                         //分别获取表单中的值
    String author=news.getAuthor().trim();
    String content=news.getContent().trim();
    if(title!=null&&author!=null&&content!=null){                //如果表单中的值不为空
        int i=sql.Insert(s.tranC(title),s.tranC(author),s.tranC(content));    //进行数据库插入操作
        String x=null;
        if(i==1){
            x="恭喜，添加完毕！";                                  //如果插入成功，做“成功”的 session
        }
        else{
            x="添加失败";                                          //如果插入失败，做“失败”的 session
        }
        session.setAttribute("test",x);                          //做名称为 test 的 session
        response.sendRedirect("news.jsp");                       //转入新闻添加页面
    }
%>
```

2．新闻浏览

实现新闻浏览功能的步骤如下：

（1）在实现新闻浏览功能时，需要一个承载新闻相关信息的 JavaBean，名为 news.java。关键代码如下：

例程 48　代码位置：光盘\TM\09\net\src\com\wsy\news.java

```
public class news {
    String title;                                                //新闻标题
    String author;                                               //新闻作者
    String content;                                              //新闻内容
    String id;                                                   //新闻编号
    String submittime;                                           //新闻提交时间
    public news(){                                               //构造函数
        title=null;
        author=null;
        content=null;
        id=null;
        submittime=null;
    }
    public String getAuthor() {                                  //新闻作者 getXXX() 方法
        return author;
    }
```

```
    public void setAuthor(String author) {                    //新闻作者 setXXX() 方法
        this.author = author;
    }
    public String getContent() {                              //新闻内容 getXXX() 方法
        return content;
    }
    public void setContent(String content) {                  //新闻内容 setXXX() 方法
        this.content = content;
    }
    public String getId() {                                   //新闻编号 getXXX() 方法
        return id;
    }
    public void setId(String id) {                            //新闻编号 setXXX() 方法
        this.id = id;
    }
    public String getSubmittime() {                           //提交时间 getXXX() 方法
        return submittime;
    }
    public void setSubmittime(String submittime) {            //提交时间 setXXX() 方法
        this.submittime = submittime;
    }
    public String getTitle() {                                //新闻标题 getXXX() 方法
        return title;
    }
    public void setTitle(String title) {                      //新闻标题 setXXX() 方法
        this.title = title;
    }
```

（2）在 selectsql.java 文件中，添加一个查询新闻相关信息的方法。关键代码如下：

例程 49 代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
public Collection selectNewsAll(){
    Collection ret=new ArrayList();                           //初始化一个 Collection 对象
    try{
❶       String sql="select * from tb_news";                   //定义查询新闻相关信息的 SQL 语句
❷       ResultSet rs=connsqlserver.executeQuery(sql);         //执行 SQL 语句
        while(rs.next()){
❸           news news1=new news();                            //实例化 news 对象
            news1.setId(rs.getString("id"));        //将数据库中的数值赋予 news.java 中的 setXXX() 方法
            news1.setTitle(rs.getString("title"));
            news1.setContent(rs.getString("content"));
            news1.setAuthor(rs.getString("author"));
            news1.setSubmittime(rs.getString("submittime"));
❹           ret.add(news1);                                   //在 Collection 中添加 news 对象
        }
    }catch(Exception e){
        e.printStackTrace();                                  //捕捉异常
    }
❺   connsqlserver.close();                                    //关闭数据库连接
```

```
    return ret;                                          //返回 Collection 对象
}
```

代码贴士

❶ sql：查询新闻表的全部信息的 SQL 语句。

❷ connsqlserver.executeQuery(sql)：执行 SQL 语句。

❸ news1：实例化 news 类对象。

❹ ret.add(news1)：将当前对象添加到集合中。

❺ connsqlserver.close()：调用 close()方法关闭数据库连接。

（3）在新闻浏览页面中取得新闻相关信息。关键代码如下：

例程 50　代码位置：光盘\TM\09\net\WebRoot\houtai\newsbrowse.jsp

```
<%
Collection temp=sql.selectNewsAll();                     //取出新闻相关信息
Iterator it=temp.iterator();                             //将集合作为 iterator 函数显示
int count=0;
while(it.hasNext()){
    news newsl=(news)it.next();                          //将集合的值转换为 news.java 形式表示
    if(count%2==0)
    out.println("<tr bordercolor='#FFFFCC' bgcolor='#CCFFFF'>"); //双数行背景颜色为 #CCFFFF
    else
    out.println("<tr bgcolor='#CCCCFF'>");               //单数行背景颜色为 #CCCCFF
    out.println("<td><div align='center' class='zczi'>"+newsl.getTitle()+"</div></td>");
    //设置“修改”链接，转入新闻修改页面，参数为新闻 id
    out.println("<td><div align='center'><strong><a href='newsEdit.jsp?id="+newsl.getId()+"' class='zczi'>
修改</a></div></td>");
    out.println("<td><div align='center'><strong>
    //设置“删除”链接，弹出新闻删除页面，参数为新闻 id
    <a href='#'
onclick=window.open('newsDel.jsp?id="+newsl.getId()+"','newwindow','width=276,height=174,top=400,l
eft=500') class='zczi'>删除</a></div></td>");
    out.println("</tr>");
    count++;                                             //count 自增
}
%>
```

3．新闻删除

新闻删除功能主要实现删除新闻表中某一行中的新闻信息。使用 delete 语句，可以在新闻删除页面调用 JavaBean 的数据表删除方法。关键代码如下：

例程 51　代码位置：光盘\TM\09\net\WebRoot\houtai\newsDeltest.jsp

```
<%
    String id=(String)session.getAttribute("id");       //id 从 newsDel.jsp 中所得
    int i=0;
    i=sql.delNews(id);                                   //调用数据库删除方法
%>
```

在 JavaBean 中数据表删除方法的关键代码如下：

例程 52 代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
public int delNews(String id){
    int i=0;
    try{
        String sql="delete from tb_news where id='"+id+"'";          //删除新闻表中数据的 SQL 语句
        i=connsqlserver.executeUpdate(sql);                          //执行 SQL 语句
    }catch(Exception e){
            e.printStackTrace();                                     //捕捉异常
    }
    connsqlserver.close();                                           //关闭数据库连接
    return i;                                                        //将删除结果返回
}
```

4．新闻修改

新闻修改功能主要实现修改新闻表中的某一行。使用 update 语句，可以在新闻修改页面获取用户输入的修改后的数值，然后调用 JavaBean 的数据表修改方法。关键代码如下：

例程 53 代码位置：光盘\TM\09\net\WebRoot\houtai\newsEdit.jsp

```
<%
String id=news.getId().trim();                                       //获取表单中的值
String title=news.getTitle().trim();
String content=news.getContent().trim();
String author=news.getAuthor().trim();
int i=sql.updateNews(s.tranC(title),s.tranC(content),s.tranC(author),id);   //数据表修改 SQL 语句
if(i==1){                                                            //如果修改成功
%>
    <script language="javascript">
    alert("修改成功!");                                               //弹出成功对话框
    window.close();                                                  //关闭窗口
    </script>
<%
    response.sendRedirect("newsBrowse.jsp");                         //转入新闻浏览页面
}
 %>
```

在 JavaBean 中数据表修改方法的关键代码如下：

例程 54 代码位置：光盘\TM\09\net\src\com\wsy\selectsql.java

```
public int updateNews(String title,String content,String author,String id){
        int i=0;
        try{
            //数据表修改 SQL 语句
            String sql="UPDATE tb_news SET title = '"+title+"', content = '"+content+"', author =
'"+author+"' WHERE (id = '"+id+"')";                                 //更新数据表的 SQL 语句
            i=connsqlserver.executeUpdate(sql);                      //执行 SQL 语句
        }catch(Exception e){
```

```
            e.printStackTrace();
        }
        connsqlserver.close();                                    //关闭数据库连接
        return i;                                                 //将修改结果返回
    }
```

9.11　开发技巧与难点分析

9.11.1　页面弹出窗口控制

在实现新闻管理模块设计的新闻删除功能中，用到了页面弹出技术，这种弹出窗口在网站中经常会用到，如人尽皆知的弹出式网站公告或广告、打开新窗口显示公告的详细内容等。下面将通过具体实例介绍如何控制弹出窗口。

新闻删除功能中弹出页面主要是应用 JavaScript 脚本的 window 对象的 open()方法实现的。在 JavaScript 中，window 对象代表的是一个 Web 浏览器窗口或者窗口中的一个框架，可以使用 window 对象来实现对 Web 浏览器窗口或者窗口中的框架进行控制。window 对象的常用方法如表 9.3 所示。

表 9.3　window 对象的常用方法

方　法	描　述
alert()	弹出一个警告对话框
close()	关闭被引用的窗口
confirm()	弹出确认对话框
focus()	将被引用的窗口放在所有打开窗口的前面
open()	打开新浏览器窗口并且显示由 URL 或名字引用的文档，并设置创建窗口的属性
prompt()	弹出一个提示对话框
print()	打印窗口或框架中的内容
resizeTo(x,y)	将窗口的大小设置为（x,y），x、y 分别为宽度和高度
resizeBy(offsetx,offsety)	按照指定的位移量设置窗口的大小，当 offsetx、offsety 的值大于 0 时为扩大，小于 0 时为缩小

下面将对本实例中应用的 open()方法进行详细介绍。

window 对象的 open()方法用于打开浏览器窗口。使用 window 对象打开窗口的语法格式如下：

```
windowVar=window.open(url,windowname[,location]);
```

- ☑ windowVar：当前打开窗口的句柄。如果 open()方法执行成功，则 windowVar 的值为一个 window 对象的句柄，否则 windowVar 的值是一个空值。
- ☑ url：目标窗口的 URL。如果 URL 是一个空字符串，则浏览器将打开一个空白窗口，允许用 write()方法创建动态 HTML。
- ☑ windowname：window 对象的名称。

☑ location：对窗口属性进行设置，其可选参数如表 9.4 所示。

表 9.4 对窗口属性进行设置的可选参数

可选参数	说明	可选参数	说明
width	窗口的宽度	toolbar	浏览器工具条，包括后退及前进按钮等
height	窗口的高度	menubar	菜单条，一般包括文件、编辑及其他一些条目
scrollbars	是否显示滚动条	location	定位区，也叫地址栏，是可以输入 URL 的浏览器文本区
resizable	设定窗口大小是否固定	direction	更新信息的按钮

可以使用如下代码实现在网页中弹出新页面：

```
<script language="javascript">
<!--
window.open("ad.htm","advertise","width=620,height=130,top=10,left=20");
-->
</script>
```

9.11.2 FileUpload 组件获取表单中的值

通常添加<ENCTYPE="multipart/form-data">属性的表单使用 request.getParameter()方法获取不到表单中的值，FileUpload 组件提供了获取表单中值的方法。关键代码如下：

```
Iterator iter=fileitems.iterator();
while(iter.hasNext()){
    FileItem item = (FileItem) iter.next();
    if(item.isFormField()){                          //判断为表单中的非文件域的控件
        String name=item.getFieldName();             //表单中控件的名称
        String value=item.getString();               //表单中控件的值
    }
}
```

9.11.3 配置全局 Tomcat 连接池

在 Tomcat 5.5 版本中使用 9.3.5 节中连接池的配置方法会发生以下异常：

```
Cannot create JDBC driver of class '' for connect URL 'null', cause: No suitable driver
```

这时需要配置全局 Tomcat 连接池来解决以上问题。步骤如下：

（1）打开 Tomcat 所在目录下的 conf\server.xml 文件，找到<GlobalNamingResources>标签，在<GlobalNamingResources>与</ GlobalNamingResources>之间加入配置连接池的代码：

```
<Resource name="jdbc/ConnectionPool"                 <!--连接池名称-->
    auth="Container"
    type="javax.sql.DataSource"
    maxActive="20"                                   <!--连接池处于活动状态的数据库连接的最大数目-->
```

```
maxIdle="5"                                          <!--连接池处于空闲状态的数据库连接的最大数目-->
maxWait="10000"                                      <!--连接池中数据库连接处于空闲状态的最长时间-->
username="sa"                                        <!--数据库登录名-->
password="123456"                                    <!--数据库登录密码-->
<!--指定数据库的 JDBC 驱动程序-->
driverClassName="com.microsoft.jdbc.sqlserver.SQLServerDriver"
<!--指定连接数据库 URL -->
url="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=tb_net"/>
```

（2）打开 Tomcat 所在目录下的 conf\context.xml 文件，在<Context>与</Context>标签处添加如下代码：

```
<ResourceLink name="jdbc/ConnectionPool" global="jdbc/ConnectionPool" type="javax.sql.DataSource"/>
```

9.12　Proxool 连接池

除了在 Tomcat 配置连接池外，还有一个功能更为强大的连接池工具 Proxool。Porxool 为开源软件，这里笔者使用的是 proxool-0.9.0RC3.jar，读者可以在网络上自行下载最新版本。下面笔者介绍如何安装与使用 Proxool 连接池。

9.12.1　Proxool 安装

将 proxool-0.9.0RC3.zip 压缩文件解压后，将 proxool-0.9.0RC3\lib 目录下的 proxool-0.9.0RC3.jar 文件放入项目目录下的 WEB-INF/lib 目录中，这样即可使用 Proxool 连接池工具。

9.12.2　Proxool 使用

Proxool 的使用步骤如下：

（1）配置项目目录下的 WEB-INF/web.xml 文件，将如下代码放入<web-app></web-app>标签之间：

```
<!-- 使用 Proxool 配置连接池 -->
<servlet>
    <servlet-name>ServletConfigurator</servlet-name>
    <servlet-class>org.logicalcobwebs.proxool.configuration.ServletConfigurator</servlet-class>
        <init-param>
        <param-name>propertyFile</param-name>
        <!--指定 Proxool.properties 文件所在的位置-->
            <param-value>WEB-INF/classes/Proxool.properties</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup><!--当 Tomcat 启动时，Proxool 参数会自动设定加载在内存中-->
</servlet>
<servlet>
    <servlet-name>Admin</servlet-name>
```

```
    <servlet-class>org.logicalcobwebs.proxool.admin.servlet.AdminServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Admin</servlet-name>
    <url-pattern>/Admin</url-pattern>
</servlet-mapping>
```

（2）在项目目录下的 WEB-INF/classes 文件夹下创建名为 Proxool.properties 的文件。关键代码如下：

```
❶ jdbc-0.proxool.alias=net
❷ jdbc-0.proxool.driver-class=com.microsoft.jdbc.sqlserver.SQLServerDriver
❸ jdbc-0.proxool.driver-url=jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_net
❹ jdbc-0.user=sa
❺ jdbc-0.password=
❻ jdbc-0.proxool.maximum-connection-count=10
   jdbc-0.proxool.prototype-count=4
   jdbc-0.proxool.house-keeping-test-sql=select CURRENT_DATE
   jdbc-0.proxool.verbose=true
   jdbc-0.proxool.statistics=10s,1m,1d
   jdbc-0.proxool.statistics-log-level=DEBUG
```

代码贴士

❶ jdbc-0.proxool.alias：连接池的别名。

❷ jdbc-0.proxool.driver-class：连接池的驱动包名。

❸ jdbc-0.proxool.driver-url：数据库连接的 URL。

❹ jdbc-0.user：数据库连接的用户名。

❺ jdbc-0.password：数据库连接的密码。

❻ jdbc-0.proxool.maximum-connection-count：连接池中的最大连接数。

（3）创建测试 Proxool 连接池的 JSP 文件，名为 Proxool-test.jsp。关键代码如下：

```
try{
    con = DriverManager.getConnection("proxool.net");                    //取得数据库连接池
    stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
    String query = "SELECT * FROM tb_usertable";                         //查询 tb_usertable 表
    rs = stmt.executeQuery(query);                                       //执行数据库查询
    while(rs.next()) {
        String name = rs.getString(2);                                   //取用户名
        String password = rs.getString(3);                               //取密码
%>
姓名为：<%= name %><br>                                                  //打印数据库中字段的值
密码为：<%=password %>                                                   //打印数据库中字段的值
<%  }
        stmt.close();                                                    //关闭 statement
        con.close();                                                     //关闭 connection
    }
catch(SQLException sqle){
        out.println("sqle="+sqle);                                       //捕捉异常
```

```
}
finally{
        try {
                if(con != null){
                        con.close();                                    //关闭 connection
                }
        }
        catch(SQLException sqle){
                out.println("sqle="+sqle);                              //捕捉异常
        }
}
```

运行 http://localhost:8080/net/Proxool-test.jsp 文件，运行结果如图 9.24 所示。

图 9.24　Proxool-test.jsp 运行结果

9.13　本 章 小 结

本章运用软件工程的设计思想，开发了一个完整的企业门户网站。在开发过程中，采用了 JavaBean+JSP 开发模式，使整个网站的设计思路更为清晰。通过本章的学习，读者不仅可以了解一般网站的开发流程，而且还对 JavaBean+JSP 开发模式有了比较深入的了解，为以后实际项目开发奠定了坚实的基础。

第10章

棋牌游戏系统之网络五子棋

（Swing+ Socket 实现）

五子棋是起源于中国古代的传统黑白棋种之一。五子棋不仅能增强思维能力，提高智力，而且富含哲理，有助于修身养性。五子棋既有现代休闲的明显特征“短、平、快”，又有古典哲学的高深学问“阴阳易理”；既具有简单易学的特性，为人们所喜爱，又有深奥的技巧和高水平的国际性比赛。五子棋文化源远流长，具有东方的神秘和西方的直观；既有“场”的概念，亦有“点”的连接。五子棋起源于中国古代，发展于日本，风靡于欧洲，可以说五子棋是中西方文化的交流点，是古今哲学的结晶。在本章中，笔者将介绍曾经为 XXX 公司开发的一个网络五子棋，它是该公司游戏平台的棋牌游戏系统的一部分。

通过阅读本章，可以学习到：

- 如何进行项目的可行性分析
- 如何进行系统设计
- 绘制半透明的登录界面
- 游戏记录回放
- 绘制可以调整大小的五子棋棋盘
- 网络连接状态检测

10.1　开 发 背 景

网络游戏是目前最大的游戏市场，各类游戏程序层出不穷，其中×××游戏大厅是由×××有限公司主导开发的一款休闲类游戏社区，其中包括棋牌、网游、对战平台等很多游戏种类，而且日后可以不断扩充。现阶段任务是完善棋牌类游戏的开发工作，开发顺序以五子棋、象棋、飞行棋……的排序为依据。

五子棋，相信是每个人都会的游戏，当游戏的一方构成 5 个连续的棋子，无论是水平方向、垂直方向，还是斜对角线方向，就表示获胜了。在游戏开发过程中，有很多功能需要严格测试，避免出现缺陷。为保持测试的方便性，项目要求具有独立性，可以脱离服务器进行测试，然后由×××有限公司技术支持部门对核心模块进行提取，整合到平台中。

10.2　需 求 分 析

通过与×××有限公司的沟通和需求分析，要求开发的五子棋游戏具有以下功能：

- ☑ 系统操作简单，界面友好。
- ☑ 界面灵活缩放，可随窗体大小绘制游戏界面。
- ☑ 实现多种界面特效，使界面美观绚丽。
- ☑ 支持游戏悔棋与游戏回放功能。
- ☑ 提供游戏背景更换功能，避免视疲劳。
- ☑ 支持聊天功能，增强游戏沟通能力。

10.3　系 统 设 计

10.3.1　系统目标

根据需求分析的描述以及与用户的沟通，现制定系统实现目标如下：

- ☑ 界面设计简洁、美观、支持背景更换，要吸引游戏者的眼球。
- ☑ 提供游戏棋局的回放功能，让用户找出失败或胜利的关键。
- ☑ 人性化的悔棋功能，在双方同意的情况下可以悔棋。
- ☑ 提供聊天功能，让游戏者保持沟通。
- ☑ 支持悔棋与认输，给予游戏者放弃的权利，不浪费游戏时间。
- ☑ 支持和棋，友谊第一，比赛第二。
- ☑ 对导致游戏结束的 5 颗棋子做明显标注。

10.3.2 系统功能结构

网络五子棋游戏项目包括聊天室、游戏操作、游戏回放、更换背景 4 大部分。其中聊天室与游戏操作部分又可细分为几个子功能。系统结构如图 10.1 所示。

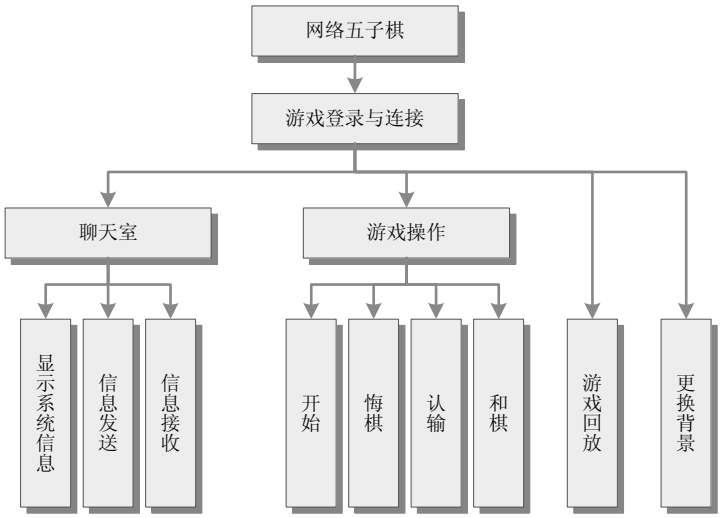


图 10.1 网络五子棋功能结构

10.3.3 系统流程图

网络五子棋的系统流程如图 10.2 所示。

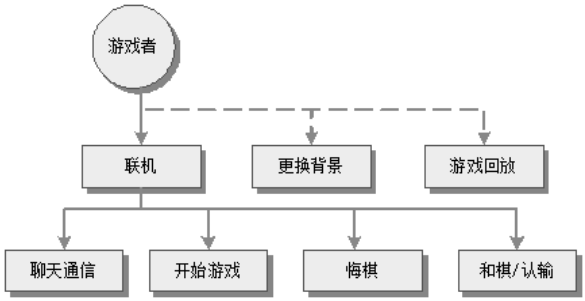


图 10.2 网络五子棋系统流程图

10.3.4 构建开发环境

在开发网络五子棋系统时，使用了下面的软件环境。

- ☑ 操作系统：Windows Server 2003（SP1）以上。
- ☑ Java 开发包：JDK 1.6 以上。
- ☑ 开发工具：Eclipse 3.5 以上。
- ☑ 数据库：SQL Server 2000。
- ☑ 分辨率：最佳效果为 1024×768 像素。

SP（Service Pack）为 Windows 操作系统补丁。

10.3.5　系统预览

程序运行以后，首先显示登录界面，这个登录界面使用半透明效果将主窗体遮罩，然后显示登录界面，用户必须输入自己的昵称和对方主机的 IP 地址才能登录。程序运行效果如图 10.3 所示。

和对方建立网络连接后，会进入游戏主窗体。单击“开始”按钮将开始进行游戏。当自己头像下方有一盒棋子时，就轮到自己下棋，棋子的颜色和自己头像下方棋盒里的棋子颜色相同。游戏主窗体的运行效果如图 10.4 所示。

图 10.3　登录联机的程序界面（光盘\TM\10\Gobang\src\com\lzw\gobang\LoginPanel.java）

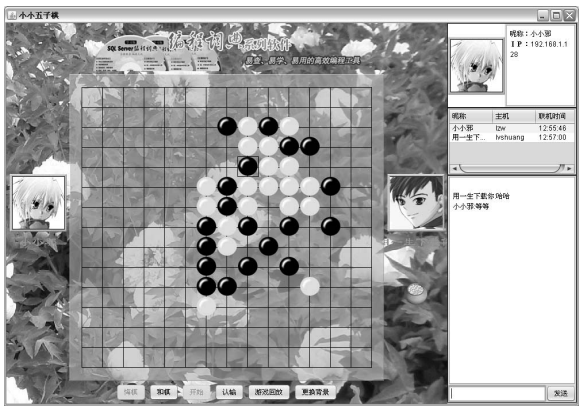


图 10.4　游戏主窗体效果（光盘\TM\10\Gobang\src\com\lzw\gobang\MainFrame.java）

当游戏的一方胜利时，程序界面将提示“对方胜利”，并且把对方的 5 颗连线棋子用星号标注，并禁止棋盘的落子行为。程序界面如图 10.5 所示。

当自己一方有 5 颗棋子连成一线时，将提示“你胜利了”的信息，并且将自己一方相连的 5 颗棋子用星号标注，效果如图 10.6 所示。

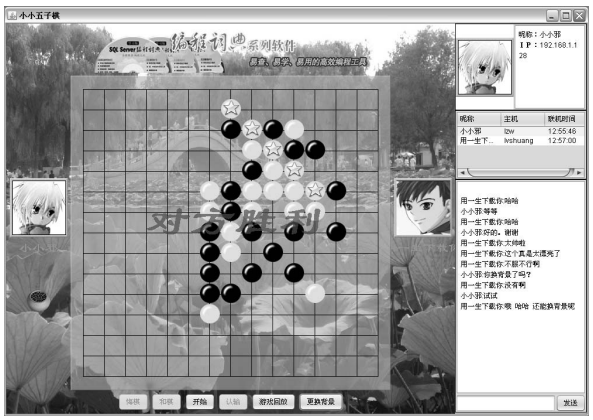


图 10.5　对方胜利后的效果（光盘\TM\10\Gobang\src\com\lzw\gobang\MainFrame.java）

图 10.6　自己胜利后的界面（光盘\TM\10\Gobang\src\com\lzw\gobang\MainFrame.java）

游戏的开始、悔棋、和棋、游戏回放等动作，由棋盘下方的控制面板组成，该面板还包含一个“更换背景”按钮可以更换程序界面的背景图片。控制面板如图 10.7 所示。

10.3.6 文件夹组织结构

在进行系统开发前，需要规划文件夹组织结构，也就是说，建立多个文件夹，对各个功能模块进行划分，实现统一管理。这样做的好处是易于开发、管理和维护。本系统的文件夹组织结构如图 10.8 所示。

图 10.7 控制按钮界面（光盘\TM\10\Gobang\src\com\lzw\gobang\ChessPanel.java）

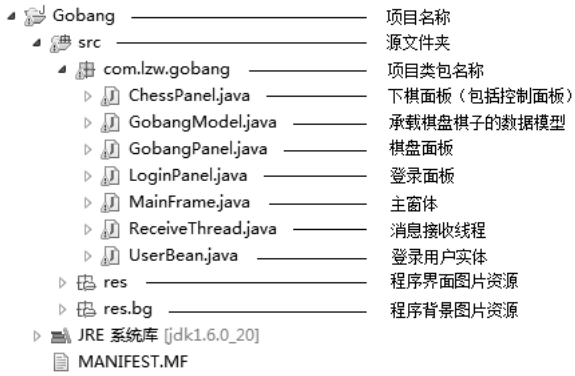


图 10.8 文件夹组织结构图

10.4 公共模块设计

公共模块的设计是软件开发的一个重要组成部分，它既起到了代码重用的作用，又起到了规范代码结构的作用，尤其在团队开发的情况下，是解决重复编码的最好方法，这样对软件的后期维护也起到了积极的作用。

10.4.1 绑定属性的 JavaBean

本项目定义了棋盘模型类 GobangModel，它是一个 JavaBean，用于记录棋盘的当前棋子的布局，它使用一个二维数组保存所有棋子。

这个记录棋盘的棋子数据的 JavaBean 将棋子数组定义为绑定属性，当该属性被修改时会自动产生属性变更事件，并通知所有监听该属性的监听器。在棋盘类中就定义了一个监视该属性的监听器，它在棋盘模型的数据发生改变时立刻更新棋盘界面。

要实现 JavaBean 的绑定属性，必须实现以下两个机制。

☑ 产生 PropertyChange 事件

无论任何情况，只要 JavaBean 中的绑定属性发生了变化，该 JavaBean 就必须发送一个 PropertyChange 属性改变事件给所有已经注册的事件监听器，棋盘模型 JavaBean 在设置棋盘数组属性的 setChessmanArray()方法中产生了该事件。关键代码如下：

例程 01　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\GobangModel.java

```
private PropertyChangeSupport propertySupport;                          //定义属性工具类
private static GobangModel model;                                       //定义自身的变量
private byte[][] chessmanArray = new byte[15][15];                      //定义棋子数组
public static final String PROP_CHESSMANARRAY = "chessmanArray";  //定义属性名称
…//省略部分代码
/**
 * 设置棋子数组的方法
* @param chessmanArray
 *                - 一个代表棋盘棋子的二维数组
 */
public void setChessmanArray(byte[][] chessmanArray) {
      this.chessmanArray = chessmanArray;
      propertySupport.firePropertyChange(PROP_CHESSMANARRAY, null,
                chessmanArray);                                         //通知所有已注册监听器属性被更新
}
```

☑　实现事件监听器的注册与注销

对棋盘数据模型 JavaBean 感兴趣的监听器必须通过该 JavaBean 提供的方法进行注册或注销，这两个方法分别是 addPropertyChangeListener()和 removePropertyChangeListener()。只有注册到该 JavaBean 的事件监听器才能监听 JavaBean 属性的改变事件。棋盘数据模型对这两个方法的实现代码如下：

例程 02　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\GobangModel.java

```
/**
 * 添加事件监听器的方法
 *
 * @param listener
 *                - 事件监听器
 */
public void addPropertyChangeListener(PropertyChangeListener listener) {
      propertySupport.addPropertyChangeListener(listener);              //添加事件监听器
}

/**
 * 移除事件监听器的方法
 *
 * @param listener
 *                - 事件监听器
 */
public void removePropertyChangeListener(PropertyChangeListener listener) {
      propertySupport.removePropertyChangeListener(listener);           //移除事件监听器
}
```

10.4.2　在棋盘中绘制棋子

在设计网络五子棋时，需要在棋盘中绘制棋子，并且在窗口更新时保证棋子仍然在棋盘上。笔者采用的方式是定义一个二维数组，数组的大小与棋盘中表格的行和列相对应，描述棋盘中可以放置棋

子的所有点。绘制棋子的界面效果如图 10.9 所示。

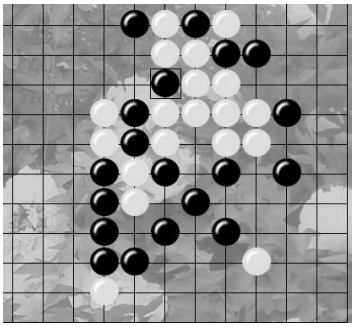


图 10.9　绘制棋子的界面效果

关键代码如下：

例程 03　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\GobangPanel.java

```
byte[][] chessmanArray = gobangModel1.getChessmanArrayCopy();        //棋盘二维数组
for (int i = 0; i < chessmanArray.length; i++) {                     //遍历二维数组元素绘制棋子
    for (int j = 0; j < chessmanArray[i].length; j++) {
        byte chessman = chessmanArray[i][j];
        int x = i * chessWidth;
        int y = j * chessHeight;
        if (chessman != 0)
            System.out.println("chess is:" + chessman);
        if (chessman == WHITE_CHESSMAN) {                            //绘制白棋
            g.drawImage(white_chessman_img, x, y, chessWidth,
                    chessHeight, this);
        } else if (chessman == BLACK_CHESSMAN) {                     //绘制黑棋
            g.drawImage(black_chessman_img, x, y, chessWidth,
                    chessHeight, this);
        } else if (chessman == (WHITE_CHESSMAN ^ 3)) {               //绘制最近的白棋落子
            g.drawImage(white_chessman_img, x, y, chessWidth,
                    chessHeight, this);
            g.drawRect(x, y, chessWidth, chessHeight);
        } else if (chessman == (BLACK_CHESSMAN ^ 3)) {               //绘制最近的黑棋落子
            g.drawImage(black_chessman_img, x, y, chessWidth,
                    chessHeight, this);
            g.drawRect(x, y, chessWidth, chessHeight);
        } else if (chessman == ((byte) (WHITE_CHESSMAN ^ 8))) {      //绘制导致胜利的连线白棋
            g.drawImage(white_chessman_img, x, y, chessWidth,
                    chessHeight, this);
            g.drawImage(rightTop_img, x, y, chessWidth, chessHeight,
                    this);
        } else if (chessman == (BLACK_CHESSMAN ^ 8)) {               //绘制导致胜利的连线黑棋
            g.drawImage(black_chessman_img, x, y, chessWidth,
                    chessHeight, this);
            g.drawImage(rightTop_img, x, y, chessWidth, chessHeight,
```

```
                this);
            }
        }
}
```

10.4.3　实现动态调整棋盘大小

在设计网络五子棋时，为了突出游戏的特点，允许用户在游戏进行的过程中调整窗口的大小，效果如图 10.10 和图 10.11 所示。

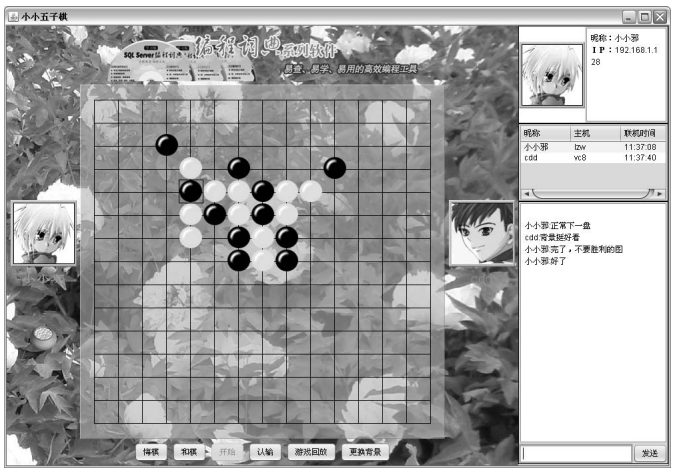


图 10.10　下棋窗口

图 10.11　棋盘缩放

实现该功能的难点在于窗口调整大小后，棋盘和棋子的大小需要调整，棋盘表格的大小需要调整，棋盘中当前棋子的位置需要调整。笔者采用的方式是根据棋盘面板的宽度和高度计算棋格和棋子大小。然后使用计算出的棋格宽度和高度绘制棋盘、使用计算出的棋子高度与宽度绘制指定大小的棋子。绘制棋盘的关键代码如下：

例程 04　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\GobangPanel.java

```
int w = getWidth();                                          //棋盘宽度
int h = getHeight();                                         //棋盘高度
int chessW = w / 15, chessH = h / 15;                        //棋子宽度和高度
int left = chessW / 2 + (w % 15) / 2;                        //棋盘左边界
int right = left + chessW * 14;                              //棋盘右边界
int top = chessH / 2 + (h % 15) / 2;                         //棋盘上边界
int bottom = top + chessH * 14;                              //棋盘下边界
for (int i = 0; i < 15; i++) {
    //画每条横线
    g.drawLine(left, top + (i * chessH), right, top + (i * chessH));
}
for (int i = 0; i < 15; i++) {
    //画每条竖线
    g.drawLine(left + (i * chessW), top, left + (i * chessW), bottom);
}
```

10.4.4 游戏悔棋

为了增加网络五子棋的交互、灵活性，在本程序中设计了悔棋功能。当用户想要悔棋时，需要向对方发送悔棋请求，如果对方同意悔棋，则双方都进行悔棋操作，如图 10.12 所示。

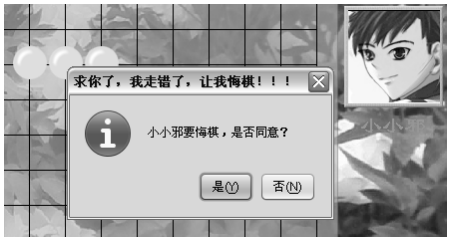


图 10.12 悔棋时的确认界面

五子棋游戏模块设计了一个存储下棋步骤的双向队列，要实现悔棋功能，需要从该队列中弹出两步落子动作，其中包括自己的下棋步骤和对方的下棋步骤。关键代码如下：

例程 05 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * 悔棋的业务处理方法
 */
public synchronized void repentOperation() {

    Deque<byte[][]> chessQueue = gobangPanel1.getChessQueue();          //获取下棋队列
    if (chessQueue.isEmpty()) {
        return;
    }
    for (int i = 0; i < 2 && !chessQueue.isEmpty(); i++) {              //获取上两次走棋的棋谱
        byte[][] pop = chessQueue.pop();                                //废弃走棋步骤
    }
    if (chessQueue.size() < 1) {
        chessQueue.push(new byte[15][15]);
    }
    byte[][] pop = chessQueue.peek();
    GobangModel.getInstance().updateChessmanArray(pop);                 //更新棋盘的棋子布局
    repaint();
}
```

10.4.5 游戏回放

为了让游戏的双方了解下棋的整个过程，网络五子棋模块设计了游戏回放功能。当游戏结束时，用户可以通过游戏回放了解整个下棋的过程，分析对方下棋的思路，总结成功与失败的经验。

五子棋游戏设计了一个存储下棋步骤的双向队列，要实现游戏回放功能，只需要把队列中记录的下棋步骤（每一个队列元素保存了下棋的每一步的棋谱）从头演示一边即可。但是要注意，每个步骤需要停顿一秒，给玩家一个分析的时间。关键代码如下：

例程 06 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * 游戏回放按钮的事件处理方法
 *
```

```
     * @param evt - 事件对象
     */
    private void backplayToggleButtonActionPerformed(
            java.awt.event.ActionEvent evt) {
        //如果游戏进行中，提示用户游戏结束后再观看游戏回放
        if (gobangPanel1.isStart()) {
            JOptionPane.showMessageDialog(this, "请在游戏结束后，观看游戏回放。");
            backplayToggleButton.setSelected(false);
            return;
        }
        if (!backplayToggleButton.isSelected()) {
            backplayToggleButton.setText("游戏回放");
        } else {
            backplayToggleButton.setText("终止回放");
            new Thread() {                                          //开启新的线程播放游戏记录
                public void run() {
                    Object[] toArray = gobangPanel1.getOldRec();
                    if (toArray == null) {
                        JOptionPane.showMessageDialog(ChessPanel.this,
                                "没有游戏记录", "游戏回放", JOptionPane.WARNING_MESSAGE);
                        backplayToggleButton.setText("游戏回放");
                        backplayToggleButton.setSelected(false);
                        return;
                    }
                    //清除界面的结局文字，包括对方胜利、你胜利了、此战平局
                    gobangPanel1.setTowardsWin(false);
                    gobangPanel1.setWin(false);
                    gobangPanel1.setDraw(false);
                    for (int i = toArray.length - 1; !gobangPanel1.isStart()
                            && i >= 0; i--) {
                        try {
                            Thread.sleep(1000);                     //线程休眠 1 秒
                        } catch (InterruptedException ex) {
                            Logger.getLogger(ChessPanel.class.getName()).log(
                                    Level.SEVERE, null, ex);
                        }
                        GobangModel.getInstance().updateChessmanArray(
                                (byte[][]) toArray[i]);             //根据游戏记录更换每一步游戏的棋谱
                        gobangPanel1.repaint();                     //重绘棋盘
                    }
                    backplayToggleButton.setSelected(false);
                    backplayToggleButton.setText("游戏回放");
                }
            }.start();
        }
    }
```

10.5 实现登录界面

主窗体的登录界面是五子棋模块的开始，它主要不是验证用户名与密码，而是定义自己游戏时的昵称和对方主机的 IP 地址。昵称将显示在游戏界面中，包括自己的和对家的昵称。IP 地址是确定对家的唯一条件，只有确定双方的 IP 地址，并且双方都运行了五子棋模块后，才能进行互联。

图 10.13 登录界面的背景半透明效果

登录界面的登录面板实现了透明的效果，并且将登录面板下的主窗体用半透明遮罩效果变暗，而登录界面正常显示，这样，用户的注意力就会放在登录界面上。登录面板的运行效果如图 10.13 所示。

实现登录界面的关键技术，使用了 GlassPane 面板，它位于窗体的最顶层，Swing 默认该面板为隐藏模式。本程序继承 JPanel 类编写了登录面板，其中包含登录信息的文本框和“登录”按钮等信息，然后调用 JFrame 窗体的 setGlassPane()方法将该面板设置为 GlassPanel 玻璃面板。

实现登录界面的关键步骤如下：

（1）继承 JPanel 类编写登录面板，重写 paintComponent()方法，在方法中获取 Java2D 的绘图对象，备份绘图的合成模式，然后设置新的 80%透明的合成模式，并使用矩形填充整个登录面板。关键代码如下：

例程 07 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\LoginPanel.java

```
/**
 * 登录面板
 * @author Li Zhong Wei
 */
public class LoginPanel extends javax.swing.JPanel {
    private Socket socket;
    private UserBean user;
    protected boolean linked;
    /**
     * 构造方法
     */
    public LoginPanel() {
        initComponents();                                    //调用初始化界面的方法
    }
    /**
     * 绘制组件界面的方法
     * @see javax.swing.JComponent#paintComponent(java.awt.Graphics)
     */
    @Override
```

```
    protected void paintComponent(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;                          //获取 2D 绘图上下文
        Composite composite = g2.getComposite();                 //备份合成模式
        g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER,
                0.8f));                                          //设置绘图使用透明合成规则
        g2.fillRect(0, 0, getWidth(), getHeight());              //使用当前颜色填充矩形空间
        g2.setComposite(composite);                              //恢复原有合成模式
        super.paintComponent(g2);                                //执行超类的组件绘制方法
    }
…//省略其他代码
}
```

（2）编写“登录”按钮的事件处理方法，在该方法中接收用户的昵称和对方主机信息，使用昵称和本机 IP 地址创建本地用户对象发送给对方主机。使用对方主机 IP 地址创建 Socket 连接对象，实现互联操作。关键代码如下：

例程 08　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\LoginPanel.java

```
/**
 * “登录”按钮的事件处理方法
 *
 * @param evt
 *              - 按钮的事件对象
 */
private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        //获取主窗体的实例对象
        MainFrame mainFrame = (MainFrame) getParent().getParent();
        String name = nameTextField.getText();                   //获取用户昵称
        if (name.trim().isEmpty()) {
            JOptionPane.showMessageDialog(this, "请输入昵称");
            return;
        }
        String ipText = ipTextField.getText();                   //获取对方 IP 地址
        ipTextField.setEditable(true);
        InetAddress ip = InetAddress.getByName(ipText);
        socket = new Socket(ip, 9528);                           //创建 Socket 连接对方主机
        if (socket.isConnected()) {                              //如果连接成功
            user = new UserBean();                               //创建用户对象
            //获取当前时间对象
            Time time = new Time(System.currentTimeMillis());
            user.setName(name);                                  //初始化用户昵称
            user.setHost(InetAddress.getLocalHost());            //初始化用户 IP
            user.setTime(time);                                  //初始化用户登录时间
            socket.setOOBInline(true);                           //启用紧急数据的接收
            mainFrame.setSocket(socket);                         //设置主窗体的 Socket 连接对象
            mainFrame.setUser(user);                             //添加本地用户对象到主窗体对象
            mainFrame.send(user);                                //发送本地用户对象到对方主机
```

```
                setVisible(false);                                    //隐藏登录窗体
            }
        } catch (UnknownHostException ex) {
            Logger.getLogger(LoginPanel.class.getName()).log(Level.SEVERE,
                    null, ex);
            JOptionPane.showMessageDialog(this, "输入的 IP 不正确");
        } catch (IOException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "对方主机无法连接");
        }
}
```

（3）在主窗体创建登录面板的实例对象，调用 setGlassPane()方法，设置登录面板为主窗体的 GlassPane 面板（即玻璃面板），调用登录面板的 setVisible()方法显示登录界面。关键代码如下：

例程 09 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\MainFrame.java

```
loginPanel1 = new com.lzw.gobang.LoginPanel();                        //创建登录面板的实例对象
/**
 * 主窗体的构造方法
 */
public MainFrame() {
    initComponents();                                                 //初始化窗体界面
    setGlassPane(loginPanel1);                                        //设置登录面板为玻璃面板
    loginPanel1.setVisible(true);                                     //显示登录面板
}
```

10.6 编写游戏主窗体

游戏的主窗体包括下棋面板、用户信息面板、用户列表和聊天面板，界面的运行效果如图 10.14 所示。

除了下棋面板以外，用户信息面板、用户列表和聊天面板都由游戏主窗体实现。本节将主要介绍这些内容，至于下棋面板将在后面的章节介绍。

实现游戏主窗体的关键步骤如下：

（1）编写 setSocket()方法，该方法曾在登录面板中调用，用于设置联机的 Socket 对象，但是该方法同时也初始化了 object 对象输出流。它用于发送字符串对象或其他对象到对方主机。关键代码如下：

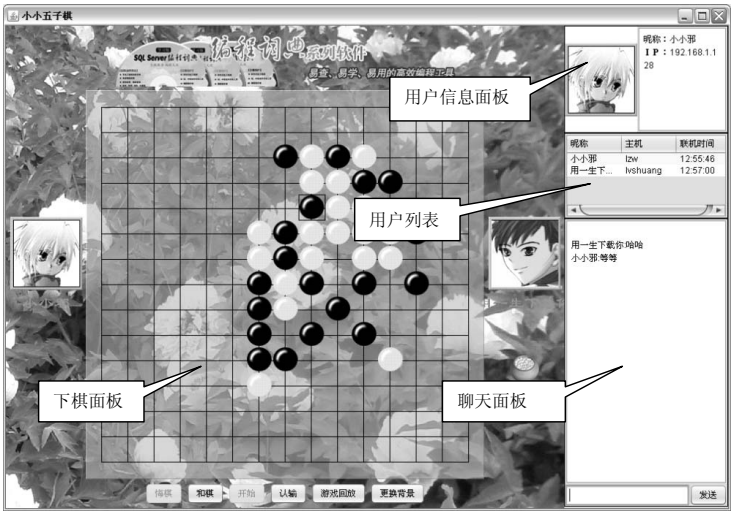


图 10.14 游戏主窗体界面

例程 10　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\MainFrame.java

```
/**
 * 设置 Socket 连接和初始化对象输出流的方法
 * @param chatSocketArg - Socket 对象
 */
public void setSocket(Socket chatSocketArg) {
    try {
        socket = chatSocketArg;
        OutputStream os = socket.getOutputStream();        //获取 Socket 的输出流
        objout = new ObjectOutputStream(os);               //创建对象输出流
    } catch (IOException ex) {
        Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null,
                ex);
    }
}
```

（2）编写 send()方法，该方法用于发送信息到对方主机，这个信息可以是文本字符串对象也可以是其他类型的对象。在主类中调用该方法发送聊天的文本字符串信息，但是在其他面板类中，调用该方法发送用户对象、游戏指令、棋盘信息等内容，所以该方法的参数是 Object 类型。关键代码如下：

例程 11　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\MainFrame.java

```
/**
 * 向对方发送信息的方法
 * @param message - 要发送的文本或其他类型的对象
 */
public void send(Object message) {
    try {
        objout.writeObject(message);                       //向对象输出流添加对象
        objout.flush();
    } catch (IOException ex) {
        Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null,
                ex);
    }
}
```

（3）编写聊天面板的“发送”按钮的事件处理方法，该方法将获取用户输入的聊天字符串，并把该字符串的内容追加到聊天记录的文本区域组件中，然后调用 send()方法将聊天信息发送给对方。关键代码如下：

例程 12　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\MainFrame.java

```
/**
 * 聊天面板的发送按钮事件处理方法
 * @param evt - 事件对象
 */
private void sendButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String message = (String) chatTextField.getText();          //获取文本信息
```

```
        if (message == null || message.isEmpty()) {
            return;
        }
        chatTextField.setText("");                                  //清空文本框内容
        appendMessage(user.getName() + ":" + message);              //将发送的信息添加到聊天记录
        send(message);                                              //发送信息
    }
```

（4）编写 appendMessage()方法，该方法用于向聊天面板的文本区域组件追加换行的聊天信息。关键代码如下：

例程 13 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\MainFrame.java

```
/**
 * 添加聊天信息的方法
 * @param message - 聊天信息文本
 */
protected void appendMessage(final String message) {
    Runnable runnable = new Runnable() {                            //创建线程对象
        @Override
        public void run() {
            chatArea.append("\n" + message);                        //向聊天文本区域组件追加换行文本
        }
    };
    if (SwingUtilities.isEventDispatchThread()) {
        runnable.run();                                             //在事件队列线程中执行该线程对象
    } else {
        SwingUtilities.invokeLater(runnable);
    }
}
```

（5）编写启动 Socket 服务器的 startServer()方法，该方法将创建 ServerSocket 类的实例对象，该对象接收远程用户的连接。关键代码如下：

例程 14 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\MainFrame.java

```
/**
 * 启动 Socket 服务器
 */
public void startServer() {
    try {
        //创建 Socket 服务器对象
        final ServerSocket chatSocketServer = new ServerSocket(9528);
        //创建接收信息的线程
        new ReceiveThread(chatSocketServer, this).start();
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(this, "本程序禁止重复运行，只能同时存在一个实例。",
                "你敢重复运行？", JOptionPane.ERROR_MESSAGE);
        System.exit(0);
        Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null,
```

```
                ex);
        }
}
```

（6）编写 setUser()方法，该方法用于设置本地用户信息，包括游戏下棋面板的本地用户昵称、用户列表中的本地用户和用户信息面板的内容等。关键代码如下：

例程 15　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\MainFrame.java

```
/**
 * 设置用户信息的方法
 * @param user - 本地用户对象
 */
public void setUser(UserBean user) {
    this.user = user;
    //向用户信息面板添加昵称
    userInfoTextArea.setText("昵称：" + user.getName() + "\n");
    //添加 IP 信息
    userInfoTextArea.append("ＩＰ：" + user.getHost().getHostAddress() + "\n");
    //获取用户信息表格组件的数据模型对象
    DefaultTableModel model = (DefaultTableModel) userInfoTable.getModel();
    Vector dataVector = model.getDataVector();
    Vector row = new Vector();                          //使用用户信息创建单行数据的向量
    row.add(user.getName());
    row.add(user.getHost().getHostName());
    row.add(user.getTime());
    if (!dataVector.contains(row)) {
        model.getDataVector().add(row);                 //把用户信息添加到表格组件中
    }
    //设置本地用户的昵称
    chessPanel1.leftInfoLabel.setText(user.getName());
    userInfoTable.revalidate();
}
```

（7）编写 setTowardsUser()方法，该方法和 setUser()方法功能类似，但是它用于设置对方信息。关键代码如下：

例程 16　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\MainFrame.java

```
/**
 * 设置对方用户信息的方法
 * @param user - 对方通过网络发送来的用户对象
 */
public void setTowardsUser(UserBean user) {
    this.towardsUser = user;                            //对方用户对象
    //获取用户信息列表的表格数据模型
    DefaultTableModel model = (DefaultTableModel) userInfoTable.getModel();
    Vector row = new Vector();                          //创建承载表格单行数据的向量集合对象
    row.add(towardsUser.getName());                     //添加用户姓名
    row.add(towardsUser.getHost().getHostName());       //添加主机名称
```

```
        row.add(towardsUser.getTime());                    //添加用户登录时间
        Vector dataVector = model.getDataVector();
        if (!dataVector.contains(row)) {
            model.getDataVector().add(row);                //添加用户信息到表格中
        }
        //设置对方用户头像的昵称
        chessPanel1.rightInfoLabel.setText(towardsUser.getName());
        userInfoTable.revalidate();
}
```

10.7 编写下棋面板

下棋面板用于游戏的控制，包括游戏的开始、悔棋、和棋、认输、清屏、更改游戏背景图等，它还负责游戏开始时，为上方玩家分配棋子颜色等业务。程序界面如图 10.15 所示。

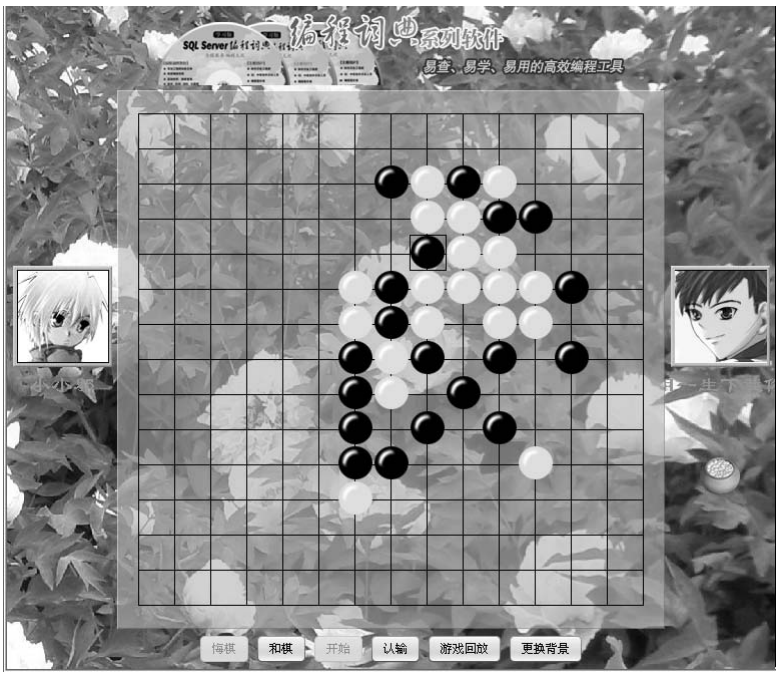


图 10.15 下棋面板的界面

实现下棋面板的关键步骤如下：

（1）继承 JPanel 自定义下棋面板类，在该类中定义各种指令的编码，这些编码将用于游戏的网络互动，它们包括悔棋命令、和棋命令、认输命令、开始命令、胜利代码等。关键代码如下：

例程 17 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * 下棋面板
 * @author Li Zhong Wei
```

```
 */
public class ChessPanel extends javax.swing.JPanel {
    static ImageIcon WHITE_CHESS_ICON;
    static ImageIcon BLACK_CHESS_ICON;
    final static int OPRATION_REPENT = 0xEF;                //悔棋命令
    final static int OPRATION_NODE_REPENT = 0xCF;           //接受悔棋命令
    final static int OPRATION_DRAW = 0xFE;                  //和棋命令
    final static int OPRATION_NODE_DRAW = 0xEE;             //接受和棋命令
    final static int OPRATION_START = 0xFd;                 //开始命令
    final static int OPRATION_ALL_START = 0xEd;             //接受开始命令
    final static int OPRATION_GIVEUP = 0xFc;                //认输命令
    final static int WIN = 88;                              //胜利代码
…//省略部分代码
}
```

（2）在构造方法中初始化双方棋盒的图片和背景图片对象。另外，该构造方法负责调用初始化界面的 initComponents()方法完成界面布局。关键代码如下：

例程 18　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * 下棋面板的构造方法
 */
public ChessPanel() {
    WHITE_CHESS_ICON = new javax.swing.ImageIcon(getClass().getResource(
            "/res/whiteChess.png"));                        //初始化白棋棋盒图片
    BLACK_CHESS_ICON = new javax.swing.ImageIcon(getClass().getResource(
            "/res/blackChess.png"));                        //初始化黑棋棋盒图片
    URL url = getClass().getResource("/res/bg/1.jpg");
    backImg = new ImageIcon(url).getImage();                //初始化背景图片
    initComponents();                                       //调用初始化界面的方法
}
```

（3）重写父类的 paintComponent()方法，在方法中绘制游戏的背景图片，从而定义新的组件界面。关键代码如下：

例程 19　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * 重写 paintComponent 方法，绘制背景图片
 * @see javax.swing.JComponent#paintComponent(java.awt.Graphics)
 */
@Override
protected void paintComponent(Graphics g) {
    //绘制背景图片
    g.drawImage(backImg, 0, 0, getWidth(), getHeight(), null);
}
```

（4）编写设置棋盒颜色的 setChessColor()方法，该方法用于设置自己和对方头像下方的棋盒颜色。关键代码如下：

例程 20 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * 设置棋子颜色的方法，以棋盒颜色为主
 * @param color - 指定颜色的棋盒图片
 */
public void setChessColor(ImageIcon color) {
    myChessColorLabel.setIcon(color);                          //设置本地用户的棋盒图标
    if (color.equals(WHITE_CHESS_ICON)) {                      //设置白棋
        gobangPanel1.setMyColor(GobangPanel.WHITE_CHESSMAN);
        towardsChessColorLabel.setIcon(BLACK_CHESS_ICON);
    } else if (color.equals(BLACK_CHESS_ICON)) {               //设置黑棋
        gobangPanel1.setMyColor(GobangPanel.BLACK_CHESSMAN);
        towardsChessColorLabel.setIcon(WHITE_CHESS_ICON);
    }
    revalidate();
}
```

（5）编写 setTurn()方法，该方法用于设置自己走棋的权限，如果没有走棋权限，不能在棋盘上任何位置落子。关键代码如下：

例程 21 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * 设置轮回状态的方法
 *
 * @param turn
 *                - 是否获得走棋权利
 */
public void setTurn(boolean turn) {
    if (turn) {                                                //如果获得走棋权利
        myChessColorLabel.setVisible(true);                    //显示棋盒
        towardsChessColorLabel.setVisible(false);              //隐藏对方棋盒
    } else {                                                   //否则
        myChessColorLabel.setVisible(false);                   //隐藏自己的棋盒
        towardsChessColorLabel.setVisible(true);               //显示对方的棋盒
    }
}
```

（6）编写 repentOperation()方法，该方法用于执行悔棋动作，它首先从记录下棋动作的队列中取出两个走棋的记录（包含对方走棋和自己走棋），然后使用队列顶层的当前棋局更新棋盘上的棋子实现悔棋动作。关键代码如下：

例程 22 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * 悔棋的业务处理方法
 */
public synchronized void repentOperation() {
    //获取下棋队列
```

```
    Deque<byte[][]> chessQueue = gobangPanel1.getChessQueue();
    if (chessQueue.isEmpty()) {
        return;
    }
    //获取上次走棋的棋谱
    for (int i = 0; i < 2 && !chessQueue.isEmpty(); i++) {
        byte[][] pop = chessQueue.pop();
    }
    if (chessQueue.size() < 1) {
        chessQueue.push(new byte[15][15]);
    }
    byte[][] pop = chessQueue.peek();
    GobangModel.getInstance().updateChessmanArray(pop);        //更新棋盘的棋子布局
    repaint();
}
```

（7）编写 fenqi()方法，该方法用于分配双方玩家棋子的颜色，区分规则是先开始的玩家使用白色棋子。关键代码如下：

例程 23　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * 为双方玩家分配棋子的方法
 */
private void fenqi() {
    MainFrame frame = (MainFrame) getRootPane().getParent();  //获取主窗体对象
    //获取对家开始游戏的时间
    long towardsTime = frame.getTowardsUser().getTime().getTime();
    //获取自己开始游戏的时间
    long meTime = frame.getUser().getTime().getTime();
    //根据两个玩家开始游戏时间的先后，分配棋子的颜色
    if (meTime >= towardsTime) {
        frame.getChessPanel1().setChessColor(ChessPanel.WHITE_CHESS_ICON);
        frame.getChessPanel1().getGobangPanel1().setTurn(true);
    } else {
        frame.getChessPanel1().setChessColor(ChessPanel.BLACK_CHESS_ICON);
        frame.getChessPanel1().getGobangPanel1().setTurn(false);
    }
}
```

（8）编写 fillChessBoard()方法，该方法用于实现开始游戏时的刷屏动画，根据方法的参数决定使用哪个颜色的棋子填充棋盘，并最终调用该方法的代码，再次调用该方法，使用参数 0，清除棋盘上的所有棋子。关键代码如下：

例程 24　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * 清屏、填充棋盘的方法。可以使用 1 或-1 指定填充棋盘的棋子，使用 0 清屏
 *
```

```
     * @param chessman
     *                - 填充棋盘的棋子的颜色代码
     */
    private void fillChessBoard(final byte chessman) {
        try {
            Runnable runnable = new Runnable() {                    //创建清屏的动画线程
                /**
                 * 线程的主体方法
                 *
                 * @see java.lang.Runnable#run()
                 */
                public void run() {
                    byte[][] chessmanArray = GobangModel.getInstance()
                            .getChessmanArray();                    //获取棋盘数组
                    for (int i = 0; i < chessmanArray.length; i += 2) {
                        try {
                            Thread.sleep(10);                       //动画间隔时间
                        } catch (InterruptedException ex) {
                            Logger.getLogger(ChessPanel.class.getName()).log(
                                    Level.SEVERE, null, ex);
                        }
                        //使用指定颜色的棋子填充数组的一列
                        Arrays.fill(chessmanArray[i], chessman);
                        Arrays.fill(chessmanArray[(i + 1) % 15], chessman);
                        GobangModel.getInstance().updateChessmanArray(
                                chessmanArray);                     //更新棋盘上的棋子
                        gobangPanel1.paintImmediately(0, 0, getWidth(),
                                getHeight());                       //立即重绘指定区域的棋盘
                    }
                }
            };
            //在事件队列中执行清屏
            if (SwingUtilities.isEventDispatchThread()) {
                runnable.run();
            } else {
                SwingUtilities.invokeAndWait(runnable);
            }
        } catch (Exception ex) {
            Logger.getLogger(ChessPanel.class.getName()).log(Level.SEVERE,
                    null, ex);
        }
    }
```

（9）编写“开始”按钮的事件处理方法，该方法将初始化游戏状态并发送开始命令到对方，然后分配对方棋子的颜色，清除棋盘的棋子。关键代码如下：

例程 25 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * “开始”按钮的事件处理方法
```

```
     *
     * @param evt
     *                - 事件对象
     */
    private void startButtonActionPerformed(java.awt.event.ActionEvent evt) {
        //获取主窗体对象
        MainFrame mainFrame = (MainFrame) getRootPane().getParent();
        if (mainFrame.serverSocket == null) {
            JOptionPane.showMessageDialog(this, "请等待对方连接。");
            return;
        }
        if (gobangPanel1.isStart()) {
            return;
        }
        //设置各个按钮的可用状态
        startButton.setEnabled(false);
        giveupButton.setEnabled(true);
        heqiButton.setEnabled(true);
        backButton.setEnabled(true);
        gobangPanel1.setStart(true);                          //设置游戏的开始状态
        gobangPanel1.setTowardsWin(false);                    //设置对方胜利状态
        gobangPanel1.setWin(false);                           //设置自己胜利状态
        gobangPanel1.setDraw(false);                          //设置和棋状态
        send(OPRATION_START);                                 //发送开始指令
        fenqi();                                              //分配双方棋子
        fillChessBoard(gobangPanel1.getMyColor());            //使用自己的棋子颜色清屏
        fillChessBoard((byte) 0);                             //使用空棋子清屏
        byte[][] data = new byte[15][15];                     //创建一个空的棋盘布局
        GobangModel.getInstance().setChessmanArray(data);     //设置棋盘使用空布局
    }
```

（10）编写“认输”按钮的事件处理方法，该方法向对方发送一个认输的指令编码，并启动一个线程使“认输”按钮在 5 秒钟内处于禁用状态。关键代码如下：

例程 26　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
    /**
     * “认输”按钮的事件处理方法
     *
     * @param evt
     *                - 按钮的事件对象
     */
    private void giveupButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if (!gobangPanel1.isTurn()) {                 //如果没到自己走棋，提示用户等待
            JOptionPane.showMessageDialog(this, "没到你走棋呢。", "请等待...",
                    JOptionPane.WARNING_MESSAGE);
            return;
        }
        send(OPRATION_GIVEUP);                        //发送认输指令
        new Thread() {                                //启动一个新的线程，使“认输”按钮 5 秒不可用
```

```
            @Override
            public void run() {
                try {
                    giveupButton.setEnabled(false);
                    sleep(5000);
                    giveupButton.setEnabled(true);
                } catch (InterruptedException ex) {
                    Logger.getLogger(ChessPanel.class.getName()).log(
                            Level.SEVERE, null, ex);
                }
            }
        }.start();
    }
```

（11）编写“悔棋”按钮的事件处理方法，该方法将向对方发送一个悔棋命令的编码，并且该按钮 5 秒钟内处于禁用的状态。关键代码如下：

例程 27 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * “悔棋”按钮的事件处理方法
 *
 * @param evt
 */
private void backButtonActionPerformed(java.awt.event.ActionEvent evt) {
    if (!gobangPanel1.isTurn()) {                    //如果没到自己走棋，提示用户
        JOptionPane.showMessageDialog(this, "没到你走棋呢。", "请等待...",
                JOptionPane.WARNING_MESSAGE);
        return;
    }
    send(OPRATION_REPENT);                           //发送悔棋命令
    new Thread() {                                   //开启新的线程，使“悔棋”按钮禁用 5 秒
        @Override
        public void run() {
            try {
                backButton.setEnabled(false);
                sleep(5000);
                backButton.setEnabled(true);
            } catch (InterruptedException ex) {
                Logger.getLogger(ChessPanel.class.getName()).log(
                        Level.SEVERE, null, ex);
            }
        }
    }.start();
}
```

（12）“和棋”按钮的事件处理方法将向对家发送和棋指令的编码，并且“和棋”按钮在 5 秒钟内不得使用。关键代码如下：

例程 28　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * “和棋”按钮的事件处理方法
 *
 * @param evt
 *              - 按钮的 action 事件对象
 */
private void heqiButtonActionPerformed(java.awt.event.ActionEvent evt) {
    send(OPRATION_DRAW);                        //发送和棋指令
    new Thread() {                              //开启新的线程，使“和棋”按钮 5 秒不可用
        @Override
        public void run() {
            try {
                heqiButton.setEnabled(false);
                sleep(5000);
                heqiButton.setEnabled(true);
            } catch (InterruptedException ex) {
                Logger.getLogger(ChessPanel.class.getName()).log(
                        Level.SEVERE, null, ex);
            }
        }
    }.start();
}
```

（13）游戏面板包含一个广告标题栏，它位于棋盘的上方，如图 10.16 所示。当单击该广告栏时，相应的事件监听器会调用 bannerLabelMouseClicked()方法，该方法会调用 Desktop 类的 browse()方法使用本地的浏览器打开编程词典网站。关键代码如下：

图 10.16　游戏界面的广告标题栏

例程 29　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * 广告图片的鼠标单击事件处理方法
 *
 * @param evt
 *              - 鼠标事件对象
 */
private void bannerLabelMouseClicked(java.awt.event.MouseEvent evt) {
    try {
        //调用 Desktop 类的 browse 方法浏览编程词典首页
        if (Desktop.isDesktopSupported()) {
```

```
                Desktop.getDesktop().browse(
                        new URL("http://www.mrbccd.com").toURI());
            } else {
                JOptionPane.showMessageDialog(this, "当前系统不支持该操作");
            }
        } catch (Exception ex) {
            Logger.getLogger(ChessPanel.class.getName()).log(Level.SEVERE,
                    null, ex);
        }
}
```

（14）游戏回放功能可以观看游戏的比赛记录，该功能将每间隔 1 秒钟根据棋谱的记录，演示双方的游戏过程。“游戏回放”按钮的事件处理方法由 backplayToggleButtonActionPerformed()方法实现。关键代码如下：

例程 30 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * “游戏回放”按钮的事件处理方法
 *
 * @param evt
 */
private void backplayToggleButtonActionPerformed(
        java.awt.event.ActionEvent evt) {
    //如果游戏进行中，提示用户游戏结束后再观看游戏回放
    if (gobangPanel1.isStart()) {
        JOptionPane.showMessageDialog(this, "请在游戏结束后，观看游戏回放。");
        backplayToggleButton.setSelected(false);
        return;
    }
    if (!backplayToggleButton.isSelected()) {
        backplayToggleButton.setText("游戏回放");
    } else {
        backplayToggleButton.setText("终止回放");
        new Thread() {                                    //开启新的线程播放游戏记录
            public void run() {
                Object[] toArray = gobangPanel1.getOldRec();
                if (toArray == null) {
                    JOptionPane.showMessageDialog(ChessPanel.this,
                            "没有游戏记录", "游戏回放", JOptionPane.WARNING_MESSAGE);
                    backplayToggleButton.setText("游戏回放");
                    backplayToggleButton.setSelected(false);
                    return;
                }
                //清除界面的结局文字，包括对方胜利、你胜利了、此战平局
                gobangPanel1.setTowardsWin(false);
                gobangPanel1.setWin(false);
```

```
                    gobangPanel1.setDraw(false);
                    for (int i = toArray.length - 1; !gobangPanel1.isStart()
                            && i >= 0; i--) {
                        try {
                            Thread.sleep(1000);                    //线程休眠 1 秒
                        } catch (InterruptedException ex) {
                            Logger.getLogger(ChessPanel.class.getName()).log(
                                    Level.SEVERE, null, ex);
                        }
                        GobangModel.getInstance().updateChessmanArray(
                                (byte[][]) toArray[i]);             //根据游戏记录更换每一步游戏的棋谱
                        gobangPanel1.repaint();                    //重绘棋盘
                    }
                    backplayToggleButton.setSelected(false);
                    backplayToggleButton.setText("游戏回放");
                }
            }.start();
        }
    }
```

（15）“更换背景”按钮的事件监听器由 ButtonActionListener 类定义，它实现了 ActionListener 接口和接口的 actionPerformed()方法，并在该方法中重新定义背景图片对象 backImg，然后调用 repaint()方法使用新的背景绘制界面。关键代码如下：

例程 31　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ChessPanel.java

```
/**
 * “更换背景”按钮的事件监听器
 *
 * @author Li Zhong Wei
 */
private class ButtonActionListener implements ActionListener {
    public void actionPerformed(final ActionEvent e) {
        backIndex = backIndex % 9 + 1;                             //获取 9 张背景图片的索引递增
        URL url = getClass().getResource("/res/bg/" + backIndex + ".jpg");
        backImg = new ImageIcon(url).getImage();                   //初始化棋盘图片
        repaint();                                                 //重新绘制下棋面板
    }
}
```

10.8　编写棋盘面板

棋盘面板和下棋面板是密不可分的，实际上是由这两个面板一同组成的下棋功能。棋盘面板负责五子棋游戏的落子和游戏规则，如图 10.17 所示。

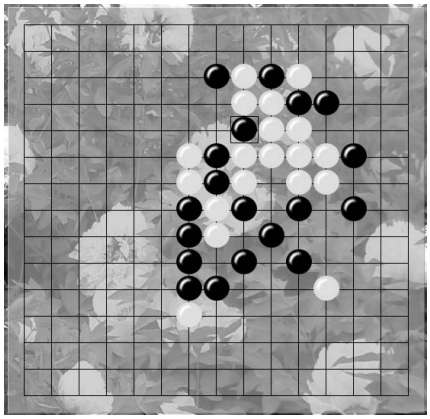


图 10.17　棋盘面板的界面效果

实现棋盘面板的关键步骤如下：

（1）继承 JPanel 类编写 GobangPanel 棋盘面板组件。在 GobangPanel 类中定义开始、胜利、和棋等游戏状态的控制变量、记录游戏过程的队列等变量，并在构造方法中初始化棋盘面板需要的图片对象。关键代码如下：

例程 32　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\GobangPanel.java

```
/**
 * 棋盘面板
 * @author Li Zhong Wei
 */
public class GobangPanel extends javax.swing.JPanel {
    //黑白棋盒的图标对象和星号的图像对象以及背景图片对象
    private Image backImg, white_chessman_img, black_chessman_img,
            rightTop_img;
    int chessWidth, chessHeight;                                    //棋子宽度与高度
    public final static byte WHITE_CHESSMAN = 1, BLACK_CHESSMAN = -1;
    Dimension size;                                                 //棋盘面板的大小
    private boolean start = false;                                  //开始
    private Object[] oldRec;
    Deque<byte[][]> chessQueue = new LinkedList<byte[][]>();        //游戏的队列记录
    private boolean turn = false;                                   //是否到自己走棋
    private boolean towardsWin;                                     //对方胜利
    private boolean win;                                            //胜利
    private boolean draw;                                           //和棋
    private ChessPanel chessPanel;
    /**
     * 棋盘面板的构造方法
     */
    public GobangPanel() {
        URL white_url = getClass().getResource("/res/whiteChessman.png");
        URL black_url = getClass().getResource("/res/blackChessman.png");
        URL rightTop_url = getClass().getResource("/res/rightTop.gif");
        white_chessman_img = new ImageIcon(white_url).getImage(); //初始化白棋图片
```

```
            black_chessman_img = new ImageIcon(black_url).getImage(); //初始化黑棋图片
            rightTop_img = new ImageIcon(rightTop_url).getImage();        //初始化连成线的棋子上的星图
            size = new Dimension(getWidth(), getHeight());
            setPreferredSize(size);
            initComponents();
        }
…//省略部分代码
}
```

（2）重写父类的 paint()方法，该方法负责绘制组件的界面，本模块在棋盘面板的 paint()方法中绘制棋盘、棋子和游戏的提示信息。绘制棋子时又包含多种落子状态，例如分别绘制白棋、黑棋、绘制最近白棋或黑棋的落子（带边框的）、胜利后为连成线的棋子绘制五星等。关键代码如下：

例程 33　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\GobangPanel.java

```
/**
 * 重写父类的 paint()方法，绘制自己的组件界面
 * @see javax.swing.JComponent#paint(java.awt.Graphics)
 */
@Override
public void paint(Graphics g1) {
    Graphics2D g = (Graphics2D) g1;
    super.paint(g);                                          //调用父类的绘图方法
    if (chessPanel != null) {
        chessPanel.setTurn(turn);
    }
    Composite composite = g.getComposite();                  //备份合成模式
    drawPanel(g);                                            //调用绘制棋盘的方法
    g.translate(4, 4);
    size = new Dimension(getWidth(), getHeight());           //获取棋盘面板的大小
    chessWidth = size.width / 15;                            //初始化棋子宽
    chessHeight = size.height / 15;                          //初始化棋子高
    byte[][] chessmanArray = gobangModel1.getChessmanArrayCopy();
    for (int i = 0; i < chessmanArray.length; i++) {         //遍历棋盘数据模型绘制棋子
        for (int j = 0; j < chessmanArray[i].length; j++) {
            byte chessman = chessmanArray[i][j];
            int x = i * chessWidth;
            int y = j * chessHeight;
            if (chessman != 0)
                System.out.println("chess is:" + chessman);
            if (chessman == WHITE_CHESSMAN) {                //绘制白棋
                g.drawImage(white_chessman_img, x, y, chessWidth,
                        chessHeight, this);
            } else if (chessman == BLACK_CHESSMAN) {         //绘制黑棋
                g.drawImage(black_chessman_img, x, y, chessWidth,
                        chessHeight, this);
            } else if (chessman == (WHITE_CHESSMAN ^ 3)) {   //绘制最近的白棋落子
                g.drawImage(white_chessman_img, x, y, chessWidth,
                        chessHeight, this);
```

```
                g.drawRect(x, y, chessWidth, chessHeight);
            } else if (chessman == (BLACK_CHESSMAN ^ 3)) {          //绘制最近的黑棋落子
                g.drawImage(black_chessman_img, x, y, chessWidth,
                        chessHeight, this);
                g.drawRect(x, y, chessWidth, chessHeight);
            } else if (chessman == ((byte) (WHITE_CHESSMAN ^ 8))) {  //绘制导致胜利的连线白棋
                g.drawImage(white_chessman_img, x, y, chessWidth,
                        chessHeight, this);
                g.drawImage(rightTop_img, x, y, chessWidth, chessHeight,
                        this);
            } else if (chessman == (BLACK_CHESSMAN ^ 8)) {          //绘制导致胜利的连线黑棋
                g.drawImage(black_chessman_img, x, y, chessWidth,
                        chessHeight, this);
                g.drawImage(rightTop_img, x, y, chessWidth, chessHeight,
                        this);
            }
        }
    }
    if (!isStart()) {                                               //如果游戏不处于开始状态
        if (towardsWin || win || draw) {   //如果游戏处于胜利或和棋状态，绘制棋盘提示信息
            g.setComposite(AlphaComposite.SrcOver.derive(0.7f));    //设置 70%透明的合成规则
            String mess = "对方胜利";                                //定义提示信息
            g.setColor(Color.RED);                                  //设置前景色为红色
            if (win) {                                              //如果是自己胜利
                mess = "你胜利了";                                   //设置胜利提示信息
                g.setColor(new Color(0x007700));                    //设置绿色前景色
            } else if (draw) {                                      //如果是和棋状态
                mess = "此战平局";                                   //定义和棋提示信息
                g.setColor(Color.YELLOW);                           //设置和棋信息，使用黄色提示
            }
            //设置提示文本的字体为隶书、粗斜体、大小 72
            Font font = new Font("隶书", Font.ITALIC | Font.BOLD, 72);
            g.setFont(font);
            //获取字体渲染上下文对象
            FontRenderContext context = g.getFontRenderContext();
            //计算提示信息的文本所占用的像素空间
            Rectangle2D stringBounds = font.getStringBounds(mess, context);
            double fontWidth = stringBounds.getWidth();             //获取提示文本的宽度
            g.drawString(mess, (int) ((getWidth() - fontWidth) / 2),
                    getHeight() / 2);                               //居中绘制提示信息
            g.setComposite(composite);                              //恢复原有合成规则
        } else {                                                    //如果当前处于其他未开始游戏的状态
            String mess = "等待开始…";                               //定义等待提示信息
            Font font = new Font("隶书", Font.ITALIC | Font.BOLD, 48);
            g.setFont(font);                                        //设置 48 号隶书字体
            FontRenderContext context = g.getFontRenderContext();
            Rectangle2D stringBounds = font.getStringBounds(mess, context);
            double fontWidth = stringBounds.getWidth();             //获取提示文本的宽度
            g.drawString(mess, (int) ((getWidth() - fontWidth) / 2),
```

```
                getHeight() / 2);                    //居中绘制提示文本
        }
    }
}
```

（3）编写绘制棋盘的 drawPanel()方法，该方法将利用半透明合成规则绘制透明背景的棋盘，另外，为了更好地支持棋盘缩放，使用户能自由调整棋盘大小，这里使用 drawLine()方法绘制了棋盘的网格。程序的界面如图 10.18 所示。

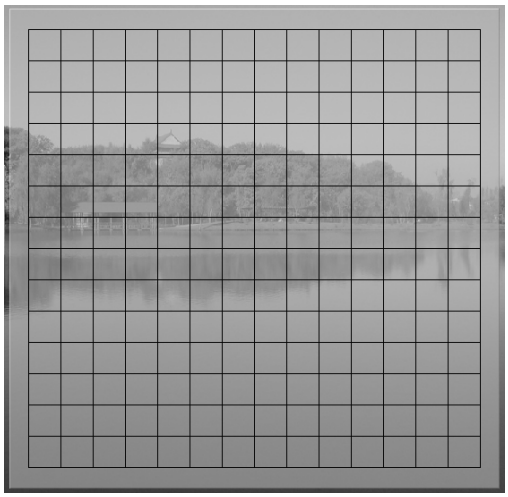


图 10.18　绘制半透明的棋盘效果

绘制棋盘方法的关键代码如下：

例程 34　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\GobangPanel.java

```
/**
 * 绘制棋盘的方法
 *
 * @param g
 *                - 绘图对象
 */
private void drawPanel(Graphics2D g) {
    Composite composite = g.getComposite();                    //备份合成规则
    Color color = g.getColor();                                //备份前景颜色
    g.setComposite(AlphaComposite.SrcOver.derive(0.6f));       //设置透明合成
    g.setColor(new Color(0xAABBAA));                           //设置前景白色
    g.fill3DRect(0, 0, getWidth(), getHeight(), true);         //绘制半透明的矩形
    g.setComposite(composite);                                 //恢复合成规则
    g.setColor(color);                                         //恢复原来前景色
    int w = getWidth();                                        //棋盘宽度
    int h = getHeight();                                       //棋盘高度
    int chessW = w / 15, chessH = h / 15;                      //棋子宽度和高度
    int left = chessW / 2 + (w % 15) / 2;                      //棋盘左边界
    int right = left + chessW * 14;                            //棋盘右边界
```

```
        int top = chessH / 2 + (h % 15) / 2;                    //棋盘上边界
        int bottom = top + chessH * 14;                         //棋盘下边界
        for (int i = 0; i < 15; i++) {
            //画每条横线
            g.drawLine(left, top + (i * chessH), right, top + (i * chessH));
        }
        for (int i = 0; i < 15; i++) {
            //画每条竖线
            g.drawLine(left + (i * chessW), top, left + (i * chessW), bottom);
        }
}
```

10.9　实现游戏规则算法

arithmetic()方法是根据五子棋的游戏规则编写的计算方法，它根据当前棋子的类型和位置，计算是黑棋赢还是白棋赢，计算方法是以落子点为中心，向左右两边、上下两边、正斜两边、反斜两边查找同一类型的棋子，如果棋子数大于等于 5，则表示当前下棋者为赢家。游戏胜利和失败的界面如图 10.19 和图 10.20 所示。

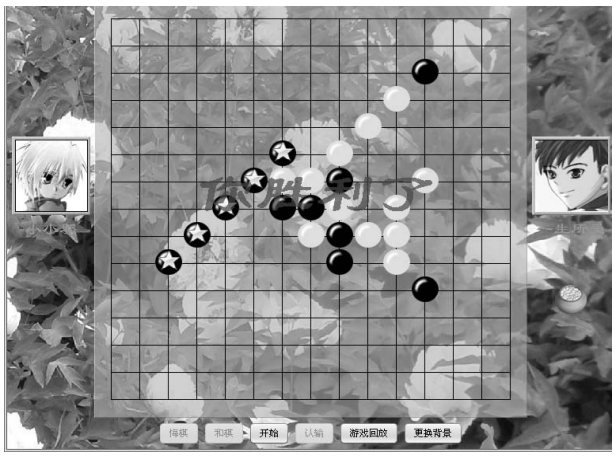


图 10.19　游戏胜利界面

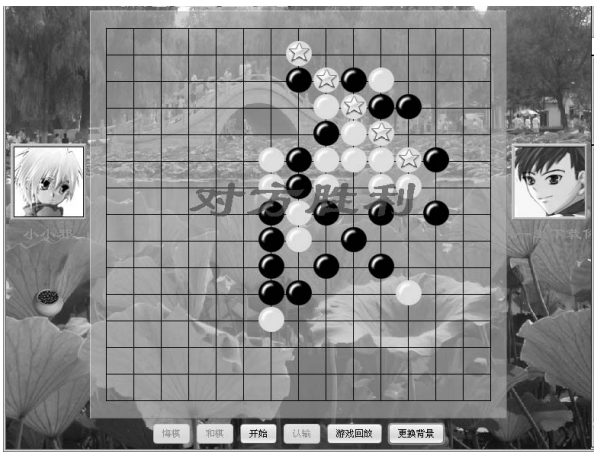


图 10.20　游戏失败界面

实现五子棋算法的关键代码如下：

例程 35　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\GobangPanel.java

```
/**
 * 五子棋算法
 * @param n - 代表棋子颜色的整数
 * @param Arow     - 行编号
 * @param Acolumn - 列编号
 * @return       胜利一方的棋子颜色的整数
 */
public int arithmetic(int n, int Arow, int Acolumn) {
```

```
int n3 = n ^ 3;
byte n8 = (byte) (n ^ 8);
byte[][] note = gobangModel1.getChessmanArrayCopy();
int BCount = 1;
//纵向查找
boolean Lbol = true;
boolean Rbol = true;
BCount = 1;
for (int i = 1; i <= 5; i++) {
    if ((Acolumn + i) > 14) {                          //如果棋子超出最大列数
        Rbol = false;
    }
    if ((Acolumn - i) < 0) {                           //如果棋子超出最小列数
        Lbol = false;
    }
    if (Rbol == true) {
        if (note[Arow][Acolumn + i] == n
                || note[Arow][Acolumn + i] == n3) {    //如果横向向右有相同的棋子
            ++BCount;
            note[Arow][Acolumn + i] = n8;
        } else {
            Rbol = false;
        }
    }
    if (Lbol == true) {
        if (note[Arow][Acolumn - i] == n
                || note[Arow][Acolumn - i] == n3) {    //如果横向向左有相同的棋子
            ++BCount;
            note[Arow][Acolumn - i] = n8;
        } else {
            Lbol = false;
        }
    }
    if (BCount >= 5) {                                 //如果同类型的棋子数大于等于 5 个
        note[Arow][Acolumn] = n8;
        gobangModel1.updateChessmanArray(note);
        repaint();
        return n;                                      //返回胜利一方的棋子
    }
}
//横向查找
note = gobangModel1.getChessmanArrayCopy();
boolean Ubol = true;
boolean Dbol = true;
BCount = 1;
for (int i = 1; i <= 5; i++) {
    if ((Arow + i) > 14) {                             //如果超出棋盘的最大行数
        Dbol = false;
    }
```

```
        if ((Arow - i) < 0) {                                    //如果超出棋盘的最小行数
            Ubol = false;
        }
        if (Dbol == true) {
            if (note[Arow + i][Acolumn] == n
                    || note[Arow + i][Acolumn] == n3) {          //如果向上有同类型的棋子
                ++BCount;
                note[Arow + i][Acolumn] = n8;
            } else {
                Dbol = false;
            }
        }
        if (Ubol == true) {
            if (note[Arow - i][Acolumn] == n
                    || note[Arow - i][Acolumn] == n3) {          //如果向下有同类型的棋子
                ++BCount;
                note[Arow - i][Acolumn] = n8;
            } else {
                Ubol = false;
            }
        }
        if (BCount >= 5) {                                       //如果同类型的棋子大于等于 5 个
            note[Arow][Acolumn] = n8;
            gobangModel1.updateChessmanArray(note);
            repaint();
            return n;                                            //返回胜利一方的棋子
        }
    }
    //正斜查找
    note = gobangModel1.getChessmanArrayCopy();
    boolean LUbol = true;
    boolean RDbol = true;
    BCount = 1;
    for (int i = 1; i <= 5; i++) {
        if ((Arow - i) < 0 || (Acolumn - i < 0)) {                //如果超出左面的斜线
            LUbol = false;
        }
        if ((Arow + i) > 14 || (Acolumn + i > 14)) {              //如果超出右面的斜线
            RDbol = false;
        }
        if (LUbol == true) {
            if (note[Arow - i][Acolumn - i] == n
                    || note[Arow - i][Acolumn - i] == n3) {       //如果左上斜线上有相同类型的棋子
                ++BCount;
                note[Arow - i][Acolumn - i] = n8;
            } else {
                LUbol = false;
            }
        }
```

```
        if (RDbol == true) {
            if (note[Arow + i][Acolumn + i] == n
                    || note[Arow + i][Acolumn + i] == n3) {      //如果右下斜线上有相同类型的棋子
                ++BCount;
                note[Arow + i][Acolumn + i] = n8;
            } else {
                RDbol = false;
            }
        }
        if (BCount >= 5) {                                       //如果同类型的棋子大于等于 5 个
            note[Arow][Acolumn] = n8;
            gobangModel1.updateChessmanArray(note);
            repaint();
            return n;                                            //返回胜利一方的棋子
        }
    }
    //反斜查找
    note = gobangModel1.getChessmanArrayCopy();
    boolean RUbol = true;
    boolean LDbol = true;
    BCount = 1;
    for (int i = 1; i <= 5; i++) {
        if ((Arow - i) < 0 || (Acolumn + i > 14)) {
            RUbol = false;
        }
        if ((Arow + i) > 14 || (Acolumn - i < 0)) {
            LDbol = false;
        }
        if (RUbol == true) {
            if (note[Arow - i][Acolumn + i] == n
                    || note[Arow - i][Acolumn + i] == n3) {      //如果左下斜线上有相同类型的棋子
                ++BCount;
                note[Arow - i][Acolumn + i] = n8;
            } else {
                RUbol = false;
            }
        }
        if (LDbol == true) {
            if (note[Arow + i][Acolumn - i] == n
                    || note[Arow + i][Acolumn - i] == n3) {      //如果右上斜线上有相同类型的棋子
                ++BCount;
                note[Arow + i][Acolumn - i] = n8;
            } else {
                LDbol = false;
            }
        }
        if (BCount >= 5) {                                       //如果同类型的棋子大于等于 5 个
            note[Arow][Acolumn] = n8;
            gobangModel1.updateChessmanArray(note);
```

```
                repaint();
                return n;                                    //返回胜利一方的棋子
            }
        }
        return 0;
    }
```

10.10 编写棋盘模型

棋盘模型是一个 JavaBean，它主要用于记录棋盘上的棋子。如果该模型的数据被改变，那么它将通知所有监听该模型的事件监听器，棋盘面板就定义了一个这样的事件监听器。关键代码如下：

例程 36 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\GobangPanel.java

```
gobangModel1 = new GobangModel();                                    //创建棋盘模型的实例对象
gobangModel1.addPropertyChangeListener(new PropertyChangeListener() {  //为棋盘添加事件监听器
    public void propertyChange(java.beans.PropertyChangeEvent evt) {
        gobangModel1PropertyChange(evt);                             //调用事件处理方法
    }
});
```

在这个事件监听器中将调用 gobangModel1PropertyChange()方法执行相应的业务逻辑，该方法将最新的棋盘数据压入队列中，然后根据新的棋盘数据重新绘制棋盘界面上的棋子。关键代码如下：

例程 37 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\GobangPanel.java

```
/**
 * 棋盘数据模型的事件处理方法
 * @param evt
 */
private void gobangModel1PropertyChange(java.beans.PropertyChangeEvent evt) {
    chessQueue.push(gobangModel1.getChessmanArrayCopy());            //将新的棋盘布局压入队列
    repaint();                                                       //重回棋盘界面
}
```

在定义棋盘模型的 JavaBean 时，必须定义一个 PropertyChangeSupport 类的实例对象作为类的字段（全局变量），并将各种事件监听器的工作委托给它。关键代码如下：

例程 38 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\GobangModel.java

```
/**
 * 承载棋盘棋子的数据模型 JavaBean
 * @author Li Zhong Wei
 */
public class GobangModel extends Object implements Serializable {
    private PropertyChangeSupport propertySupport;                   //定义属性工具类
    private static GobangModel model;                                //定义自身的变量
    private byte[][] chessmanArray = new byte[15][15];               //定义棋子数组
```

```
public static final String PROP_CHESSMANARRAY = "chessmanArray";  //定义属性名称
/**
 * 获取本类实例的方法
 * @return
 */
public static GobangModel getInstance() {
        if (model == null) {
                model = new GobangModel();
        }
        return model;
}
/**
 * 棋盘模型的构造方法
 */
public GobangModel() {
        propertySupport = new PropertyChangeSupport(this);                    //初始化属性工具栏
        model = this;
}
/**
 * 获取棋盘的棋子数组的方法
 * @return - 代表棋子的数组
 */
public byte[][] getChessmanArray() {
        return chessmanArray;                                                 //返回棋子数组
}
/**
 * 设置棋子数组的方法
 * @param chessmanArray - 一个代表棋盘棋子的二维数组
 */
public void setChessmanArray(byte[][] chessmanArray) {
        this.chessmanArray = chessmanArray;
        propertySupport.firePropertyChange(PROP_CHESSMANARRAY, null,
                        chessmanArray);                                       //报告所有已注册监听器的绑定属性更新
}
/**
 * 更新棋子数组的方法，不会产生更新事件
 * @param chessmanArray
 */
public synchronized void updateChessmanArray(byte[][] chessmanArray) {
        this.chessmanArray = chessmanArray;
}
/**
 * 添加事件监听器的方法
 * @param listener - 事件监听器
 */
public void addPropertyChangeListener(PropertyChangeListener listener) {
        propertySupport.addPropertyChangeListener(listener);                  //添加事件监听器
}
```

```
    /**
     * 移除事件监听器的方法
     * @param listener - 事件监听器
     */
    public void removePropertyChangeListener(PropertyChangeListener listener) {
        propertySupport.removePropertyChangeListener(listener);          //移除事件监听器
    }
    /**
     * 获取棋盘上棋子数组的复制
     * @return - 棋子数组
     */
    byte[][] getChessmanArrayCopy() {
        byte[][] newArray = new byte[15][15];                            //创建一个二维数组
        for (int i = 0; i < newArray.length; i++) {
            //复制数组
            newArray[i] = Arrays.copyOf(chessmanArray[i], newArray[i].length);
        }
        return newArray;
    }
}
```

10.11 编写联机通讯类

本模块使用一个联机通讯类来接收对方发送的所有信息，包括游戏命令代码的接收与处理。这个联机通讯类 ReceiveThread 是一个线程类，它继承 Thread 类并重写类的 run()方法来处理联机业务。run()方法是线程的核心方法，本类在该方法中接收远程计算机的联机请求，根据对方的联机信息填充登录面板的 IP 地址文本框，从网络中读取 Java 对象，并根据对象的类型判断信息的种类是聊天信息、登录信息还是命令代码等，并做相应的业务处理。关键代码如下：

例程 39 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ReceiveThread.java

```
/**
 * 线程的主体方法
 * @see java.lang.Thread#run()
 */
@Override
public void run() {
    while (true) {
        try {
            frame.serverSocket = chatSocketServer.accept();              //接收 Socket 连接
            Socket serverSocket = frame.serverSocket;
            host = serverSocket.getInetAddress().getHostName();          //获取对方主机信息
            String ip = serverSocket.getInetAddress().getHostAddress();  //获取对方 IP 地址
            int link = JOptionPane.showConfirmDialog(frame, "收到" + host
                    + "的联机请求，是否接受？");                           //询问是否接受联机
            if (link == JOptionPane.YES_OPTION) {                        //如果接受联机
```

```
                LoginPanel loginPanel = (LoginPanel) frame.getRootPane()
                        .getGlassPane();                          //获取登录面板的实例
                loginPanel.setLinkIp(ip);                         //设置登录面板的对方 IP
            }
            serverSocket.setOOBInline(true);                      //启用紧急数据的接收
            InputStream is = serverSocket.getInputStream();       //获取网络输入流
            ObjectInputStream objis = new ObjectInputStream(is);  //创建对象输入流
            while (frame.isVisible()) {
                serverSocket.sendUrgentData(255);                 //发送紧急数据
                Object messageObj = objis.readObject();           //从对象输入流读取 Java 对象
                if (messageObj instanceof String) {               //如果读取的对象是 String 类型
                    String name = frame.getTowardsUser().getName(); //获取对方昵称
                    //将字符串信息添加到通讯面板
                    rame.appendMessage(name + ":" + messageObj);
                } else if (messageObj instanceof byte[][]) {      //如果读取的是字节数组对象
                    //将数组对象设置为棋盘模型数据
                    obangModel.getInstance().setChessmanArray(
                            (byte[][]) messageObj);
                    frame.getChessPanel1().getGobangPanel1().setTurn(true);     //获得走棋权限
                    byte myColor = frame.getChessPanel1().getGobangPanel1()
                            .getMyColor();                        //获取自己的棋子颜色
                    frame.getChessPanel1().getGobangPanel1().zhengliBoard(
                            myColor);                             //整理棋盘
                    frame.getChessPanel1().backButton.setEnabled(true);          //悔棋按钮可用
                } else if (messageObj instanceof Integer) {       //如果是整型对象
                    oprationHandler(messageObj);                  //命令代码的接收和处理方法
                } else if (messageObj instanceof UserBean) {      //如果是用户实体对象
                    UserBean user = (UserBean) messageObj;
                    frame.setTowardsUser(user);                   //设置对方信息
                }
            }
        } catch (SocketException ex) {
            Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE,
                    null, ex);
            JOptionPane.showMessageDialog(frame, "连接中断");
        } catch (IOException ex) {
            Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE,
                    null, ex);
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE,
                    null, ex);
        }
    }
}
```

在线程的 run()方法中接收到命令请求时，将调用 operationHandler()方法处理相应的业务逻辑，该方法将判断命令的类型，根据不同的命令执行不同的操作。关键代码如下：

例程 40 代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ReceiveThread.java

```
/**
 * 处理远程命令的方法
 * @param messageObj -  命令代码
 */
private void oprationHandler(Object messageObj) {
    int code = (Integer) messageObj;                                    //获取命令代码
    String towards = frame.getTowardsUser().getName();                  //获取对方昵称
    int option;
    switch (code) {
    case ChessPanel.OPRATION_REPENT:                                    //如果是悔棋请求
        System.out.println("请求悔棋");
        //询问玩家是否同意对方悔棋
        option = JOptionPane.showConfirmDialog(frame,
                towards + "要悔棋，是否同意？", "求你了，我走错了，让我悔棋！！！",
                JOptionPane.YES_NO_OPTION, JOptionPane.INFORMATION_MESSAGE);
        //在聊天面板添加悔棋信息
        frame.appendMessage("对方请求悔棋.....");
        if (option == JOptionPane.YES_OPTION) {                         //如果同意悔棋
            frame.send(ChessPanel.OPRATION_NODE_REPENT);                //发送同意悔棋的消息
            frame.getChessPanel1().repentOperation();                   //执行本地的悔棋操作
            frame.appendMessage("接受对方的悔棋请求。");                  //添加悔棋信息到聊天面板
            frame.send(frame.getUser().getName() + "接受悔棋请求");
        } else {                                                        //如果不同意悔棋
            //添加不同意悔棋的信息到聊天面板
            frame.send(frame.getUser().getName() + "拒绝悔棋请求");
            frame.appendMessage("拒绝了对方的悔棋请求。");
        }
        break;
    case ChessPanel.OPRATION_NODE_REPENT:                               //如果是同意悔棋命令
        System.out.println("同意悔棋命令");
        frame.getChessPanel1().repentOperation();                       //执行本地的悔棋操作
        frame.appendMessage("悔棋成功");                                //把悔棋成功信息添加到聊天面板
        break;
    case ChessPanel.OPRATION_NODE_DRAW:                                 //如果是同意和棋命令
        System.out.println("同意和棋命令");
        frame.getChessPanel1().getGobangPanel1().setDraw(true);         //设置和棋状态为 true
        frame.getChessPanel1().reInit();                                //初始化游戏状态变量
        frame.appendMessage("此战平局。");                               //将和棋信息添加到聊天面板
        break;
    case ChessPanel.OPRATION_DRAW:                                      //如果是和棋请求
        System.out.println("请求和棋");
        //询问玩家是否同意和棋
        option = JOptionPane.showConfirmDialog(frame, towards
                + "请求和棋，是否同意？", "大哥，和棋吧！！！", JOptionPane.YES_NO_OPTION,
                JOptionPane.QUESTION_MESSAGE);
        frame.appendMessage("对方请求和棋.....");                        //添加信息到聊天面板
        if (option == JOptionPane.YES_OPTION) {                         //如果同意和棋
```

```
                frame.send(ChessPanel.OPRATION_NODE_DRAW);                    //发送接受和棋的消息
                frame.getChessPanel1().getGobangPanel1().setDraw(true);       //设置和棋状态为 true
                frame.getChessPanel1().reInit();                              //初始化游戏状态变量
                frame.appendMessage("接受对方的和棋请求。");                    //添加信息到聊天面板
                frame.send(frame.getUser().getName() + "接受和棋请求");
            } else {                                                          //如果不同意和棋
                frame.send(frame.getUser().getName() + "拒绝和棋请求");        //发送拒绝信息
                frame.appendMessage("拒绝了对方的和棋请求。");
            }
            break;
        case ChessPanel.OPRATION_GIVEUP:                                      //如果是对方认输的请求
            System.out.println("对方认输");
            //询问玩家是否同意对方认输
            option = JOptionPane.showConfirmDialog(frame, towards
                    + "请求认输，是否同意？", "对方认输", JOptionPane.YES_NO_OPTION);
            frame.appendMessage("对方请求认输.....");
            if (option == JOptionPane.YES_OPTION) {                           //如果同意对方认输
                frame.send(ChessPanel.WIN);                                   //发送胜利消息
                frame.getChessPanel1().getGobangPanel1().setWin(true);        //设置胜利状态为 true
                frame.getChessPanel1().reInit();                              //初始化游戏的状态变量
                frame.appendMessage("接受对方的认输请求。");
            } else {
                frame.send(frame.getUser().getName() + "拒绝认输请求");
                frame.appendMessage("拒绝了对方的认输请求。");
            }
            break;
        case ChessPanel.OPRATION_START:                                       //如果是开始游戏的请求
            System.out.println("请求开始");
            if (frame.getChessPanel1().getGobangPanel1().isStart()) {         //如果自己已经开始游戏
                frame.send((int) ChessPanel.OPRATION_ALL_START);              //发送全部开始命令
                frame.getChessPanel1().setTowardsStart(true);                 //设置对方游戏开始状态为 true
            }
            break;
        case ChessPanel.OPRATION_ALL_START:                                   //如果是回应开始请求
            System.out.println("回应开始请求");
            frame.getChessPanel1().setTowardsStart(true);                     //设置对方为开始状态
            break;
        case ChessPanel.WIN:                                                  //如果是胜利的命令代码
            System.out.println("对方胜利");
            //设置对方胜利状态为 true
            frame.getChessPanel1().getGobangPanel1().setTowardsWin(true);
            frame.getChessPanel1().reInit();                                  //初始化游戏状态变量
            break;
        default:
            System.out.println("未知操作代码：" + code);
        }
    }
```

10.12 系统打包发布

Java 应用程序可以打包成 JAR 文件，JAR 文件是一个简单的 ZIP 格式的文件，它包含程序中的类文件和执行程序的其他资源文件。在程序发布前，需要将所有编译好的 Java 文件封装到一个程序打包文件中，然后将这个程序的打包文件提交给客户使用。一旦程序打包之后，就可以使用简单的命令来执行它。另外，如果配置好 Java 环境或使用 JDK 的安装程序构建 Java 环境，那么就可以像运行本地可执行文件一样去执行 JAR 文件。在早期版本的 Eclipse 开发工具中，需要自行编写 MANIFEST.MF 描述文件，这个描述文件包含程序的配置信息，例如主类名称、类路径等。

本节将介绍如何使用 Eclipse 3.5 开发工具将程序打包成可执行的 JAR 文件。而且不需要编写任何描述文件，Eclipse 会自动完成。下面介绍关键步骤。

（1）在 Eclipse 的资源包管理器中右击项目的 src 文件夹，在弹出的快捷菜单中选择“导出”命令。

（2）在弹出的“导出”对话框中选择 Java/“可运行的 JAR 文件”子节点，如图 10.21 所示。单击“下一步”按钮。

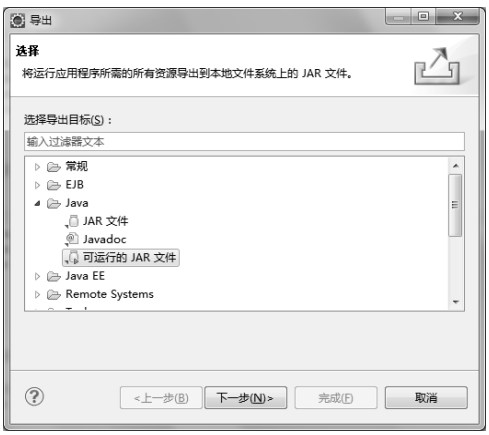


图 10.21 “导出”对话框

（3）在弹出的“可运行的 JAR 文件导出”对话框中选择要导出的“启动配置”，这个启动配置的字符串格式是项目运行的主类名称与项目名称连接。然后在“导出目标”文本框中输入要保存的可执行 JAR 文件的名称与路径，也可以通过单击右侧的“浏览”按钮来选择保存文件，如图 10.22 所示。单击“完成”按钮。

（4）在导出可运行的 JAR 文件后，如果程序的源代码中包含警告信息，那么这时会弹出对话框显示包含警告信息的源代码文件名称，如图 10.23 所示。单击“确定”按钮。

图 10.22 “可运行的 JAR 文件导出”对话框

图 10.23 警告对话框

（5）如果指定的保存 JAR 文件已经存在，向导还会弹出询问对话框，让用户确认是否覆盖目标文件，用户也可以取消，然后重新指定目标 JAR 文件。

10.13　开发技巧与难点分析

在进行游戏的过程中，为了防止由于网络故障或某一方掉线使得游戏无法结束、无法重新开始游戏，在网络五子棋模块添加了网络状态检测功能。实现网络状态检测功能对于网络应用程序非常重要，本实例采用的网络状态检测方式是在建立网络连接后，调用 Socket 的 setOOBInline()方法启用紧急数据接收，然后 Socket 的另一端会调用 sendUrgentData()方法向自己发送紧急数据，同时本地也会执行 Socket 类的 sendUrgentData()方法向对方发送紧急数据，如果对方网络有故障或者掉线了，那么该方法会抛出异常，说明网络连接断开。程序的运行效果如图 10.24 所示。

图 10.24　网络中断的提示界面

实现网络状态检测的关键代码如下：

例程 41　代码位置：光盘\mr\10\Gobang\src\com\lzw\gobang\ReceiveThread.java

```
try {
    serverSocket.setOOBInline(true);                          //启用紧急数据的接收
    InputStream is = serverSocket.getInputStream();           //获取网络输入流
    ObjectInputStream objis = new ObjectInputStream(is);      //创建对象输入流
    while (frame.isVisible()) {
        serverSocket.sendUrgentData(255);                     //发送紧急数据
        …//省略其他关键代码
    }
    …//省略其他关键代码
} catch (SocketException ex) {
    Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE,
            null, ex);
    JOptionPane.showMessageDialog(frame, "连接中断");      //提示用户网络错误
}
```

10.14　本 章 小 结

本章运用软件工程的设计思想，通过棋牌游戏系统中的五子棋程序为读者详细讲解了一个系统的开发流程。通过本章的学习，读者可以了解 Java 应用程序的开发流程及 Java Swing 的控件自定义、绘图特效、网络状态检测、事件监听等技术。

推荐与延伸阅读

《Java 开发实战 1200 例》第 I 卷、第 II 卷各精选了 600 个实例和 500 个经验技巧，涵盖了编程多个方面的应用，是目前市场上最全的编程实例类图书，堪称编程实例的“四库全书”，开发中常用的技术、技巧在书中几乎都可以找到。

书号：978-7-302-24262-8
定价：96.00元

本书适合有一些编程基础的读者使用，如果读者朋友看本书有难度，建议先看看《Java 开发实战》，它含有 41 小时专业视频，313 个实例、模块、项目分析等。

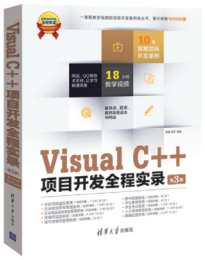
书号：978-7-302-31894-1
定价：89.80元（附DVD视频光盘1张）

软件项目开发全程实录

一套展现项目开发完整过程、配备教学视频的案例类丛书

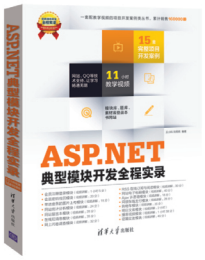
ISBN 978-7-302-33746-1

定价：49.80元



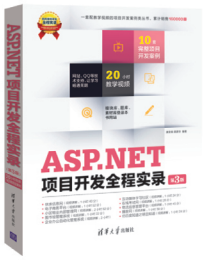
ISBN 978-7-302-33747-8

定价：69.80元



ISBN 978-7-302-33766-9

定价：59.80元



ISBN 978-7-302-33742-3

定价：69.80元

ISBN 978-7-302-33739-3

定价：69.80元

ISBN 978-7-302-33744-7

定价：59.80元

ISBN 978-7-302-33767-6

定价：79.80元

ISBN 978-7-302-33741-6

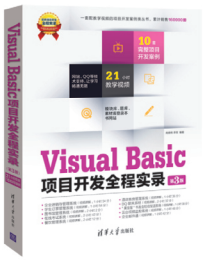
定价：69.80元

ISBN 978-7-302-33740-9

定价：59.80元

ISBN 978-7-302-33743-0

定价：69.80元



ISBN 978-7-302-33745-4

定价：69.80元

DVD VIDEO 附1DVD，含教学视频、源程序等

ISBN 978-7-302-33741-6

定价：69.80元

清华大学出版社数字出版网站

WQBook 书文局泉

www.wqbook.com